Virtualization

Universiteit Leiden
The Netherlands

Definitie

- *"A virtual machine is taken to be an efficient, isolated duplicate of the real machine".*

- Popek en Goldberg, 1974.

A little history …

- Jaren '50 – '60.
- Computers waren groot en kostbaar. Er was 1 computer voor een grote groep mensen.
- Batch processing.
- Met “peripheral computers” konden jobs worden geprepareerd.
- Gebruikers waren geïsoleerd van de echte machine: development en testing was frustrerend.

A little history … (2)

- De machines werden steeds krachtiger.
- Het werd mogelijk het batch werk tijdelijk te onderbreken om een interactief commando uit te voeren.

A little history … (3)

- Begin 1960: eerste werk aan “time sharing” werd gedaan aan de MIT.
- CTSS: Compatible Time-Sharing System.
- Subset van de machine kon worden gebruikt door batch jobs.
- Supervisor kon deze batch jobs tijdelijk onderbreken.

A little history … (4)

- Eerste implementatie “virtual machine” concept: CP/CMS in 1964.
- “Second-generation time-sharing system”.
- CP-40/CMS draaide op een IBM System/360 Model 40, speciaal aangepast met een device voor dynamic address translation.
  - De “DAT box”.

A little history... (5)

- CP-67/CMS draaide op een IBM System/360 Model 67, deze machine had address translation ingebouwd.
- Hieruit kwam weer VM/370 voort voor IBM System/370.
- En later OS/390, z/OS.

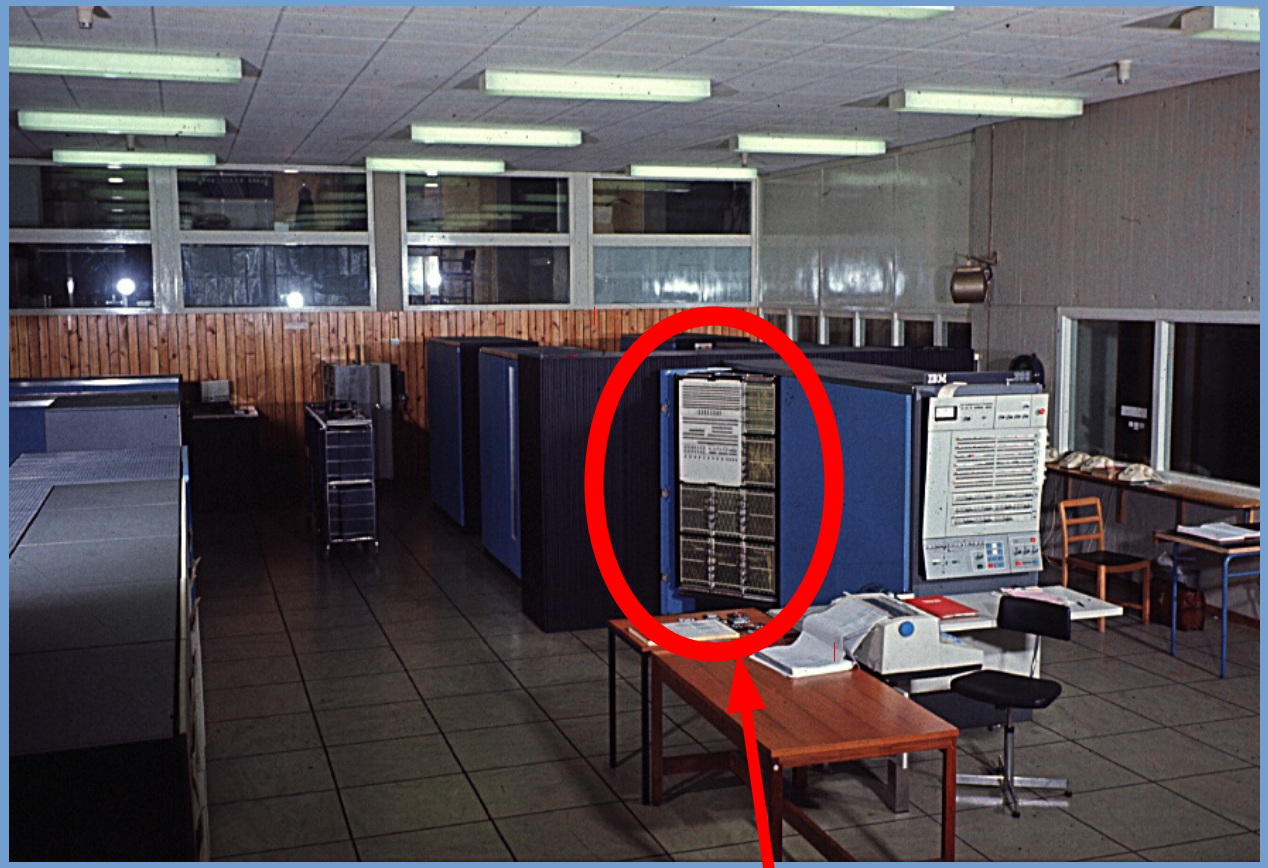
A little history … (6)

Bron: http://en.wikipedia.org/wiki/File:Bundesarchiv_B_145_Bild-F038812-0014,_Wolfsburg,_VW_Autowerk.jpg

A little history … (7)

Bron: http://en.wikipedia.org/wiki/File:IBM_System360_Model_30.jpg

A little history … (8)



Bron: http://history.cs.ncl.ac.uk/anniversaries/40th/images/ibm360_672/slide07.jpg

CP/CMS

- CP: Control Program
  - Operating system dat meerdere kopieën simuleert van de machine waarop het zelf draait.
- CMS: Conversational Monitor System
  - Interactief OS voor 1 gebruiker.
  - Kon onder CP draaien of direct op de hardware!

CP/CMS (2)

- Elke gebruiker kon een eigen instance van CMS krijgen.
- RSCS: Remote Spooling and Communications Subsystem.
  - Batch processing.

CP/CMS (3)

- CP maakte gebruik van trap-and-emulate strategie.
- Guest OS draaide in “user mode”.
- Als guest een privileged instructie uitvoerde, vond er een “trap” plaats.
- CP nam het over en emuleerde de privileged instructie.

CP/CMS (4)

- Hoe werkte dat met devices?
  - CP kon bijvoorbeeld de magnetic tape drive met een virtual machine verbinden.
  - Virtual disks werden naar disk volumes gemapped.
  - Virtuele kaartpuncher en -lezer en printer werden via een spool-systeem geimplementeerd.

Retrospect

- Toekomstvisie in 1981:

  " As larger, faster, and less expensive machines become available, the software systems supporting interconnected virtual machines can move smoothly to collections of real machines."

- R. J. Creasy, 1981.

2016

- Voor lange tijd is er inderdaad een beweging geweest van grote mainframes naar collecties van “echte” computers.
- In de laatste jaren zien we echter de tegengestelde beweging, verschillende taken gedaan door “echte” computers worden middels virtual machines weer gedaan door 1 computer.

Waarom weer virtual machines?

- 1. Kosten.
  - 10 machines die een lichte taak hebben kunnen gerust worden samengevoegd tot 1.
  - 1 zware machine gebruikt in de regel minder energie dan 10 kleine en vereist ook minder ruimte (ruimte in een data center is duur!) en koelingcapaciteit.

Waarom weer virtual machines? (2)

- 2. Flexibiliteit
  - Mogelijkheid voor “live migrations”
  - Het is makkelijker een machine te “verhuren” zonder de fysieke machine opnieuw te installeren.
  - VPS oplossingen en “cloud systems” als Amazon EC2 en Google Compute Engine.

Waarom weer virtual machines? (3)

- 3. Security
  - Taken compleet isoleren.
  - Sandboxing.

Waarom weer virtual machines? (4)

- 4. Development
  - Eenvoudige manier om verschillende operating systems op 1 machine te draaien voor testing e.d.

Waarom weer virtual machines? (5)

- Opgelet: soms is er ook sprake van “virtual machines” in het kader van migratie en backwards compatibility.
- Vaak betreft dit het draaien van software geschreven voor een andere architectuur.
- Daardoor valt dit eigenlijk onder emulatie!

Waarom weer virtual machines? (6)

- Migratie / backwards compability
  - Migratie Apple van PowerPC naar Intel CPUs. Intel Macs konden oude PowerPC Mac software draaien.
  - Microsoft VirtualPC
  - Nintendo “Virtual Console”

Formele definitie

- We zagen al eerder de definitie van een virtual machine:

  "A virtual machine is taken to be an efficient, isolated duplicate of the real machine".

- Virtual machines draaien onder een “virtual machine monitor” (VMM), tegenwoordig ook hypervisor genoemd.

Formele definitie (2)

- Popek en Goldberg definieerden ook 3 karakteristieken van een VMM:
  1. VMM schept een omgeving voor programma's welke identiek is aan dat van de originele machine.
  2. Programma's draaien binnen deze omgeving in het slechtste geval maar iets minder snel.
  3. VMM heeft volledige controle over alle system resources.

Formele definitie (3)

- Wanneer mag iets een VMM worden genoemd?
- Volgens Popek en Goldberg is dat wanneer een control program de volgende 3 eigenschappen heeft:

Formele definitie (4)

1. *Efficiency*: alle normale instructies worden direct door de hardware uitgevoerd, zonder enige interventie van het control program.
2. *Resource control*: een programma kan zichzelf geen extra resources toekennen, als dit wordt geprobeerd moet de allocator van het control program worden aangeroepen.

Formele definitie (4)

3. *Equivalence*: elk programma dat draait terwijl een control program aanwezig is, wordt uitgevoerd op een manier die niet te onderscheiden is van als het control program niet aanwezig zou zijn, en het programma kan alle privileged instructies uitvoeren zoals de programmeur had bedoeld.

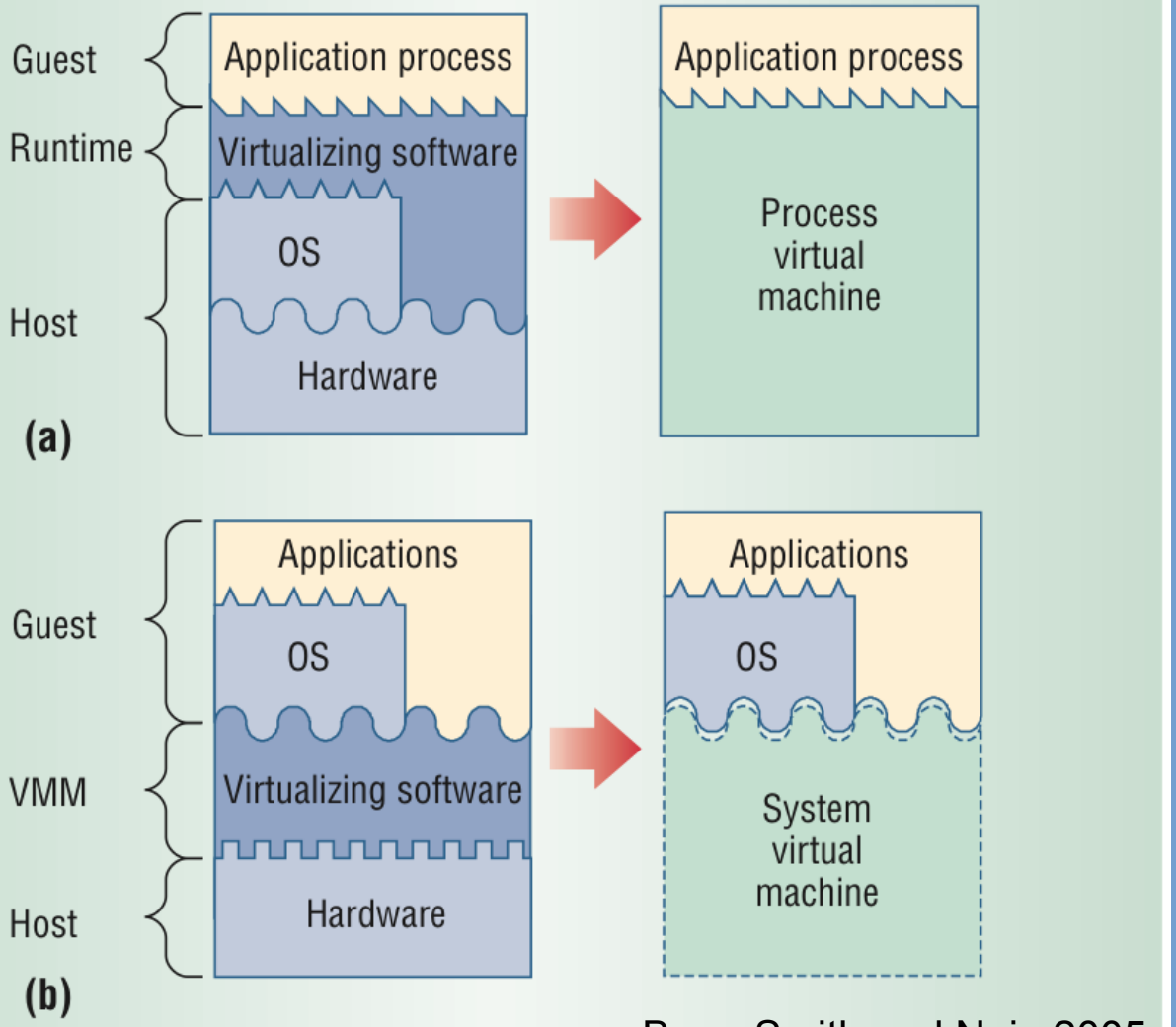
Process vs. System VM

- Process VM: virtual platform dat een individueel proces uitvoert.
  - Virtual machine wordt gemaakt bij start van proces, bij einde van proces weer afgebroken.
  - Het programma kan niet direct bij hardware of privileged instructies, dit moet via library/system calls.
  - Voorbeeld: Java Virtual Machine (JVM), .NET virtual machine, emulatoren.
  - “Runtime software”.

Process vs. System VM (2)

- System VM: voorziet in een omgeving waarin een operating system samen met verschillende user processen kan draaien.
  - Verzorgt toegang tot virtuele hardware voor een gast operating system.
  - “Virtual machine monitor” volgens definitie Popek/Goldberg.

Process vs. System VM (3)



Bron: Smith and Nair, 2005.

Virtualiseren x86 architectuur

- De x86 architectuur kan niet op de klassieke wijze worden gevirtualiseerd.
- Waarom niet?
  - “real” en “virtual 8086” modes.
  - Een guest OS kan zien dat hij niet in privileged mode draait. Dus in principe is het execution environment niet identiek aan “bare metal”.
  - Bij het uitvoeren van privileged instructies in user mode worden geen traps gegenereerd.

Virtualiseren x86 architectuur (2)

- Mogelijke oplossing: interpreteren.
  - Traag! Criterium 2 geldt niet.

- Oplossing: binary translation.
- Volledige x86 instructie set (met alle privileged instructies) wordt vertaald naar instruction set met alleen user-mode instructies.

Virtualiseren x86 architectuur (3)

- Ander probleem: we kunnen een guest geen toegang geven tot de MMU.
- De MMU kan maar met 1 niveau van VA → PA vertaling omgaan.
- Hoe lossen we dit op?

Virtualiseren x86 architectuur (4)

- We markeren de page table van de guest als read-only en vangen elke schrijfactie af.
- Voor de “gVA” → “gPA” vertaling die de guest wilde toevoegen zorgen we voor de bijbehorende gPA → hVA → hPA vertaling (shadow).
- De mapping gVA → hPA slaan we op in de page table gebruikt door de MMU.
- Deze techniek heet “shadow paging”.
- Kostbaar!

Hardware extensies voor x86 virtualisatie

- Intel en AMD hebben rond 2005 – 2006 extensies aan de CPU architectuur toegevoegd.
  - Intel VT-x (voorheen Vanderpool)
  - AMD-V (voorheen Secure Virtual Machine (SVM))

Hardware extensies voor x86 virtualisatie (2)

- Virtual Machine Control Block (VMCB), voor het bijhouden van control state met gedeelte van de state van de guest virtual CPU.
- Een nieuwe execution mode (of “ring”): “guest mode”. Hierin kunnen ook privileged guest instructies worden uitgevoerd.
- “vmrun” instructie schakelt om van host mode naar guest mode en laadt de guest state vanuit het VMCB.

Hardware extensies voor x86 virtualisatie (3)

- “exit” uit guest mode als een bepaalde conditie zich voordoet.
- Voorbeelden:
  - Exit on HLT
  - Exit on CR3 load/store
  - Page fault
  - Device access / I/O
- VMM kan dan ingrijpen en/of een gedeelte emuleren.

x86 virtualisatie performance

- “vmrun” en “exit” zijn kostbare operaties.
- Een guest dat geen exit operaties doet draait op “native speed”. Maar deze guest kan dan geen I/O doen.
- Belangrijke optimalisatie: het aantal “exit” operaties zo klein mogelijk maken.

Software VMM vs. Hardware VMM

- Aan de hand van een 2006 paper van VMware.
- Voordeel software VMM:
  - Met binary translation kunnen traps worden vervangen met snellere function calls, ipv. een vmrun exit.
- Voordeel hardware VMM:
  - Er hoeft geen vertaling plaats te vinden en system calls draaien zonder dat de VMM aan de pas komt.

Software VMM vs. Hardware VMM (2)

- Compute-intensive benchmarks draaien op native speed voor beide typen VMMs.
- Hoe meer privileged instructies, hoe meer overhead.
- Het blijkt dat software VMMs sneller zijn dan hardware VMMs (in 2006).
- Page faults en veranderingen aan de page tables kunnen in een software VMM sneller worden afgehandeld.
- Page fault traps goedkoper dan vmrun/exit roundtrip.

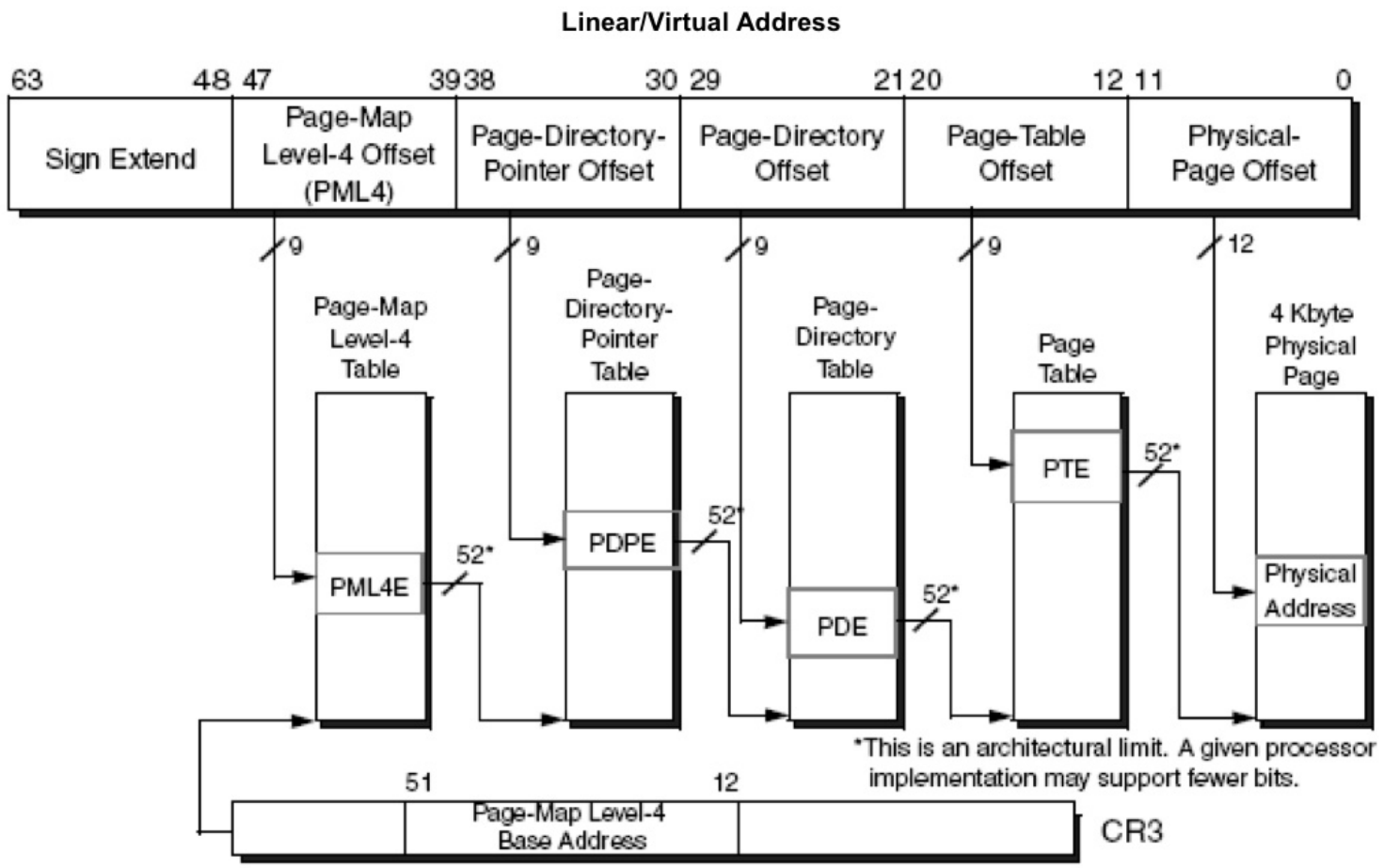
Verbeteringen architectuur

- VT-x / AMD-V niet genoeg.
- Microarchitectuur verder verbeteren, zodat vmrun/exit minder cycles in beslag nemen.
- Hardware MMU support: AMD “nested paging”, Intel “Extended Page Tables”
  - Introductie rond 2008.
- Er is ook betere ondersteuning verschenen voor I/O virtualization (VT-d / IOMMU).

Nested paging

- Idee: VMM houdt een “nested page table” bij, zodat de guest zonder traps zijn page table kan aanpassen.
- Kostbare interventies door VMMs zijn niet meer nodig.
- MMU doorloopt “guest page table” en “host page table”.

Nested paging (2)



Bron: AMD-V Nested Paging White Paper, AMD Inc., 2008.

Nested paging (3)

- Guest heeft een page table “gPT”, deze beeldt guest linear addresses af op guest physical addresses.
  - gVA → gPA
- Hypervisor zorgt voor een nested page table “nPT” deze beeldt guest physical pages af op host physical addresses.
  - gPA → hPA

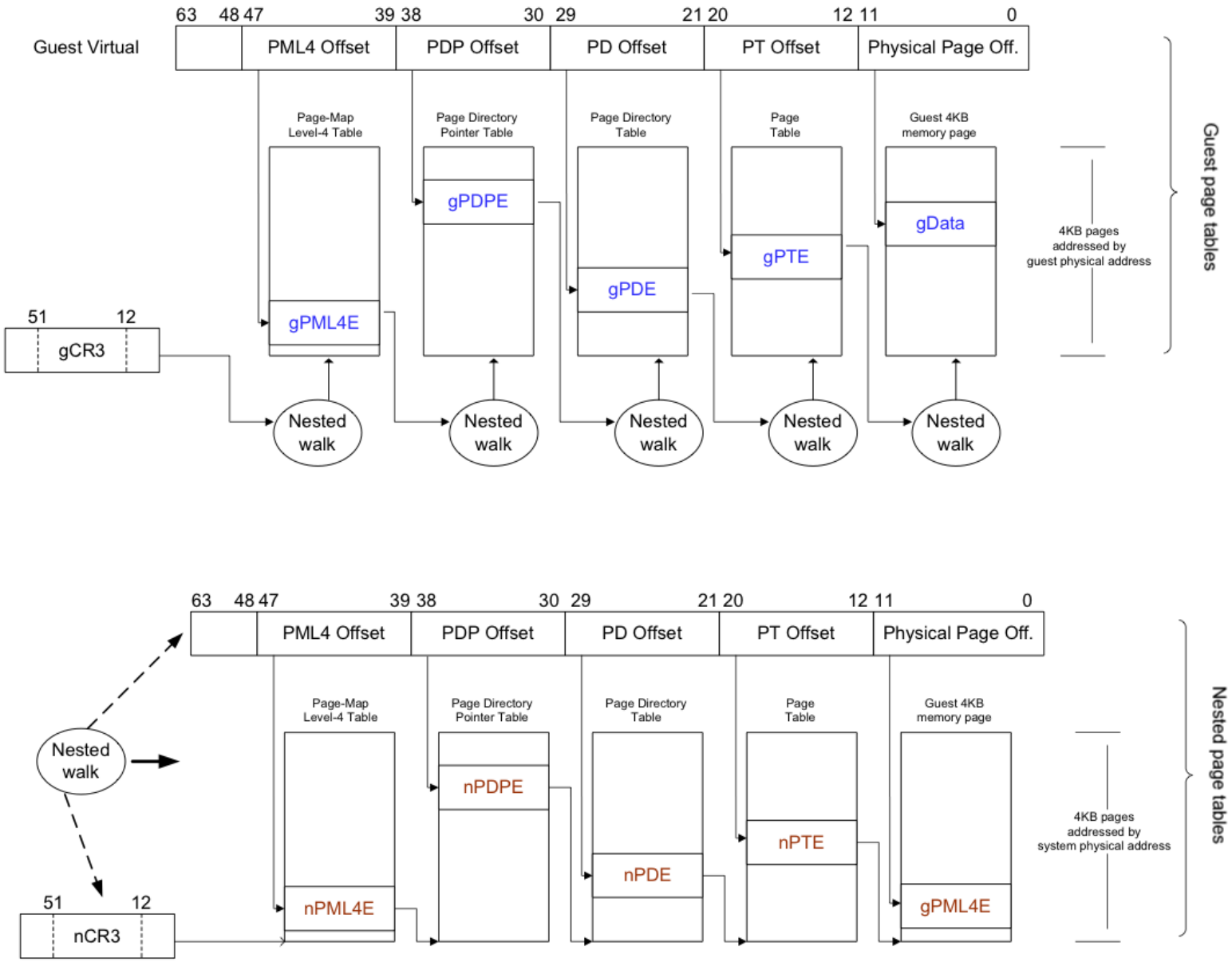
Nested paging (4)

- Hardware doet een page walk op gPT en nPT om het guest linear address te vertalen naar een host physical address.
- Guest kan gPT zelf inrichten, hypervisor hoeft hier niets over te weten.

Nested paging (5)

Vertaling gVA → hPA

1) Root table pointer: gPA
2) Nested walk zodat: gPA → hPA
3) Bepalen level 3 pointer (gPA)
4) Nested walk zodat: gPA → hPA
5) enz …
6) Uiteindelijk ons fysieke adres gPA
7) En weer een nested walk naar een hPA.



Bron: AMD-V Nested Paging White Paper, AMD Inc., 2008.

Paravirtualization

- Kunnen we het guest OS niet gewoon aanpassen om een soort system call te doen naar de hypervisor?
- In plaats van instructies uit te voeren die in een virtualized environment lastig zijn?
- Dit zijn instructies zoals I/O en privileged instructies.

Paravirtualization (2)

- Dit idee wordt gebruikt bij paravirtualization.
- Hypervisor definieërt een API, guest OS wordt aangepast om deze API te gebruiken.
- Je kan niet altijd een guest OS aanpassen (e.g. Windows).
- Maar zelfs in dat geval kan je drivers schrijven die hypervisor calls uitvoeren in plaats van I/O instructies.

Soorten Implementaties

- “Desktop” virtualization.
- Hardware/software VMM bovenop host operating systems.
- Voorbeelden:
  - VMWare Workstation / Player
  - Virtual Box
  - Parallels Desktop

-> Type-2 hypervisor.

Soorten Implementaties (2)

- VMWare ESX
- Hypervisor direct op de hardware.
  - “vmkernel” (micro kernel).
- Hier bovenop worden virtual machines aangemaakt.

-> Type-1 hypervisor.

Linux KVM

- Linux KVM: Kernel-based virtual machine.
- KVM is een kernel interface waarmee de adresruimte van een guest VM kan worden ingericht.
- Qemu gebruikt deze interface en kan guest VMs draaien op “native speed”.
  - Als dat niet lukt kan Qemu terugvallen op emulatie.

Linux KVM (2)

- Qemu emuleert standaard hardware voor video, audio, input, disk en netwerk.
  - Via KVM worden device traps ingesteld.
- Er is ook ondersteuning voor “paravirtualized” drivers.
  - Met name interessant voor netwerk & disk.

Linux KVM (3)

- Process Scheduling
  - Een guest heeft 1 of meer “cores” of “vCPUs”.
  - Het guest OS gebruikt zijn scheduler om voor elke core een proces te kiezen.
  - De “vCPUs” zijn op hun beurt weer zichtbaar als processen binnen de hypervisor / het host OS. Hier wordt de process scheduler van de host toegepast.

Linux KVM (4)

- Memory allocation
  - Het inrichten van de adresruimte van de guest gaat in samenspraak met de hypervisor (KVM API).
  - Als een guest meer geheugen nodig heeft, moet de host worden verzocht om een geheugenallocatie. De host blijft “de baas” over alle resources.

Linux KVM (5)

- Disk drives.
  - Een virtuele disk binnen de guest is vaak een disk image file, partitie of logical volume op de host.
  - Alle schrijfacties naar een virtuele drive worden omgezet naar schrijfacties op (bijv.) deze image file.
  - Als er naar een image of partitie op de host wordt geschreven, wordt de disk scheduler van de host gebruikt.
    - (Heeft een disk scheduler binnen een guest wel zin?)
  - Er hoeft **geen** lokale disk image te zijn, een guest kan ook werken met een remote disk, bijvoorbeeld via iSCSI.

Xen

- Hypervisor, draait op “bare metal”. (Type-1).
- Een virtual machine is een “domain”
- Eerste OS dat gestart wordt valt binnen “dom0”.
- dom0 kernel mag direct alle hardware van het systeem benaderen en heeft dus drivers nodig.
- dom0 kernels: Linux, NetBSD.

Xen (2)

- Guest systemen vallen onder “domU”.
- I/O, twee keuzes:
  - paravirtualisatie
  - geemuleerd.
- Binnen Xen is het gehele spectrum van “fully virtualized” tot “fully paravirtualized” mogelijk.

Microsoft HyperV

- Virtual machines worden geïsoleerd als “partities”. (Vergelijk “domain” van Xen).
- Hypervisor in ring -1.
- Er moet tenminste 1 parent partitie zijn (verg. dom0), welke Windows Server draait. Deze partitie heeft direct toegang tot de hardware devices.
- Child partities kunnen worden aangemaakt met hypervisor calls om guests in te draaien.

Cloud Computing

- Door de opkomst van Internet en virtualisatie is men heel anders gaan denken over de manier waarop infrastructuur wordt opgezet.
- Waar en op welke hardware een stuk software draait is niet meer interessant.
- Via een netwerkverbinding kan je er altijd bij.
- Een virtual machine kan draaien “ergens in the cloud”.

Cloud Computing (2)

- Waarom nog zelf fysieke servers kopen en beheren?
- Sommige bedrijven bieden een “VPS” service aan.
  - VPS: Virtual Private Server.
- Of automatisch schaalbare compute resources, “capacity on demand”
  - Amazon EC2
  - Google Compute Engine
- Of een “cloud drive”: Google Drive, DropBox.
- IaaS: Infrastructure as a Service

Cloud Computing (3)

- Een voorbeeld van software om dit soort services te implementeren is OpenStack.
- OpenStack kan een cluster van servers beheren en daarin automatisch virtual machines opstarten.
- Management van:
  - Fysieke en virtuele machines.
  - Storage
  - Netwerk
- Web interface

Cloud Computing (4)

- Oplossing nodig voor large-scale storage.
- Disk images op lokale machine werkt niet zodra je gaat nadenken over live migration, automatisch opschalen, etc.
- Storage loskoppelen van “compute” machines.
  - Disk images op een gedistribueerd file systeem.
  - Storage Area Network (SAN).
  - iSCSI.
- Lokale compute en storage maakt in theorie niet uit, als er maar een netwerkverbinding tussen ligt.

Cloud Computing (5)

- Er zijn zelfs “Cloud APIs” ontwikkeld.
- Virtual machines aanmaken met een paar simpele function calls.
- Apache “libcloud” voorbeeld:

```
from libcloud.compute.types import Provider
from libcloud.compute.providers import get_driver
cls = get_driver(Provider.RACKSPACE)
driver = cls('username', 'api key', region='iad')
sizes = driver.list_sizes()
images = driver.list_images()
size = [s for s in sizes if s.id == 'performance1-1'][0]
image = [i for i in images if 'Ubuntu 12.04' in i.name][0]
node = driver.create_node(name='libcloud', size=size,
image=image)
print(node)
```

Cloud OS kernels

- Men denkt na over toekomstige operating systems.
- Een virtual machine in een cloud voert vaak maar 1 taak uit. “Single-purpose appliance”.
  - Iets heel anders dan een **general-purpose** multi-user systeem.
- Authenticatie, verschillende users, file systems, etc. in OS niet meer nodig.
- In extreme gevallen: ook geen dual-mode operation meer nodig: single-address space kernel.
  - Als de kernel crasht, vangt de hypervisor dit op, het systeem blijft draaien.

Cloud OS kernels (2)

- Trend: ontwikkeling van “unikernels”, speciaal bedoeld voor het draaien van 1 enkel programma.
  - Single process.
  - Single address space.
  - Alleen maar drivers voor “paravirtualized devices” aangeboden door hypervisors.
- Voorbeeld: OSv.
  - Oorspronkelijk bedoeld om alleen een JVM te draaien, met daarboven op een Java applicatie.
  - Merk op: een JVM biedt ook al protectie tegen geheugenfouten en dergelijke.

Could OS kernels (3)

- Er wordt nu ook (weer) gewerkt aan “library operating systemen”.
- De OS kernel is nu een library en wordt samen met de applicatie gelinkt tot 1 binary executable.
  - Weer single-address space.
- Voorbeeld: MirageOS.
  - In dit geval wordt de applicatie geschreven in een “higher-level” language, waardoor geheugenfouten niet kunnen optreden (protectie).

Hoe zit dat met containers?

- Zeer populair: gebruik van containers met bijvoorbeeld “Docker”.
- Hier is geen sprake van virtualizatie van de gehele machine, maar alleen van de omgeving aangeboden door het OS.
- Hoe zit de omgeving van het OS eruit na het opstarten?
  - We hebben een root filesystem.
  - We kunnen processen maken, welke allemaal een PID krijgen en die met elkaar kunnen communiceren.
  - Er is een netwerk verbinding.
  - etc.

Hoe zit dat met containers? (2)

- Waarom wordt maar één zo'n omgeving aangeboden?
- Idee: 1 OS kernel kan meerdere van dit soort omgevingen aanbieden, allen van elkaar geïsoleerd.
- Binnen elke omgeving apart root filesystem, met aparte softwareinstallatie, enz.
  - In feite zitten de processen die binnen deze omgeving draaien “opgesloten” in een container. Andere containers zijn **niet** zichtbaar.

Hoe zit dat met containers? (3)

- Verschillen met “whole-system virtualization”:
  - Er wordt per container **geen** aparte kernel gedraaid. Je kunt binnen een container **geen** privileged instructies uitvoeren.
  - Zeer weinig overhead, geen hypervisor, geneste page tables enz.
  - Isolatie minder strikt: in principe mogelijk dat wanneer een container is gehackt, de “host” ook kan worden gehackt.

Literatuur / referenties

- “Formal requirements for virtualizable third generation architectures”. Popek and Goldberg, 1974.
- “The architectures of virtual machines”. Smith and Nair, 2005.
- “The Origin of the VM/370 Time-Sharing Ssystem”. R. J. Creasy, 1981.
- “A Comparison of software and hardware techniques for x86 virtualization”. Adams and Agesen, 2006.
- “45 years of mainframe virtualization: CP-67/CMS and VM/370 to z/VM”. IBM, 2012.
- AMD-V Nested Paging White Paper, AMD Inc., 2008.
- “OSv — Optimizing the Operating System for Virtual Machines”. Kivity, et al., 2014. (USENIX ATC 2014)
- “Unikernels: Rise of the Virtual Library Operating System”. Madhavapeddy and Scott, 2014. (ACM Queue, Vol. 11, Issue 11).