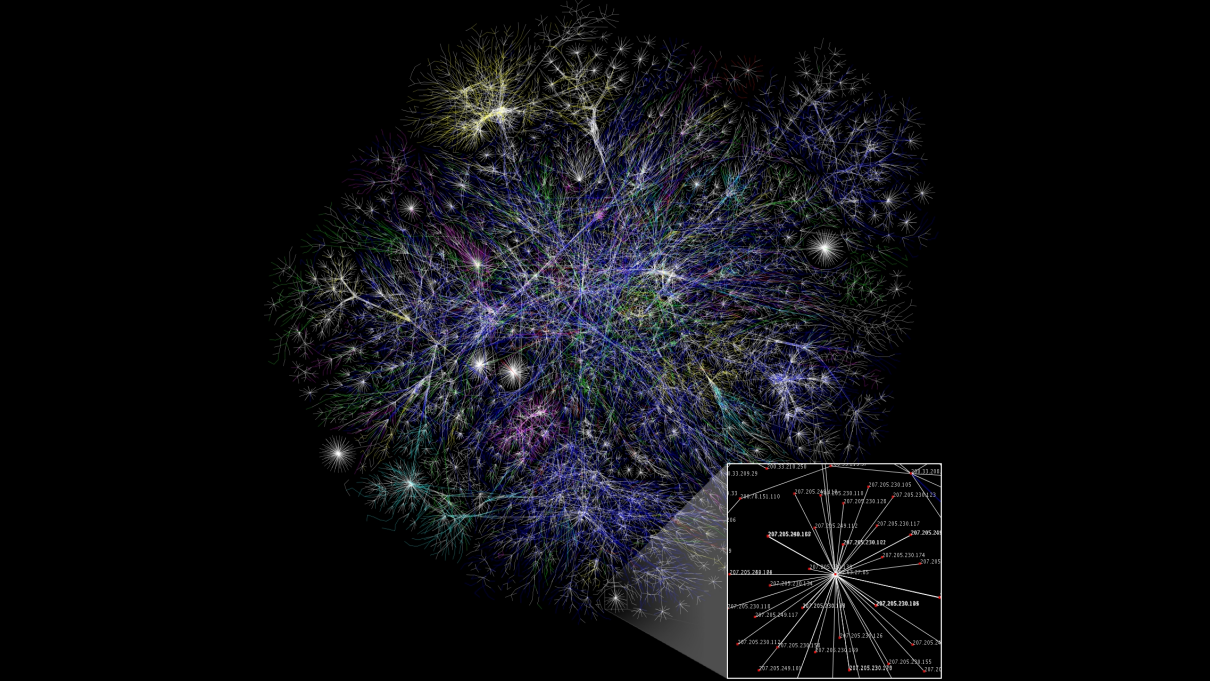
# Cloud Operating Systems

**Daniel Gruss (+ credits to Peter Lipp, Sina Karvandi (@Intel80x86), and Fabian Rauscher)**

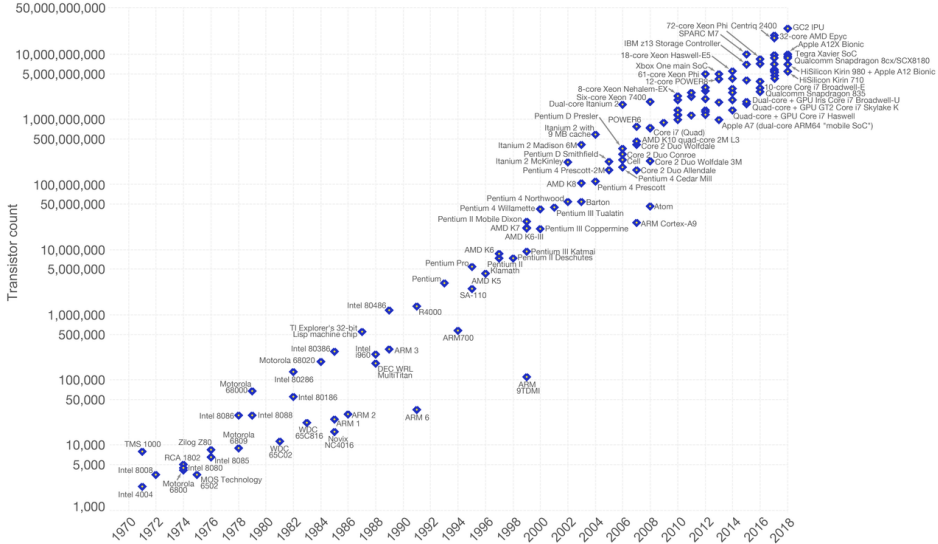
2021-03-15


207.205.230.195
207.205.230.128
207.205.230.117
207.205.230.174
207.205.230.186
207.205.230.124
207.205.230.155

# Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

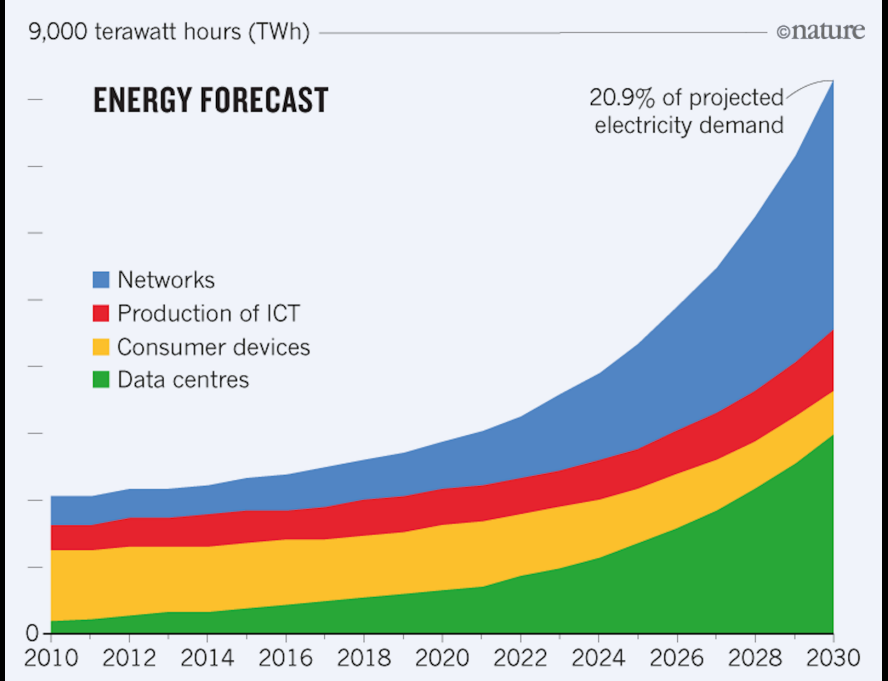
Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.




Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.


THE END IS NEAR
OF MOORE'S LAW


9,000 terawatt hours (TWh)
©nature
ENERGY FORECAST
20.9% of projected electricity demand
Networks
Production of ICT
Consumer devices
Data centres
0
2010
2012
2014
2016
2018
2020
2022
2024
2026
2028
2030

from outages blamed on cryptocurrency mining activities.[9]

Despite the fact that, in both examples, policymakers did not decide to take action because of environmental concerns, the examples illustrate how policymakers might have multiple options in putting a halt to cryptocurrency mining. Although Bitcoin might be a decentralized currency, many aspects of the ecosystem surrounding it are not. The competitive Bitcoin market drives miners to take advantage of economies of scale in lowering costs, which also makes it harder for them to operate under the radar. Large-scale miners can easily be targeted with higher electricity rates, moratoria, or, in the most extreme case, confiscation of the equipment used. Moreover, the supply chain of specialized Bitcoin mining de-

## CONCLUSION

As the price of Bitcoin rises, the negative externalities associated with Bitcoin mining increase in kind. This Commentary has shown how a simple economic model might be used to estimate the potential environmental impact of Bitcoin mining for a given Bitcoin price. These estimates reveal that the record-breaking surge in Bitcoin price at the start of 2021 could result in the network consuming as much energy as all data centers globally, with an associated carbon footprint matching London's footprint size. Beyond these environmental impacts, the production of specialized mining devices might exacerbate the global shortage of chips, which could effect the ability to work from home, the economic recovery after the COVID-19 crisis, and

Not Solve Bitcoin's Sustainability Problem. Joule 3, 893–898.

9. Arab News. (2021). Crypto-miners take down Iran electric grids, prompting crackdown. https://www.arabnews.com/node/1794836/middle-east.

10. Blandin, A., Pieters, G.C., Wu, Y., Eisermann, T., Dek, A., Taylor, S., and Njoki, D. (2020). 3rd global cryptoasset benchmarking study. https://www.jbs.cam.ac.uk/faculty-research/centres/alternative-finance/publications/3rd-global-cryptoasset-benchmarking-study/.

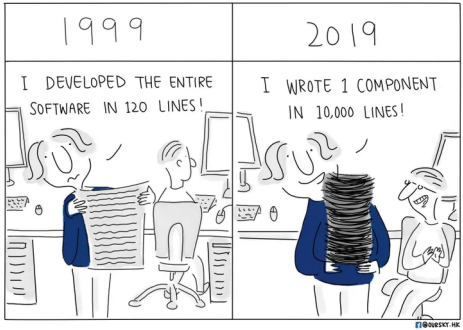
11. Stoll, C., Klaaßen, L., and Gallersdörfer, U. (2019). The Carbon Footprint of Bitcoin. Joule 3, 1647–1661.

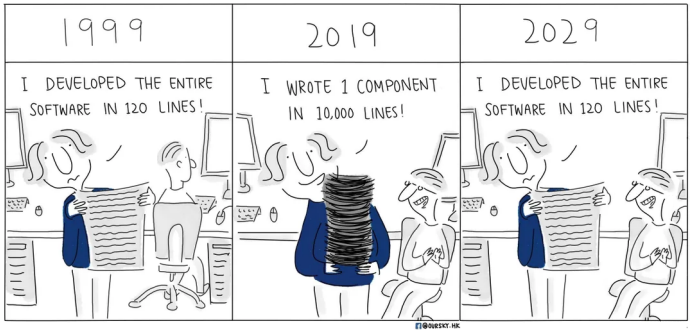
12. Gallersdörfer, U., Klaaßen, L., and Stoll, C. (2020). Energy Consumption of Cryptocurrencies Beyond Bitcoin. Joule 4, 1843–1846.

13. Jin, H., Busvine, D., and Kirton, D. (2020). Analysis: Global chip shortage threatens production of laptops, smartphones and more (Reuters).

14. de Vries, A. (2020). Bitcoin's energy consumption is underestimated: A market dynamics approach. Energy Res. Soc. Sci. 70, 101721.

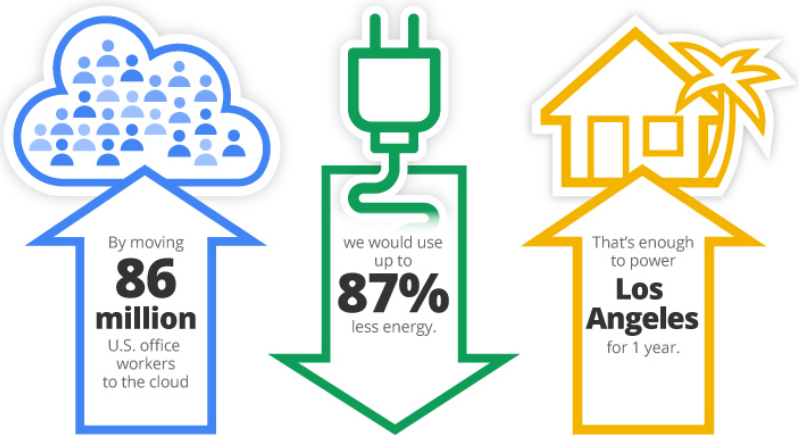

WE'RE ALL
DOOMED
memegenerator.net


1999
2019
I DEVELOPED THE ENTIRE SOFTWARE IN 120 LINES!
I WROTE 1 COMPONENT IN 10,000 LINES!
ITOSUESKY.NET


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!
FIOUESKY.NET


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 → 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

# 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!

# 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.

# 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

- Cheaper hardware: one server for one task was common

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

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

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

- 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

- 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

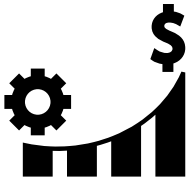
- Cheaper hardware: one server for one task was common
- most of these servers: idle time 90%
- cost issue:
  - support, maintenance
  - power consumption (operation, cooling)
  - space
- Virtualization allows consolidation
  - multiple servers on one box

# 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

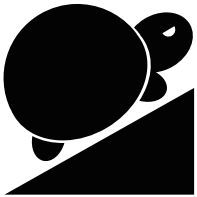
- Better hardware utilization
- Lower administration cost
- long-term operations of older applications
- lower down-times

# Advantages

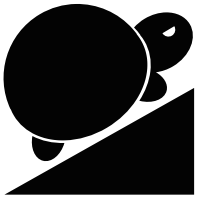
- Better hardware utilization
- Lower administration cost
- long-term operations of older applications
- lower down-times
- simple migration to more powerful hardware

# Disadvantages

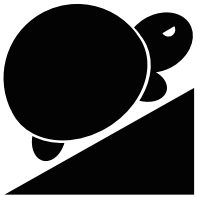
# Disadvantages



# Disadvantages

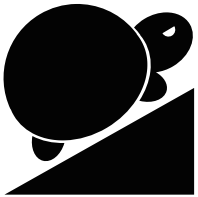


# Disadvantages



- 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 $\rightarrow$ 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 hoster (FTP access, HTTP website)

# Virtualization: Trend 1990s → 2000s

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

- OS-level Virtualization

- OS-level Virtualization
- Para-Virtualization

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

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

- 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

- Cooperation with OS: OS is aware of virtualization

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

- 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%

- 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)

- 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

- 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

- First full x86 virtualization

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

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

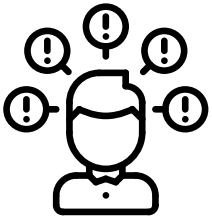
- First full x86 virtualization
- hypervisor continuously reads program code before it is executed (prescan)
- looking for relevant commands
  - change of system state

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

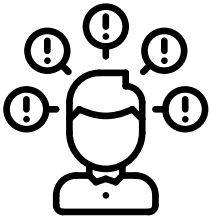
- First full x86 virtualization
- hypervisor continuously reads program code before it is executed (prescan)
- looking for relevant commands
  - change of system state
  - commands depending on CPU state
- sets breakpoint and lets OS run

# Problems virtualizing on IA-32

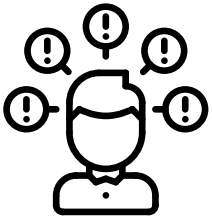
# Problems virtualizing on IA-32



# Problems virtualizing on IA-32

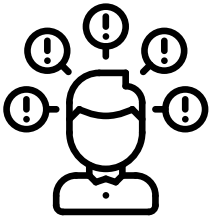


# Problems virtualizing on IA-32



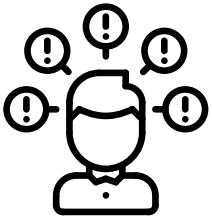
- Diverse problems were to be solved when virtualizing on IA-32:

# Problems virtualizing on IA-32



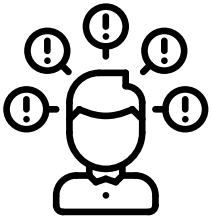
- Diverse problems were to be solved when virtualizing on IA-32:
  - Ring Problems

# Problems virtualizing on IA-32

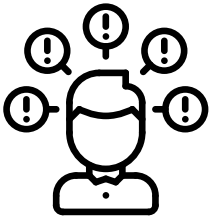


- Diverse problems were to be solved when virtualizing on IA-32:
  - Ring Problems
  - Address Space Compression

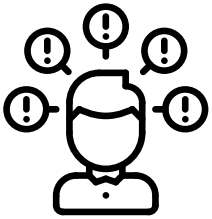
# Problems virtualizing on IA-32



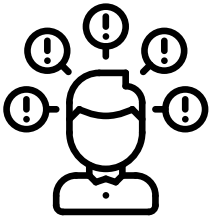
- Diverse problems were to be solved when virtualizing on IA-32:
  - Ring Problems
  - Address Space Compression
  - Non-Faulting Access to Priv. State



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



- 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



- 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 depriviledging 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 depriviledging 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)

- special commands for fast syscalls

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

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

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

- 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

- 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
  - SYSEXIT switches to hypervisor $\rightarrow$ has to emulate

- 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

- Not all state-information accessible via registers

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

- Two new operating modes:

- Two new operating modes:
  - VMX root operation

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

- 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

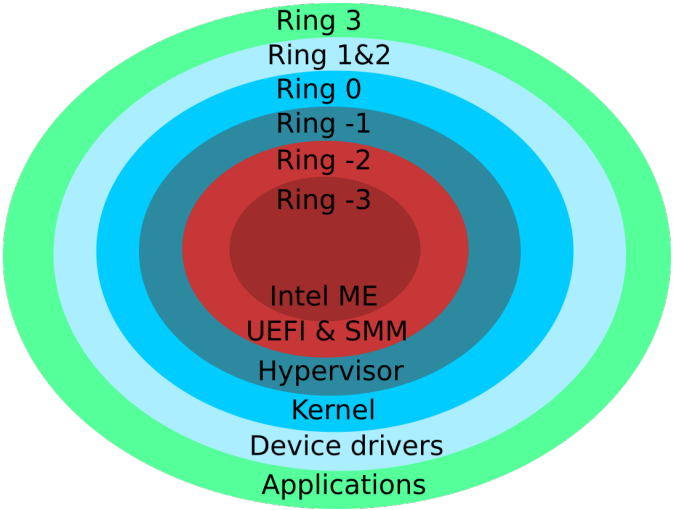
- Two new operating modes:
  - VMX root operation
    - for hypervisor
  - VMX non-root operation
    - controlled by hypervisor
    - supports VMs

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

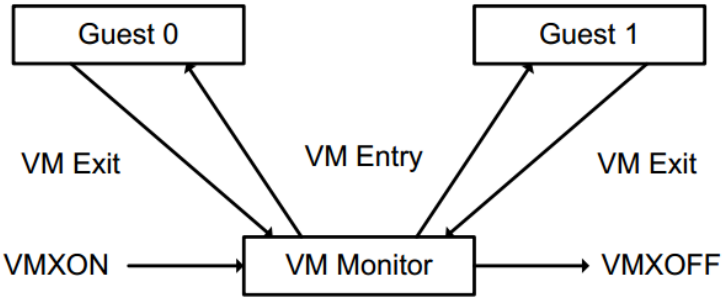
- Two new operating modes:
  - VMX root operation
    - for hypervisor
  - VMX non-root operation
    - controlled by hypervisor
    - supports VMs
- Both modes have ring 0-3
- guest can run in ring 0

- 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”

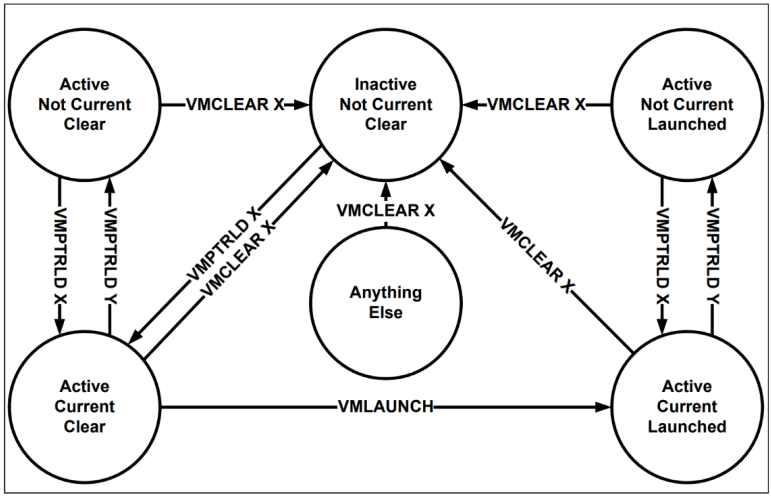
# Rings on Intel



# VMM Operation



# VMM Transitions



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

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

- VM entry: root operation → non-root operation
- VM exit: non-root operation → root operation
- VMCS: Virtual Machine Control Structure

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

- 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

- 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

- 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

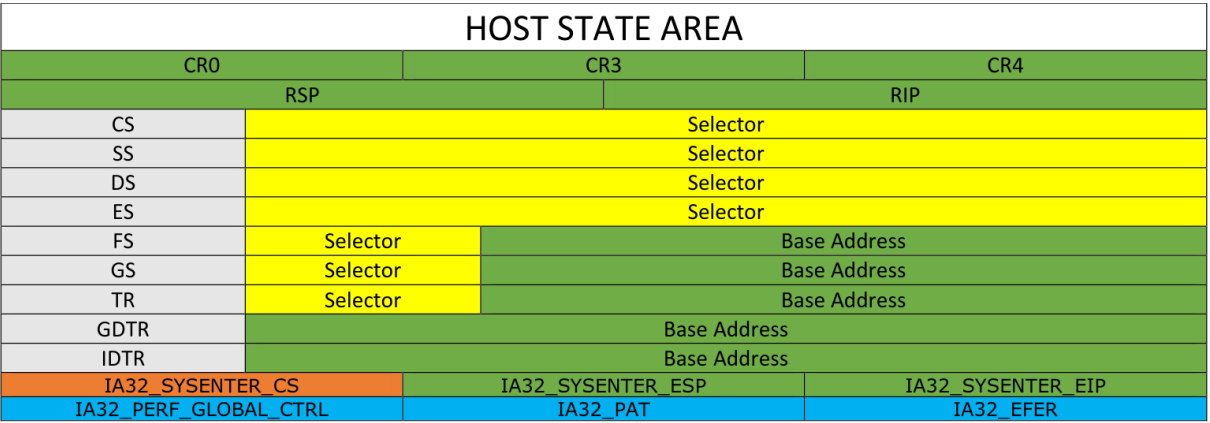
- Contains elements comprising the state of the virtual CPU of a VMCS
- VM-exit requires loading certain registers (like segment registers, CR3, IRTR...)
- GSA contains fields for these registers
- GSA contains fields for other information not readable via registers

- 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. interuptability 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

- Addressed using physical addresses

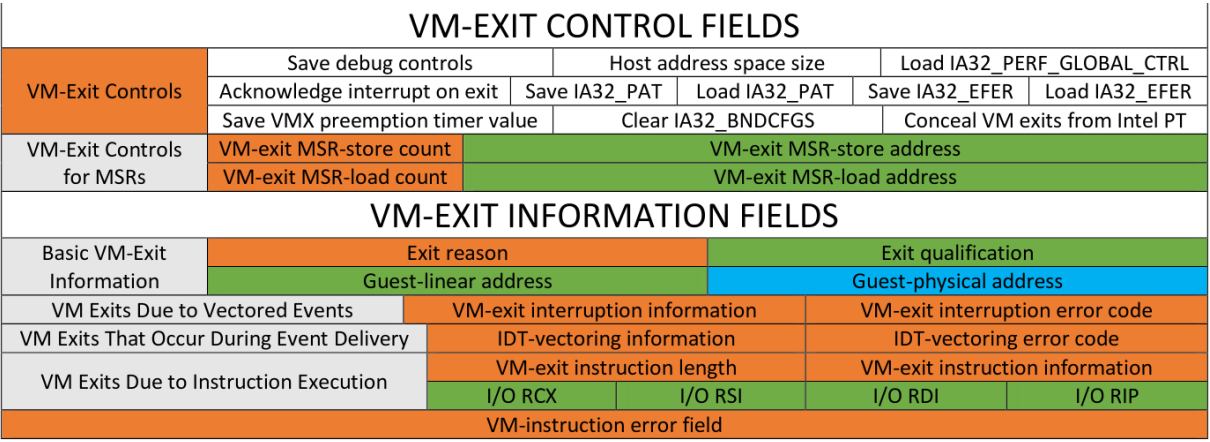
- Addressed using physical addresses
- not part of guest address space

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

- 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

- 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

- 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

- Example: MOV CR

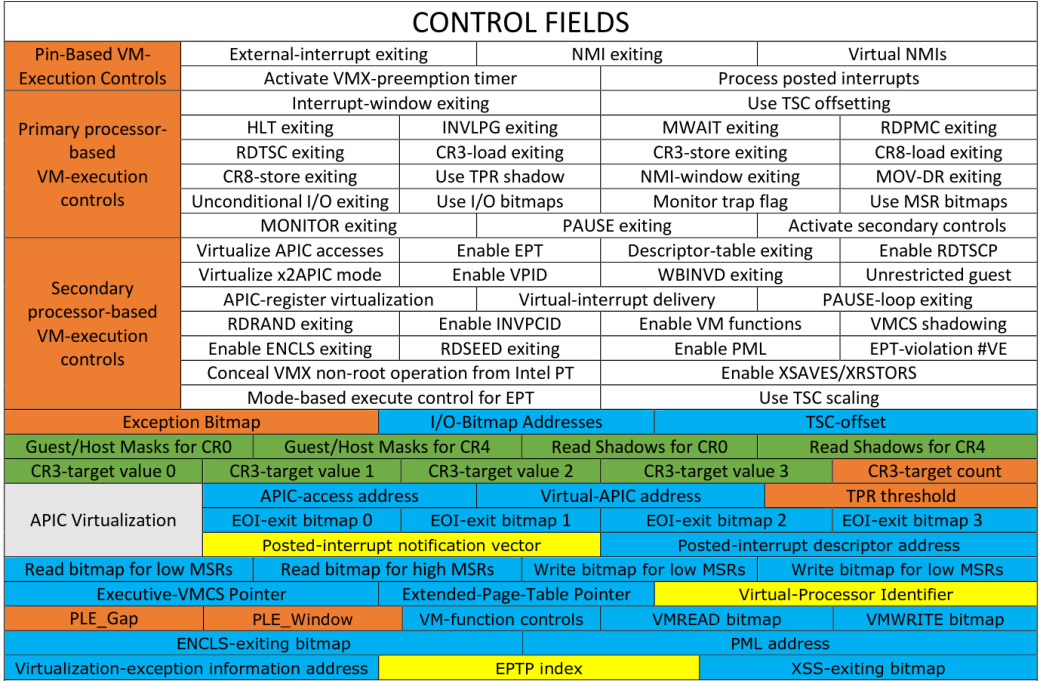
- Example: MOV CR
- Exit reason: “control register access”

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

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

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

- 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 → 2010s

The next step (≈ 2005):

- Virtualization Hardware Extensions for Intel and AMD

→ substantially lower overheads for VMs

→ better isolation

→ IaaS VMs become widely used

- Support for interrupt-virtualization

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

- 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 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

- 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

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

- 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

- Nonfaulting Access to Privileged State

- Nonfaulting Access to Privileged State
  - access raise fault into hypervisor

# Solves our problems

- Nonfaulting Access to Privileged State
  - access raise fault into hypervisor
- Hidden State

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

- Hypervisor uses virtual memory

# Hypervisor and Virtual Memory

- Hypervisor uses virtual memory
- guest OS uses virtual memory

- Hypervisor uses virtual memory
- guest OS uses virtual memory
- hardware supports pagetables

# Hypervisor and Virtual Memory

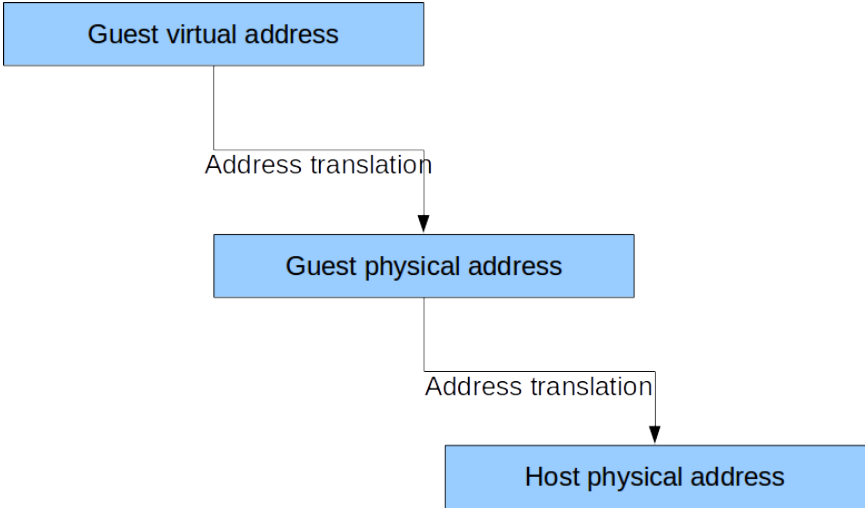
- Hypervisor uses virtual memory
- guest OS uses virtual memory
- hardware supports pagetables
- how does this work?

# Hypervisor and Virtual Memory

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

# Hypervisor and Virtual Memory

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

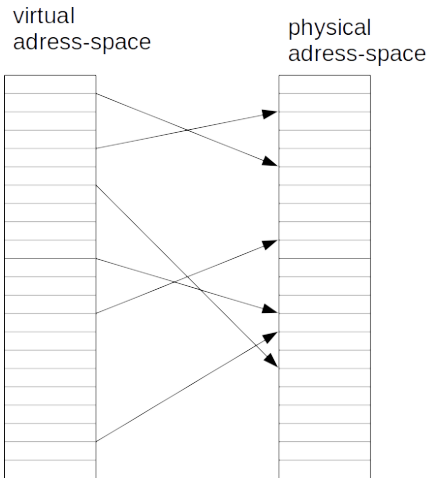

Guest virtual address
Address translation
Guest physical address
Address translation
Host physical address

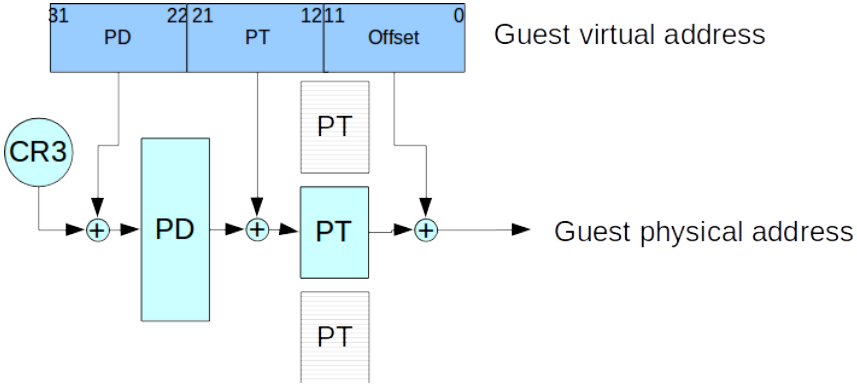
All problems in computer science can be solved by another level of indirection.

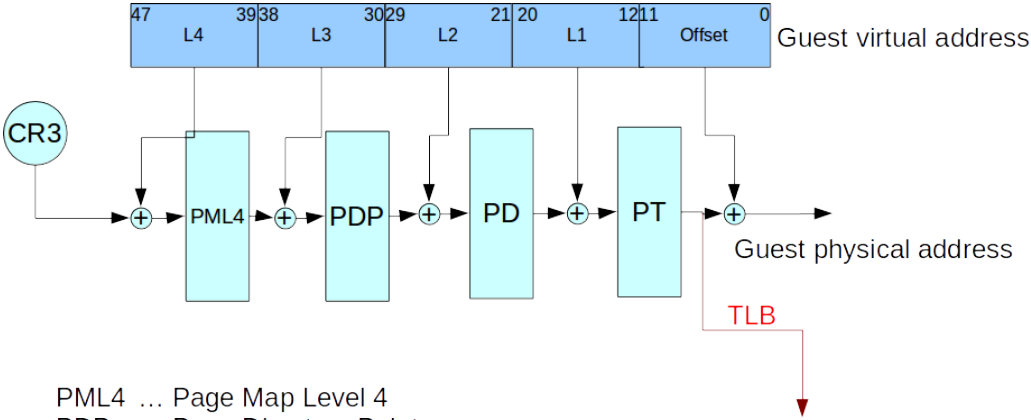
All problems in computer science can be solved by another level of indirection.

But that usually will create another problem.

David Wheeler

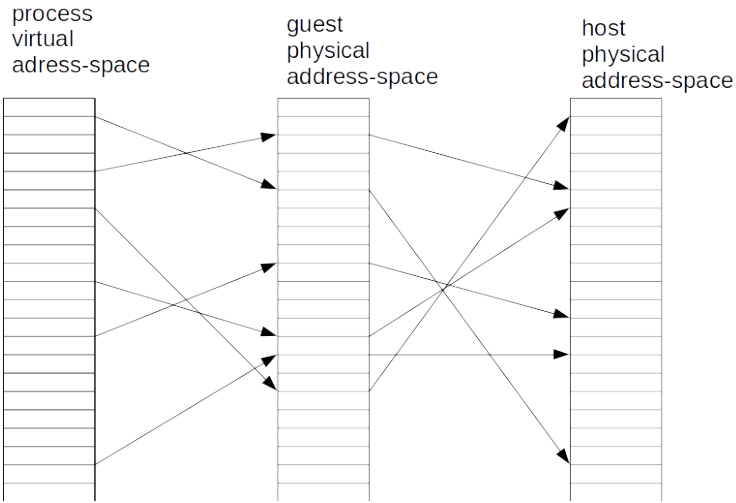

virtual
adress-space
physical
adress-space

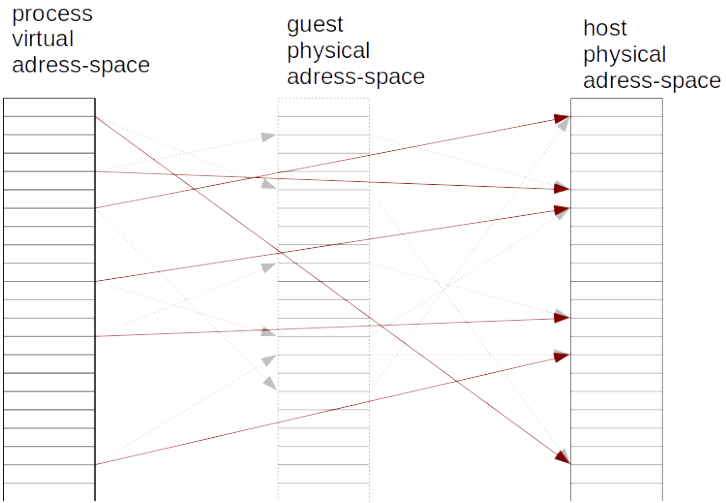

31
22
21
12
11
0
PD
PT
Offset
Guest virtual address
CR3
PD
PT
PT
PT
Guest physical address

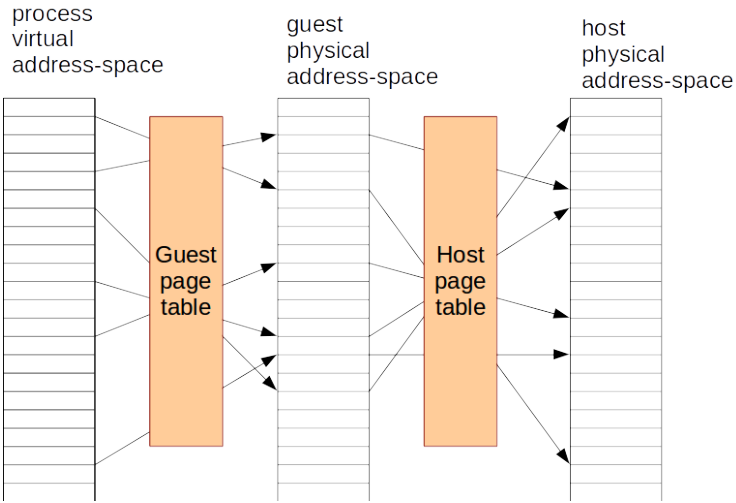


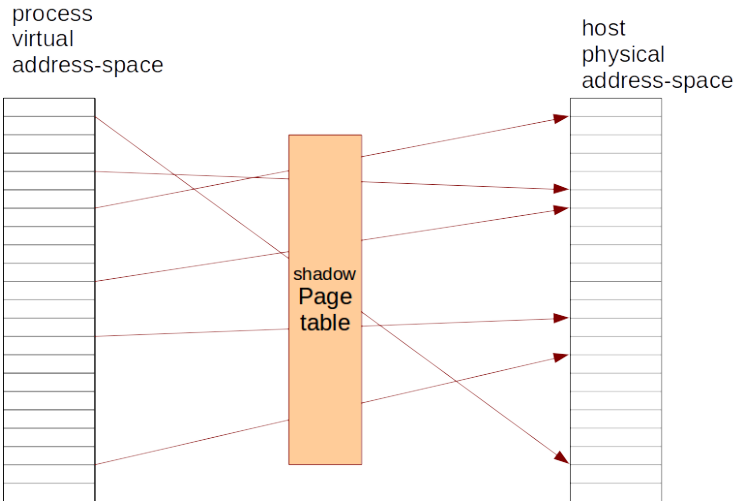

PML4 … Page Map Level 4
PDP … Page Directory Pointer
PD … Page Directory
PT … Page Table

# Combined Paging




process
virtual
adress-space
guest
physical
adress-space
host
physical
adress-space


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


process
virtual
address-space
host
physical
address-space
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

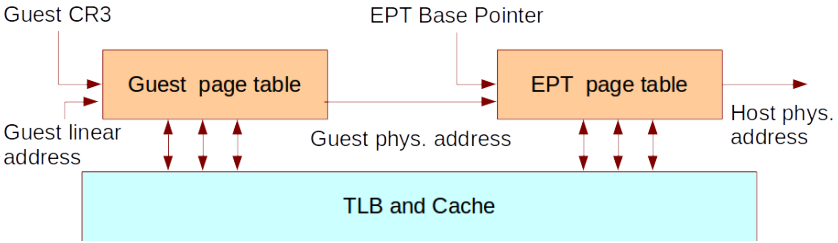
- when HW changes shadow page table
- update guest PT
  - expensive!
  - page faults caught by hypervisor

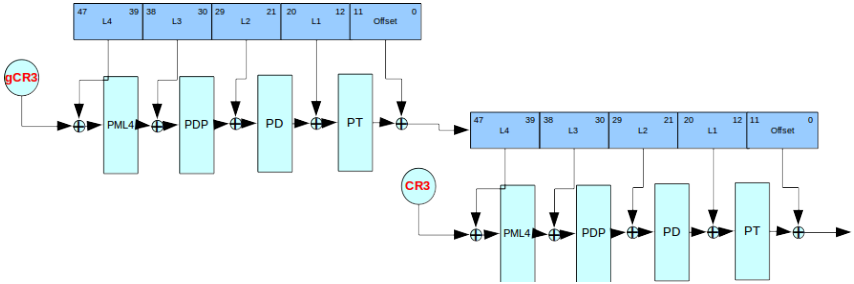
# Shadow Page Table

- 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


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



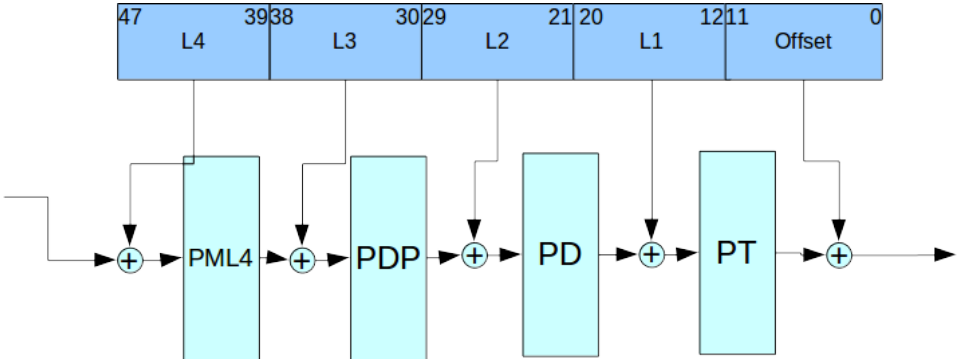

"guest page walk"

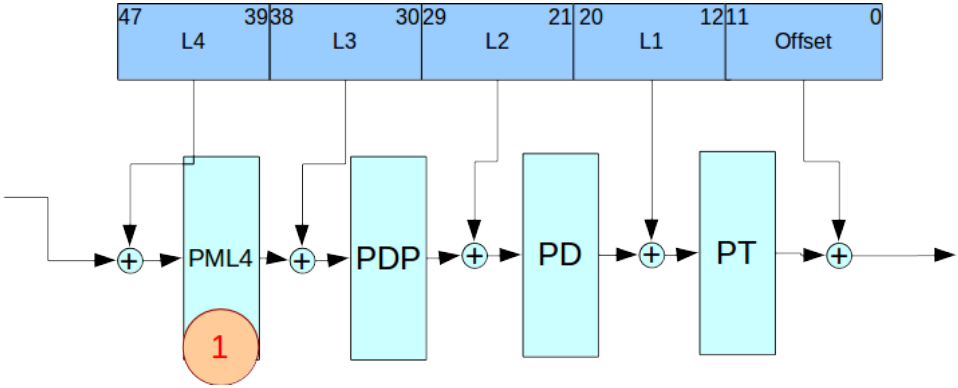
# Guest page walk

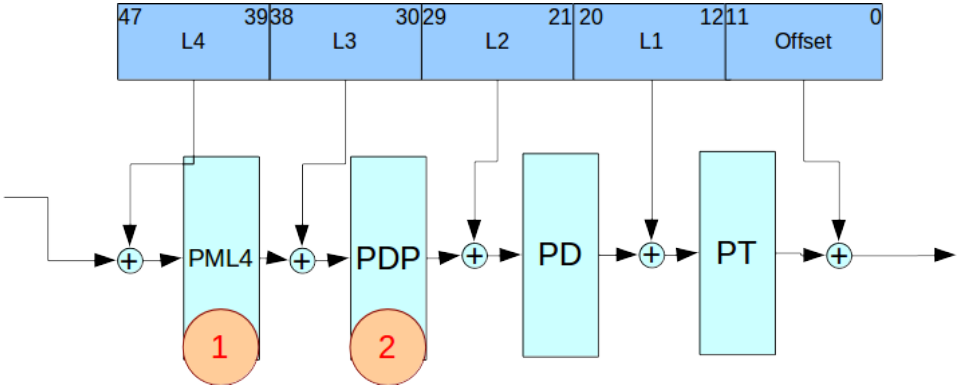
- lots of memory accesses....

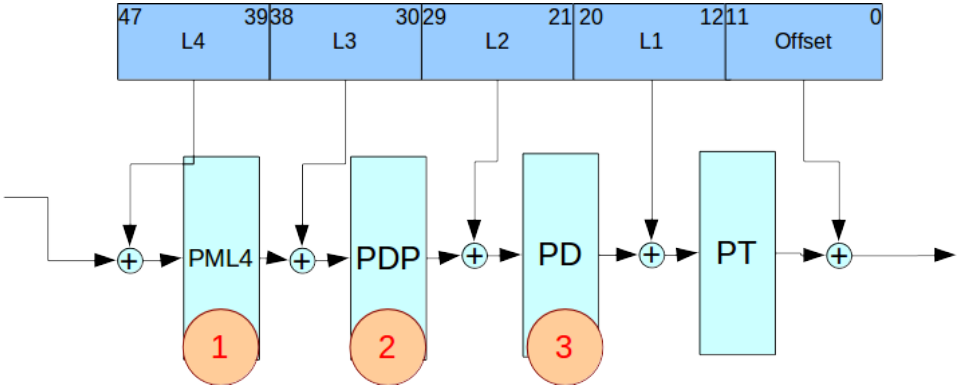
# Guest page walk

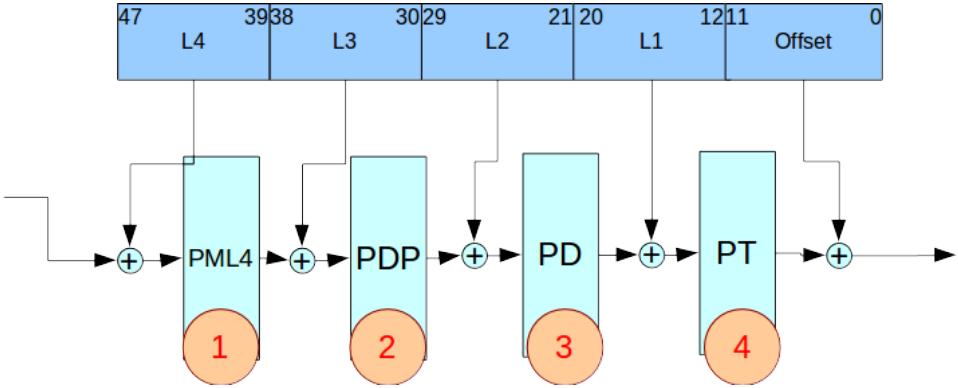
- lots of memory accesses....
- but how many exactly?

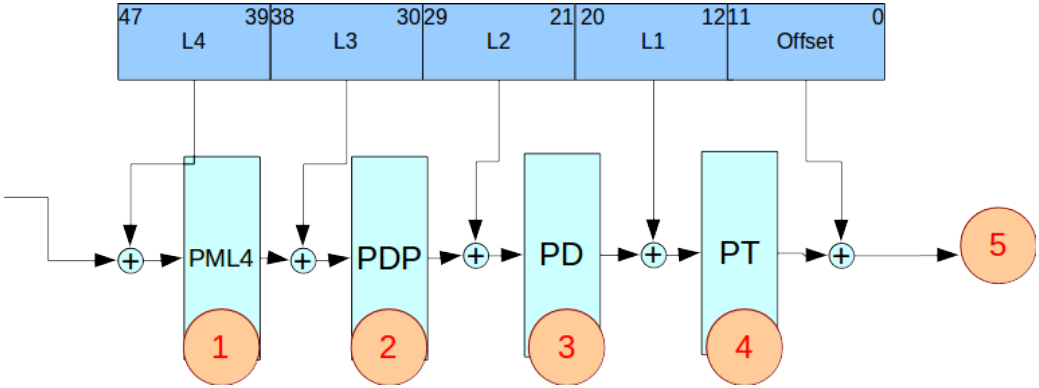

47
39
L4
38
30
L3
29
21
L2
20
12
L1
11
0
Offset
PML4
PDP
PD
PT

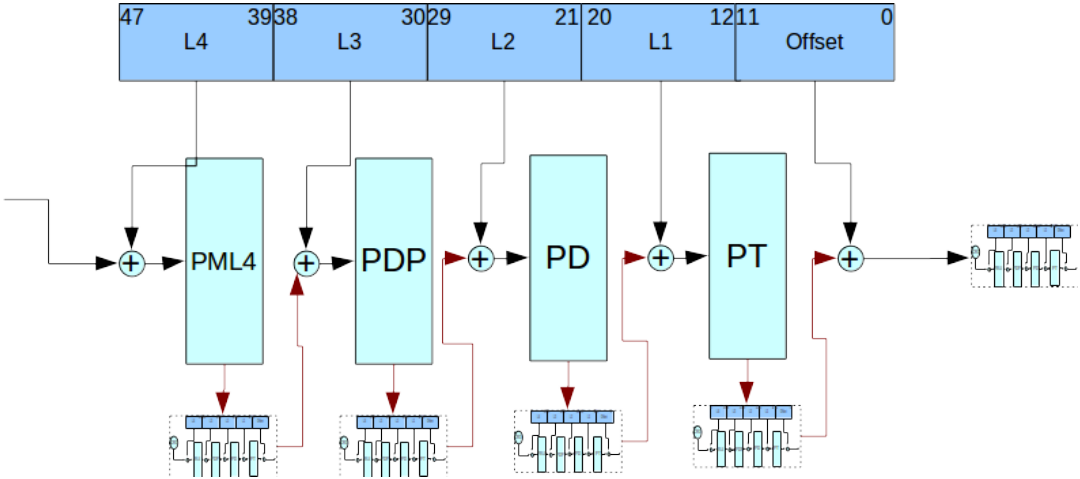

47
39
L4
38
30
L3
29
21
L2
20
12
L1
11
0
Offset
PML4
PDP
PD
PT
1


47
39
38
30
29
21
20
12
11
0
L4
L3
L2
L1
Offset
PML4
PDP
PD
PT
1
2


47
39
L4
38
30
L3
29
21
L2
20
12
L1
11
0
Offset
PML4
PDP
PD
PT
1
2
3


47
39
L4
38
30
L3
29
21
L2
20
12
L1
11
0
Offset
PML4
PDP
PD
PT
1
2
3
4


47
39
L4
38
30
L3
29
21
L2
20
12
L1
11
0
Offset
PML4
PDP
PD
PT
1
2
3
4
5


47
39
38
30
29
21
20
12
11
0
L4
L3
L2
L1
Offset
PML4
PDP
PD
PT

max. number of memory accesses per address translation

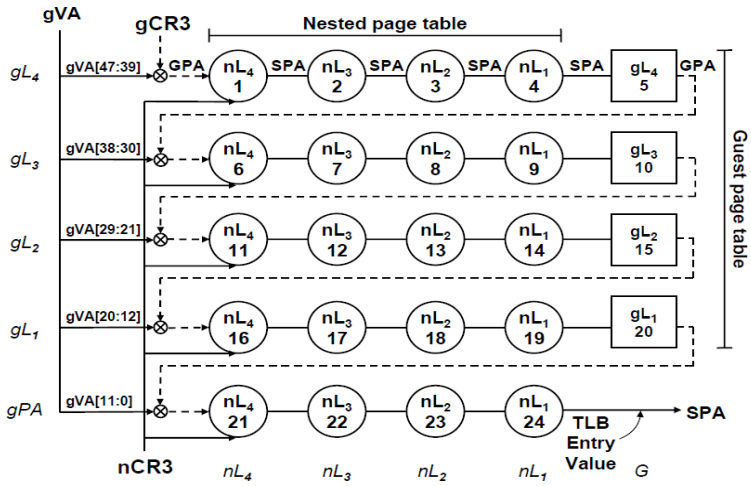
- 5 on guest level

max. number of memory accesses per address translation

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

max. number of memory accesses per address translation

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


gVA
gCR3
Nested page table
gL4
gVA[47:39]
GPA
nL4 1
SPA
nL3 2
SPA
nL2 3
SPA
nL1 4
SPA
gL4 5
GPA
gL3
gVA[38:30]
nL4 6
nL3 7
nL2 8
nL1 9
gL3 10
gL2
gVA[29:21]
nL4 11
nL3 12
nL2 13
nL1 14
gL2 15
gL1
gVA[20:12]
nL4 16
nL3 17
nL2 18
nL1 19
gL1 20
Guest page table
gPA
gVA[11:0]
nL4 21
nL3 22
nL2 23
nL1 24
SPA
TLB Entry Value
G
nCR3
nL4
nL3
nL2
nL1

# Performance

- depending on application: 3.9-4.6 times slower

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

- EPT only used if VM active

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

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

# Intel Features

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

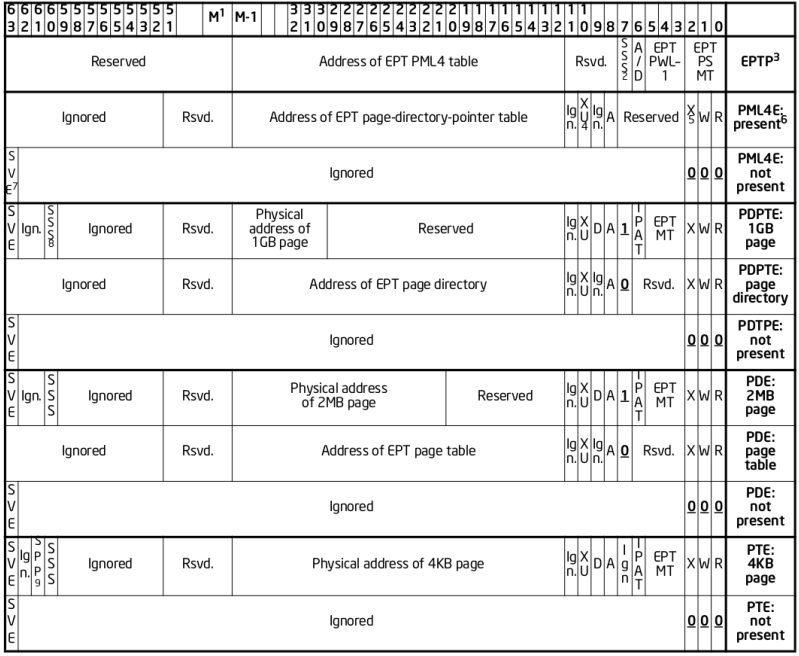



Figure 28-1. Formats of EPTP and EPT Paging-Structure Entries

1. Enable VMX via `CR4`

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

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)

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

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"

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)

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)

$\rightarrow$ 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)

$\rightarrow$ 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

Optimization

- Full virtualization often not needed

Optimization

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

# Virtualization: Trend 2010s → 2021+

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?

Optimization

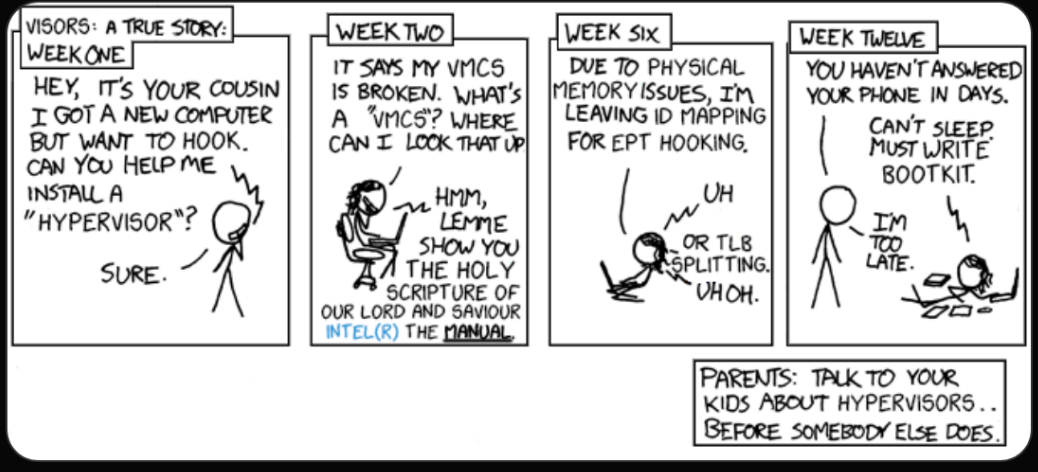
- Full virtualization often not needed
- Serverless / Edge Computing (it's still a form of cloud computing)
- Virtualization is not for free → why not skip it and just use OS level isolation?
- Context switches between processes are expensive → why not skip process isolation and just use language-level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

# This course: hardware-assisted virtualization

Cloud Operating Systems $\rightarrow$ Hardware-assisted virtualization

**cts** @gf_256 · 5. Apr. 2020

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




19 60

- Seminar-style

# CloudOS: the first time

- Seminar-style
- You code

# CloudOS: the first time

- Seminar-style
- You code
- You plan

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

Daniel Gruss

Fabian Rauscher

- 28 participants $\rightarrow$ 7 teams with each 4 participants

- 28 participants $\rightarrow$ 7 teams with each 4 participants

$\rightarrow$ send me your registration until Friday March 19

# Rules

- 28 participants $\rightarrow$ 7 teams with each 4 participants

$\rightarrow$ send me your registration until Friday March 19

- 10 Points Basic SWEB Hypervisor

# Rules

- 28 participants $\rightarrow$ 7 teams with each 4 participants

$\rightarrow$ send me your registration until Friday March 19

- 10 Points Basic SWEB Hypervisor
- 10 Points Advanced Hypervisor Feature of your choice

# Rules

- 28 participants $\rightarrow$ 7 teams with each 4 participants

$\rightarrow$ send me your registration until Friday March 19

- 10 Points Basic SWEB Hypervisor
- 10 Points Advanced Hypervisor Feature of your choice
- 10 Points Talk on Special Topic

# Rules

- 28 participants → 7 teams with each 4 participants

→ send me your registration until Friday March 19

- 10 Points Basic SWEB Hypervisor
- 10 Points Advanced Hypervisor Feature of your choice
- 10 Points Talk on Special Topic
- points based on exercise interview

# Grading

- 26 of 30 points $\rightarrow$ 1

- 26 of 30 points $\rightarrow$ 1
- 22 of 30 points $\rightarrow$ 2

# Grading

- 26 of 30 points $\rightarrow$ 1
- 22 of 30 points $\rightarrow$ 2
- 18 of 30 points $\rightarrow$ 3

- 26 of 30 points $\rightarrow$ 1
- 22 of 30 points $\rightarrow$ 2
- 18 of 30 points $\rightarrow$ 3
- 15 of 30 points $\rightarrow$ 4

- Implementation Deadline **11.6.**

- Implementation Deadline **11.6.**
- VMX works

- Implementation Deadline **11.6.**
- VMX works
- EPT works

- Implementation Deadline **11.6.**
- VMX works
- EPT works
- a virtualized SWEB boots and is usable

- Propose an advanced hypervisor feature you will support

# Task 2: Advanced Hypervisor Feature

- Propose an advanced hypervisor feature you will support
- 280 characters description

# Task 2: Advanced Hypervisor Feature

- Propose an advanced hypervisor feature you will support
- 280 characters description
- Feature Plan Deadline **7.5.**

# Task 2: Advanced Hypervisor Feature

- Propose an advanced hypervisor feature you will support
- 280 characters description
- Feature Plan Deadline **7.5.**
- What? Anything you like

- Propose an advanced hypervisor feature you will support
- 280 characters description
- Feature Plan Deadline **7.5.**
- What? Anything you like
  - running multiple VMs

# Task 2: Advanced Hypervisor Feature

- Propose an advanced hypervisor feature you will support
- 280 characters description
- Feature Plan Deadline **7.5.**
- What? Anything you like
  - running multiple VMs
  - page deduplication across VMs

# Task 2: Advanced Hypervisor Feature

- Propose an advanced hypervisor feature you will support
- 280 characters description
- Feature Plan Deadline **7.5.**
- What? Anything you like
  - running multiple VMs
  - page deduplication across VMs
  - EPT hooking

# Task 2: Advanced Hypervisor Feature

- Propose an advanced hypervisor feature you will support
- 280 characters description
- Feature Plan Deadline **7.5.**
- What? Anything you like
  - running multiple VMs
  - page deduplication across VMs
  - EPT hooking
  - shadow page tables

# Task 2: Advanced Hypervisor Feature

- Propose an advanced hypervisor feature you will support
- 280 characters description
- Feature Plan Deadline **7.5.**
- What? Anything you like
  - running multiple VMs
  - page deduplication across VMs
  - EPT hooking
  - shadow page tables
  - virtualize the running system to hook (and alter) instructions

# Task 2: Advanced Hypervisor Feature

- Propose an advanced hypervisor feature you will support
- 280 characters description
- Feature Plan Deadline **7.5.**
- What? Anything you like
  - running multiple VMs
  - page deduplication across VMs
  - EPT hooking
  - shadow page tables
  - virtualize the running system to hook (and alter) instructions
  - virtualization of any hardware devices (many options)

# Task 2: Advanced Hypervisor Feature

- Propose an advanced hypervisor feature you will support
- 280 characters description
- Feature Plan Deadline **7.5.**
- What? Anything you like
  - running multiple VMs
  - page deduplication across VMs
  - EPT hooking
  - shadow page tables
  - virtualize the running system to hook (and alter) instructions
  - virtualization of any hardware devices (many options)
  - nested virtualization

- Propose an advanced hypervisor feature you will support
- 280 characters description
- Feature Plan Deadline **7.5.**
- What? Anything you like
  - running multiple VMs
  - page deduplication across VMs
  - EPT hooking
  - shadow page tables
  - virtualize the running system to hook (and alter) instructions
  - virtualization of any hardware devices (many options)
  - nested virtualization
- Implementation Deadline **11.6.**

# Task 3: Talk on Special Topic

- 12.4., 26.4., 10.5.

# Task 3: Talk on Special Topic



- 12.4., 26.4., 10.5.
- 2 talks each

# Task 3: Talk on Special Topic

- 12.4., 26.4., 10.5.
- 2 talks each
- 20-40 minutes (=5-10 minutes per participant) + Q&A

# Task 3: Talk on Special Topic



- 12.4., 26.4., 10.5.
- 2 talks each
- 20-40 minutes (=5-10 minutes per participant) + Q&A
- Register until **29.3.** with talk topic and date

# Timeline

- 15.03. Introduction Lecture

# Timeline

- 15.03. Introduction Lecture
- 19.03. **Deadline**: Group Registration

# Timeline

- 15.03. Introduction Lecture
- 19.03. **Deadline**: Group Registration
- 22.03. Hypervisor Implementation Basics

# Timeline

- 15.03. Introduction Lecture
- 19.03. **Deadline**: Group Registration
- 22.03. Hypervisor Implementation Basics
- 29.03. **Deadline**: Talk Registration

# Timeline

- 15.03. Introduction Lecture
- 19.03. **Deadline**: Group Registration
- 22.03. Hypervisor Implementation Basics
- 29.03. **Deadline**: Talk Registration
- 12.04. Student Presentations

# Timeline

- 15.03. Introduction Lecture
- 19.03. **Deadline**: Group Registration
- 22.03. Hypervisor Implementation Basics
- 29.03. **Deadline**: Talk Registration
- 12.04. Student Presentations
- 26.04. Student Presentations

# Timeline

- 15.03. Introduction Lecture
- 19.03. **Deadline**: Group Registration
- 22.03. Hypervisor Implementation Basics
- 29.03. **Deadline**: Talk Registration
- 12.04. Student Presentations
- 26.04. Student Presentations
- 07.05. **Deadline**: Feature Plan Deadline

# Timeline

- 15.03. Introduction Lecture
- 19.03. **Deadline**: Group Registration
- 22.03. Hypervisor Implementation Basics
- 29.03. **Deadline**: Talk Registration
- 12.04. Student Presentations
- 26.04. Student Presentations
- 07.05. **Deadline**: Feature Plan Deadline
- 10.05. Student Presentations

# Timeline

- 15.03. Introduction Lecture
- 19.03. **Deadline**: Group Registration
- 22.03. Hypervisor Implementation Basics
- 29.03. **Deadline**: Talk Registration
- 12.04. Student Presentations
- 26.04. Student Presentations
- 07.05. **Deadline**: Feature Plan Deadline
- 10.05. Student Presentations
- 11.06. **Deadline**: Implementation Deadline

# Timeline

- 15.03. Introduction Lecture
- 19.03. **Deadline**: Group Registration
- 22.03. Hypervisor Implementation Basics
- 29.03. **Deadline**: Talk Registration
- 12.04. Student Presentations
- 26.04. Student Presentations
- 07.05. **Deadline**: Feature Plan Deadline
- 10.05. Student Presentations
- 11.06. **Deadline**: Implementation Deadline
- 14.06. Exercise Interviews