TSBE Bern
Höhere Fachschule

Virtuelle Systeme - Servervirtualisierung

FS-2018

Christoph Bühlmann

Agenda

1. Virtualisierung – eine Einleitung
2. Beweggründe
3. Markt / Produkte
4. Technologie und Hardware
5. Hands on

Was ist Virtualisierung

Virtualisierung

- Prozessor und Memory sind echt
  - Geteilt zwischen verschieden Guests
  - Somit enge Kopplung an die Prozessorarchitektur (x86 / x64)
- Sonstige HW wird emuliert

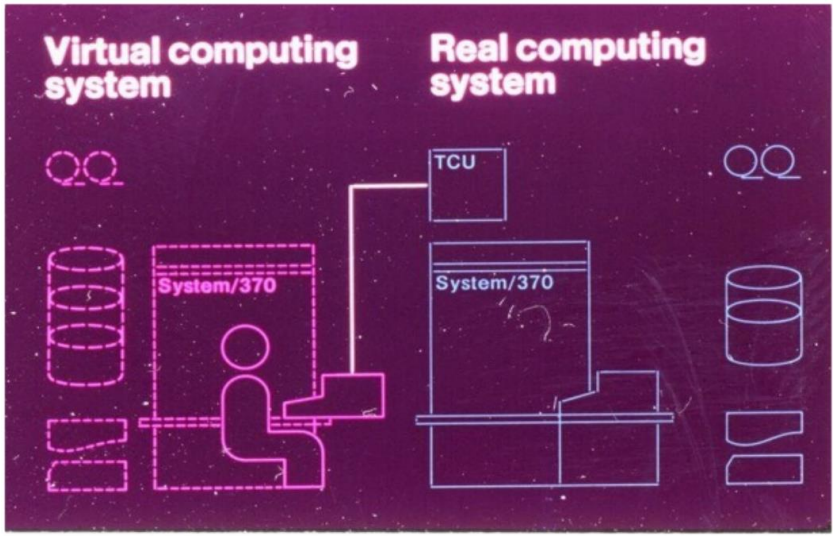
Emulation

- Alle HW wird emuliert
  - Somit können theoretisch alle Prozessor-Architekturen verwendet werden
  - Grosser Overhead

Entstehung

60er Jahre

- IBM-Mainframes
  - Aufkommen in den 60er Jahren
  - Hohe Anschaffungs- und Betriebskosten
  - Proprietäres OS, Applikationen geschrieben in COBOL
- Entwicklung von CTSS (Compatible Time Sharing System)
  - Teilen der teuren HW
  - Parallele Ausführung mehrerer Programme



Entstehung

Popek & Goldberg, 1974 (*Formal Requirements for Virtualizable Third Generation Architectures)*

- **Treue (Fidelity):** Die neue (virtuelle) Umgebung ist im wesentlichen identisch mit der ursprünglichen Hardware der physikalischen Maschine
- **Isolation oder Sicherheit:** (Isolation or Safety): Der Hypervisor/VMM muss über die komplette Kontrolle aller Systemressourcen verfügen
- **Performance:** Zwischen der Performance einer VM und ihrem physischen Gegenspieler sollte es höchstens marginale Performance-unterscheide geben

Entstehung

80er & 90er Jahre

- Aufkommen der x86 Architektur
- HW wird erschwinglich
- Aus CTSS wurde Unix

Millennium

- Geburt der Virtualisierung von x86 Systemen
- VMware GSX

Beweggründe Technisch

Auslastung / Kompatibilität

- Durchschnittliche Auslastung Bare Metal ~10%
- Teilen von HW -> Skalierbarkeit
- Unterschiede in der HW werden «versteckt»

Verfügbarkeit

- Einfache Migration von ganzen Systemen
- Sicherung durch Snapshot-Mechanismen
- Verteile Systeme und Loadbalancing
- Automatische Failovers

Markt / Varianten

Rechenzentrum:

Markt / Varianten

Übersicht (nicht abschliessend)

Produkt	Optimiert für	Technologie	Lizenz
QEMU	Integriert in Xen & Virtualbox, eingesetzt auch mit KVM	Hardware Emulation	GPL 2
Microsoft Hyper-V	Windows, SUSE Linux, CentOS	Paravirtualisierung, Hardware Virtualisierung	Microsoft EULA
XEN	Linux, (Windows)	Typ-2 Hypervisor mit Paravirtualisierung, Hardware Virtualisierung	GPL
KVM	Windows, Linux, andere	Typ-2 Hypervisor mit Paravirtualisierung	GPL
VMware ESXi	Windows, Linux, andere	Typ-2 Hypervisor mit Paravirtualisierung	Proprietär
VMware Player	Windows, Linux, andere	Typ-2 Hypervisor mit Paravirtualisierung	Proprietär
Oracle Virtual Box	Windows und Linux	Typ-2 Hypervisor mit Paravirtualisierung	Fast alles GPL 2

Markt / Varianten

Unser Test-System

- Proxmox VE 4.1 (Appliance) mit der folgenden Konfig
  - KVM als Typ-2 Hypervisor
  - Basis-System: Debian Jessie 8.2.0 mit Kernel 4.2.6
  - HW-Emulation mit QEMU 2.4
  - LXC Container Support (auch unprivilegiert)

Die vermittelten Konzepte sind auch auf andere Produkte anwendbar

Technologie / Hardware Emulation

Simulation beliebiger Hardware

- Auf beliebigen Plattformen
- Echte Geräte = Echte Treiber
- Keine Anpassungen am Guest-OS nötig
- Grosser Overhead -> eher schlechte Performance

Anwendungen

- Kompatibilität zu alten od. architekturfremden Plattformen
- Simulation zu Entwicklungszeitpunkt (Android, Symbian)
- Simulation von Peripherie (zB. Ethernet- und Grafikkarten)

Technologie / Hardware Emulation

Formal Requirements for Virtualizable Third Generation Architectures

- Treue (Fidelity): ✓
- Isolation oder Sicherheit: ✓
- Performance: ✗

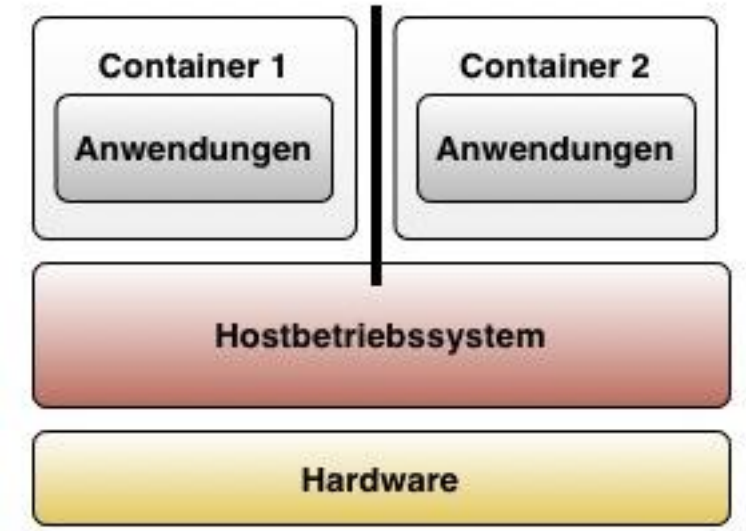
Technologie / OS Virtualisierung

Guest OS = Host OS

- Beide Systeme nutzen den gleichen OS-Kernel
  - Mit Windows kaum möglich
- Containers (bsp. LXC, Basis für Docker)
- Sehr wenig Overhead -> gute Performance

Umsetzung

- Limitation von CPU-Times oder Cores
- Limitation von Memory
- Simulation von Peripherie (zB. Ethernet- und Grafikkarten)



Technologie / OS Virtualisierung

Probleme

- Saubere Isolation wegen gemeinsamen Kernel nicht möglich
- Nur beschränkte Kontrolle über die zugestandenen Ressourcen
- Keine heterogene Systemlandschaft möglich

Formal Requirements for Virtualizable Third Generation Architectures

- Treue (Fidelity): ✗
- Isolation oder Sicherheit: ✗
- Performance: ✓

Technologie / Hardware Virtualisierung

Direkte Hardwareinteraktion

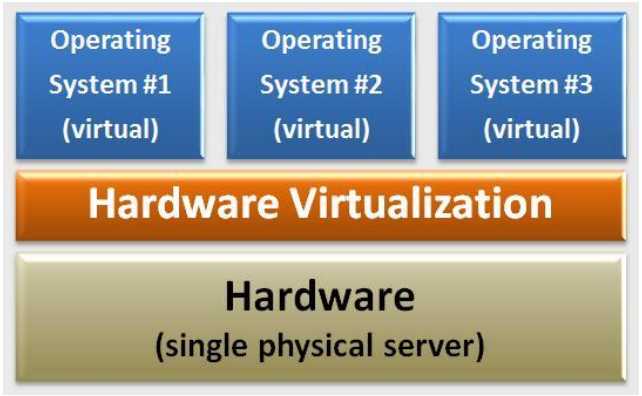
- Nicht privilegierte Befehle gehen direkt vom Guest OS an die CPU
- Nur die physikalisch vorhandene Architektur kann verwende
- Nicht in jedem Fall Anpassungen am Guest-OS nötig
- Wenig Overhead -> gute Performance

Isolation

- Ken Impact auf andere Gäste

Peripherie

- Alle Peripheriegeräte werden emuliert
- Echte Geräte = Echte Treiber



Technologie / Hardware Virtualisierung

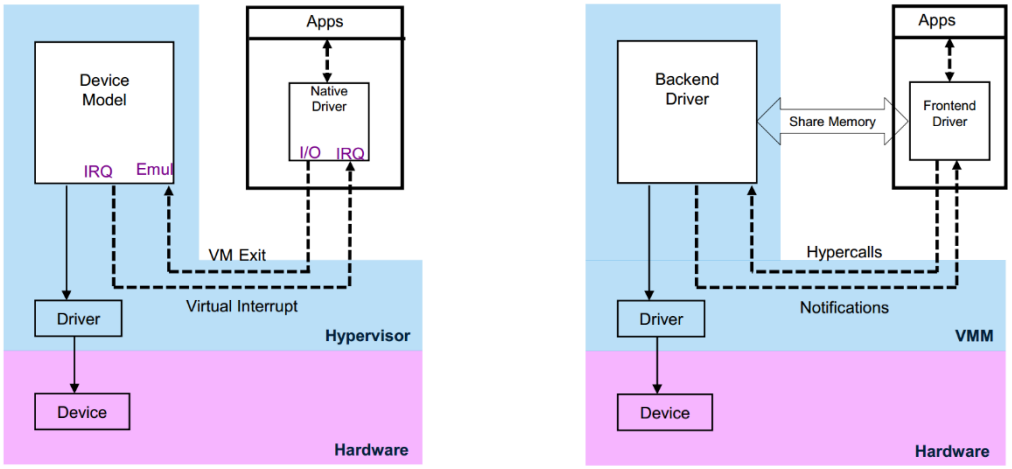
Formal Requirements for Virtualizable Third Generation Architectures

- Treue (Fidelity): ✓
- Isolation oder Sicherheit: ✓
- Performance: ~

Technologie / Paravirtualisierung

Breite Schnittstelle

- Guest kommuniziert über shared Memory mit dem Host
- Host bietet Hardware-Abstraktion
- Spezialisierte Treiber nötig!
- Wird die CPU auch so virtualisiert -> Anpassung am Guest - Kernel
- Wenig Overhead -> gute Performance



Technologie / Paravirtualisierung

Formal Requirements for Virtualizable Third Generation Architectures

- Treue (Fidelity): ✗
- Isolation oder Sicherheit: ✓
- Performance: ✓

Technologie / Hardware Virtualisierung

Alle Lösungen sind nicht perfekt -> Gehen wir tiefer in die Technik und virtualisieren CPU's mit den verschiedenen Ansätzen

Technologie / CPU Instruktionen und Memory

Unprivilegierte Instruktionen

- Sind zB. Normale Recheninstruktionen
- Diese Instruktionen können auch direkt aus Anwenderprogrammen abgesetzt werden.
- Sie sind unkritsch für den Betrieb des OS und anderer Applikationen

Privilegierte Instruktionen

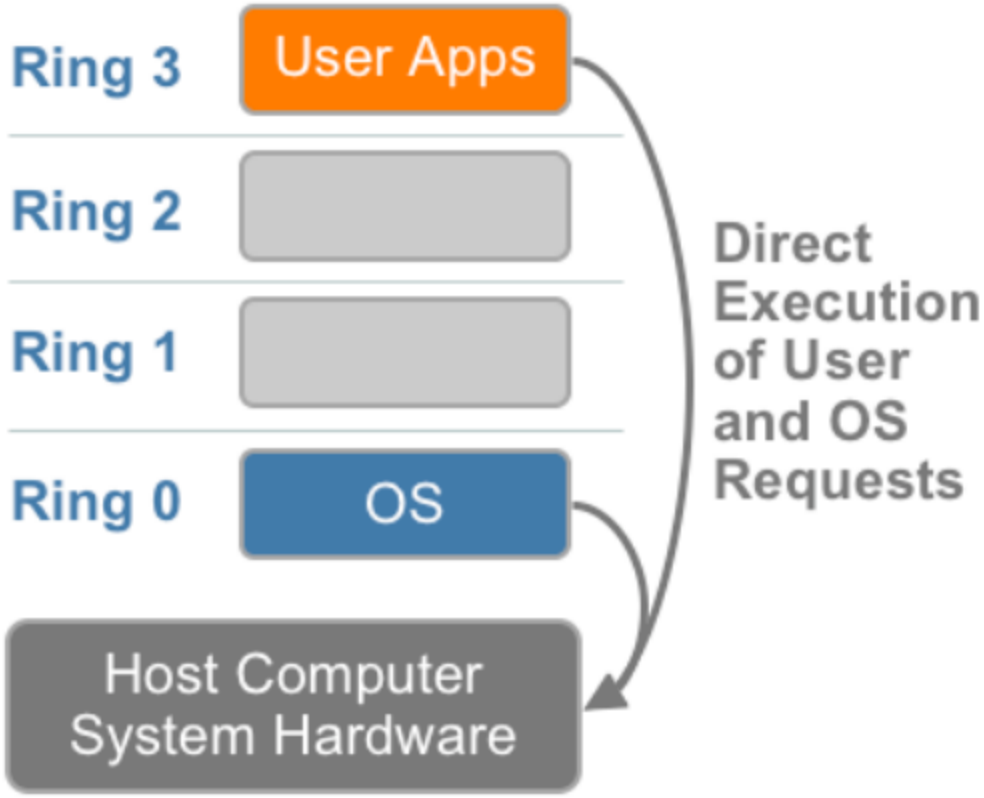
- Sind zB. Operationen auf dem Memory
- Diese Instruktionen müssen via OS und MMU (Memory Mapping Unit) erfolgen
- So wird sichergestellt dass keine Memory-Bereiche des OS oder anderer Applikationen beschrieben wird

Memory-Operationen werden somit immer durch die MMU von «virtuellen» Adressen auf reale übersetzt!

Technologie / CPU Basics

Ring Modell / Domain Modell

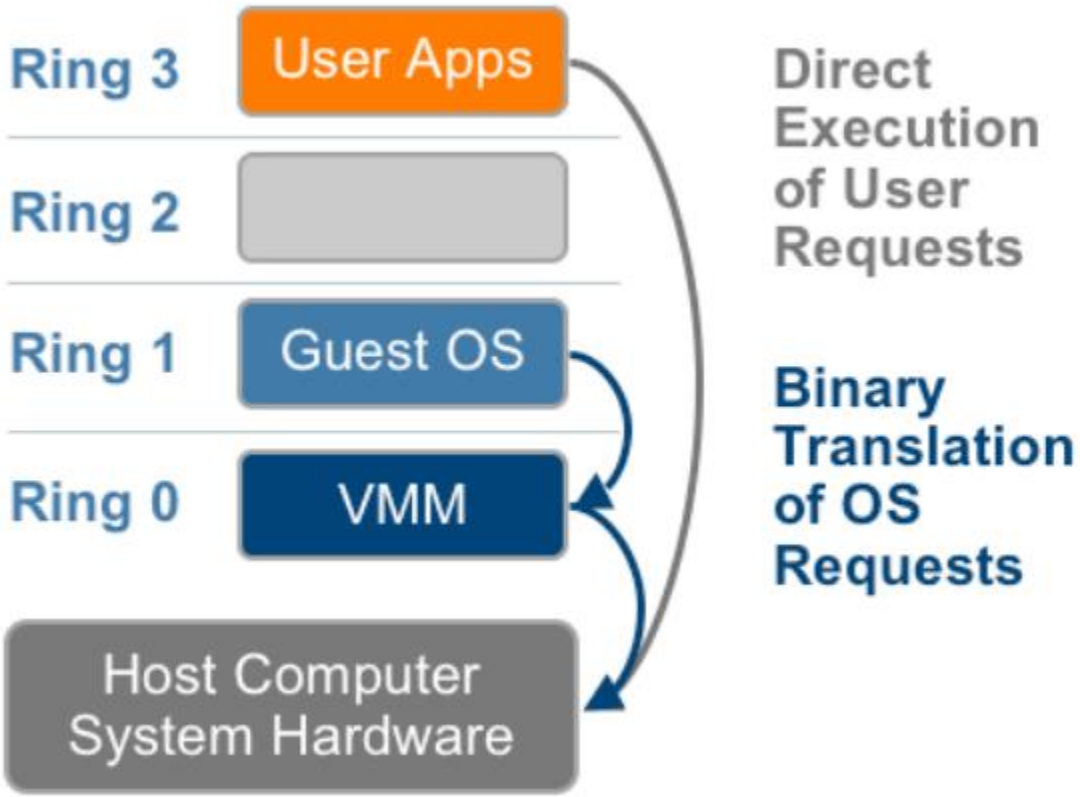
- Wie werden Instruktionen der CPU verarbeitet
  - Privilegierte vs. Unprivilegierte Instruktionen



Technologie / CPU Hardware emulation

Nur unprivilegierte Instruktionen gehen direkt an die CPU

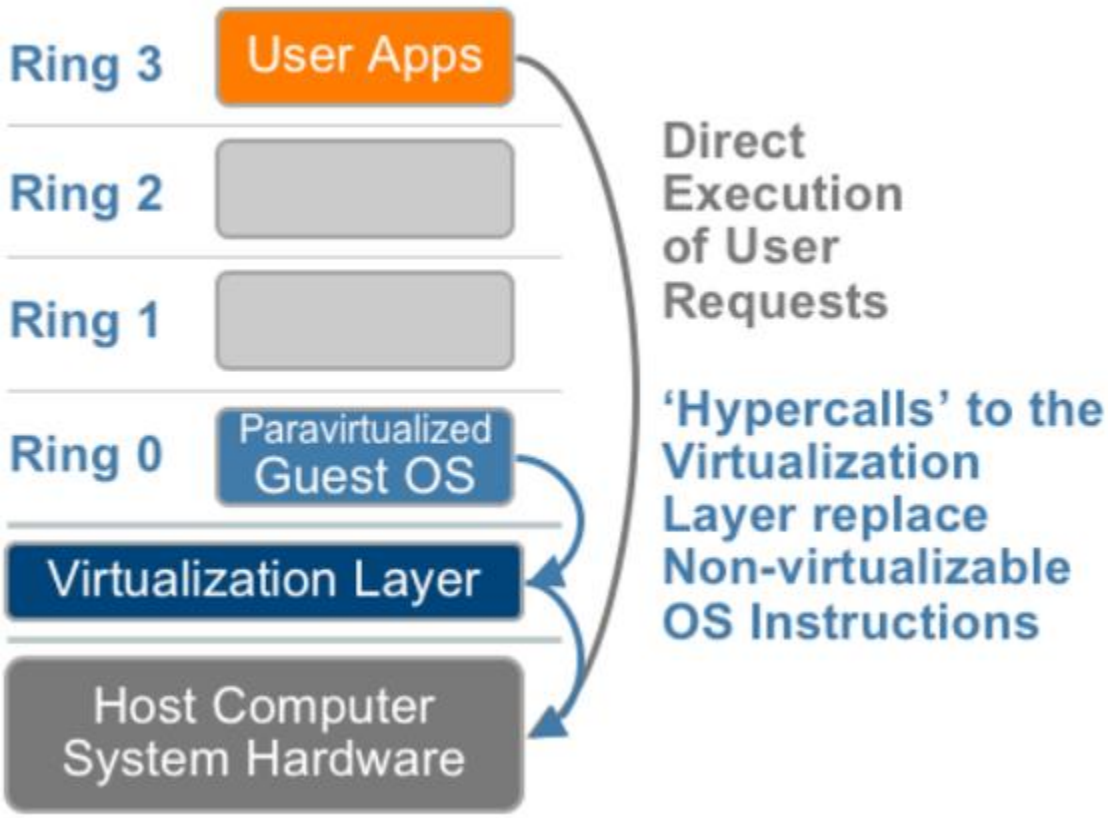
- Die privilegierten Instruktionen werden zu Laufzeit durch den Virtual Machine Monitor (VMM oder Hypervisor) abgefangen und übersetzt
- System «rutscht» einen Ring hoch



Technologie / CPU Paravirtualisierung

Nur unprivilegierte Instruktionen gehen direkt an die CPU

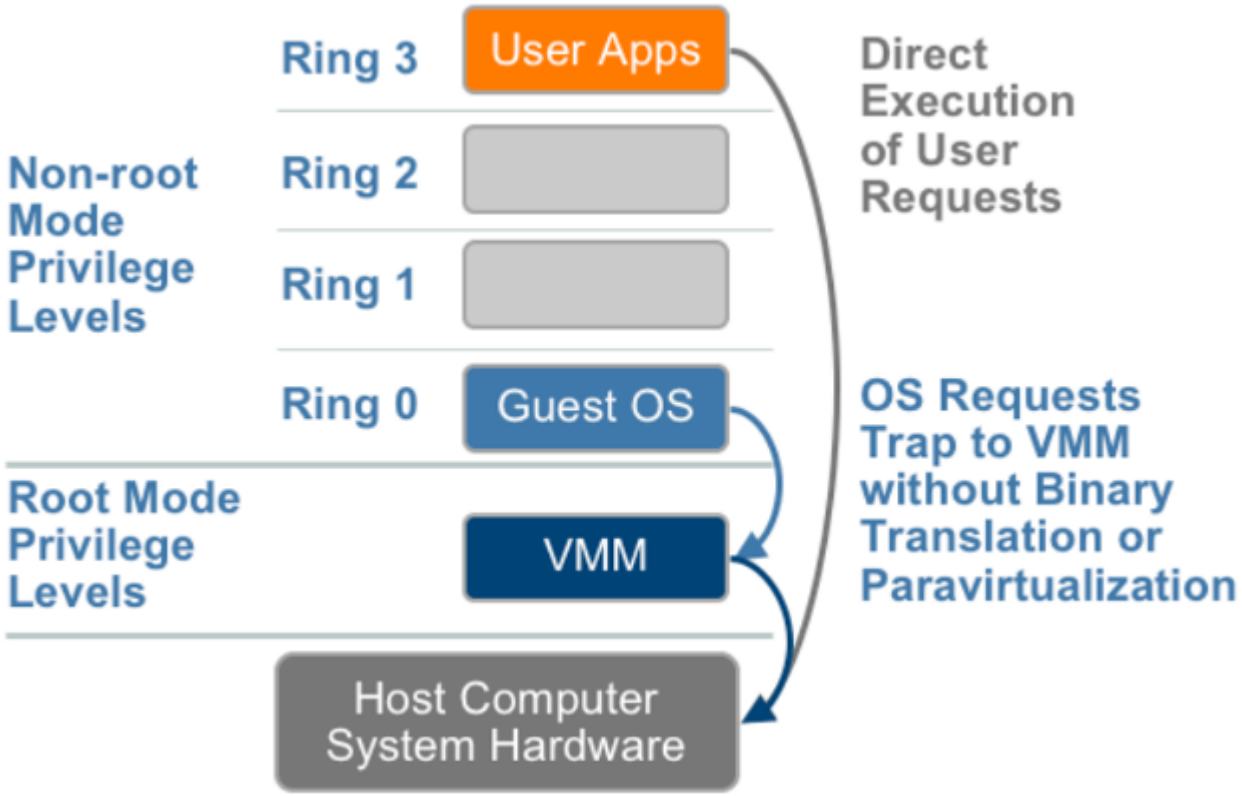
- Die privilegierten Instruktionen werden direkt im OS Source Code durch Virtual Machine Monitor (VMM oder Hypervisor) Calls ersetzt



Technologie / CPU Hardwareunterstütze Virtualisierung

Alle Instruktionen gehen direkt an die CPU aber…

- Bei privilegierten Instruktionen wird der VMM notifiziert (Trap) und kann eingreiffen
- Somit ist der VMM nun im Root-Mode oder Ring -1



Technologie / Hardware

CPU Unterstützung («Ring -1»)

- Intel VTx
- AMD-V
- Die Unterstützung muss in der Regel im BIOS aktiviert werden!

Memory-Unterstützung

- Intel EPT
- AMD-RVI

PCI-Unterstützung (für PCI-Passtrough)

- Ein PCI-Zugriff ist wie Memory-Zugriff -> Umsetzung analog MMU
- Intel VTd
- AMD-VI

GPU-Vitualisierung ist von einigen Grafik-Chips unterstützt, zB. Intel GVT-g

Technologie / Optimale Virtualisierung

Zusammenfliessen der Erkenntnisse

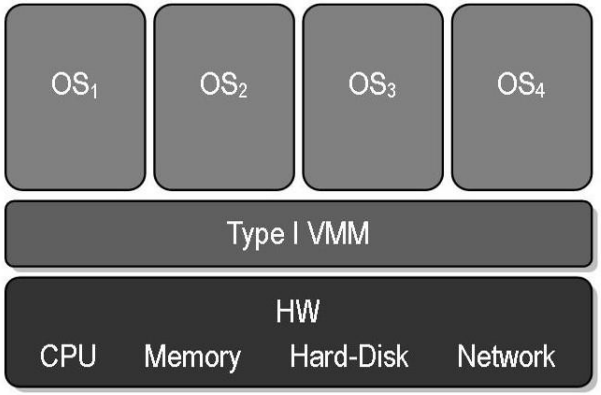
- CPU-Virtualisierung mit Hardwareunterstützung
- Memory Mapping in der MMU
- Anbinden von Peripherie (Ethernet etc..) mit Paravirtualisierung
- Spezielle HW mit PCI-Passthrough an die VM weitergeben

Technologie / Optimale Virtualisierung

CPU: Nur ein schlanker VMM ist benötight

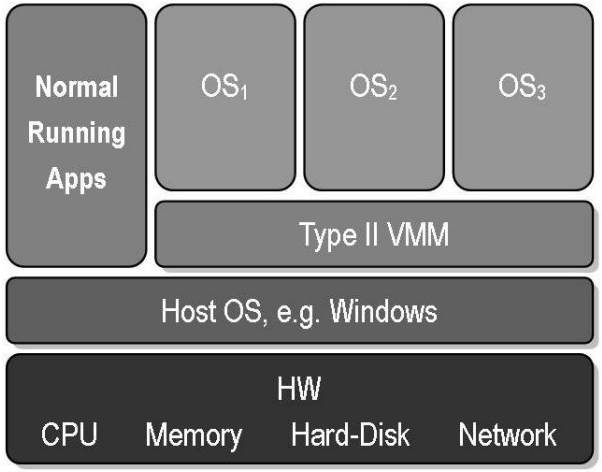
Typ-1 Hypervisor

- Auch Native / Bare-Metall
- Schlank, direkt auf der HW
- Treiber oder Build für spezifische HW nötig
- Vertreter: VMware ESXi



Typ-2 Hypervisor

- Vollwertiges OS, daher breiter
- Grosse Treiberunterstützung (vollwertiges OS halt)
- Vertreter: kvm



Technologie / Optimale Virtualisierung

Memory Mapping in der Virtualisierung

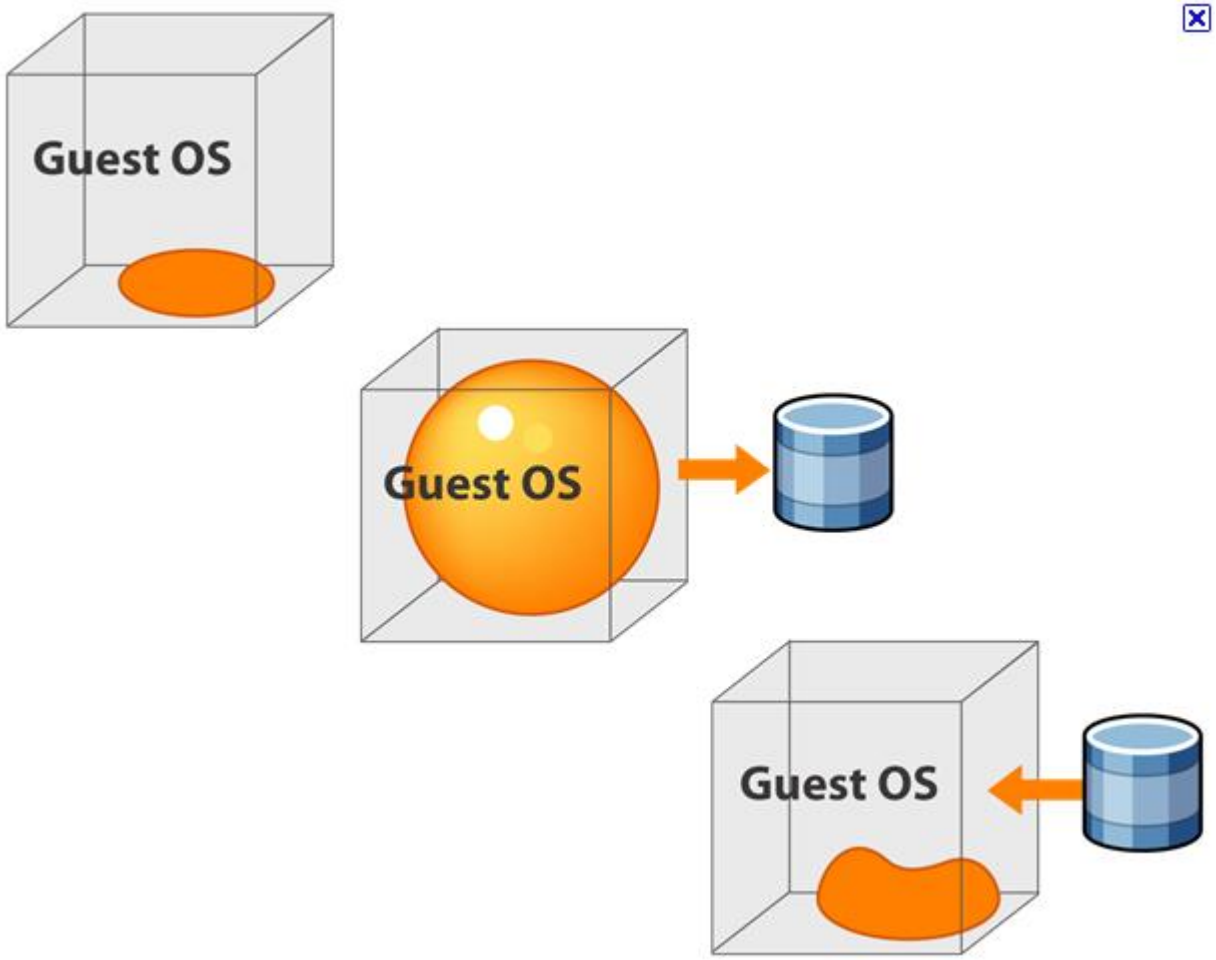
- Ein Doppeltes Mapping kann durch den Hypervisor erfolgen (Langsam)

- Oder aber mit einer erweiterten MMU direkt auf der Hardware

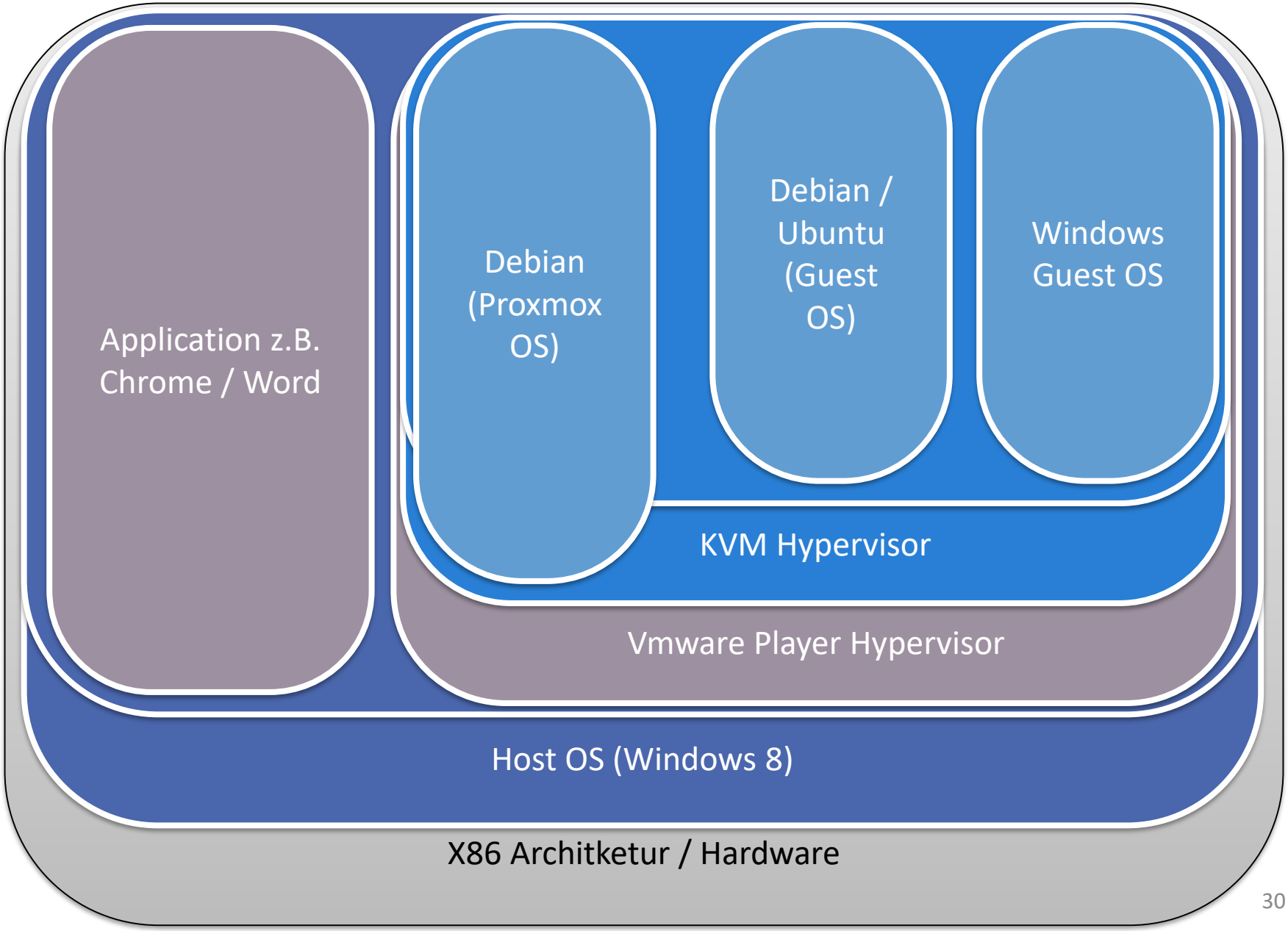
Technologie / Optimale Virtualisierung

Dynamische Memory-Zuweisung / Ballooning

- Wenn ihr Guest mehr Memory braucht wird er es kriegen
- Es werden spezielle Treiber (vor allem für die Freigabe) benötigt



Technologie / nested Virtualization



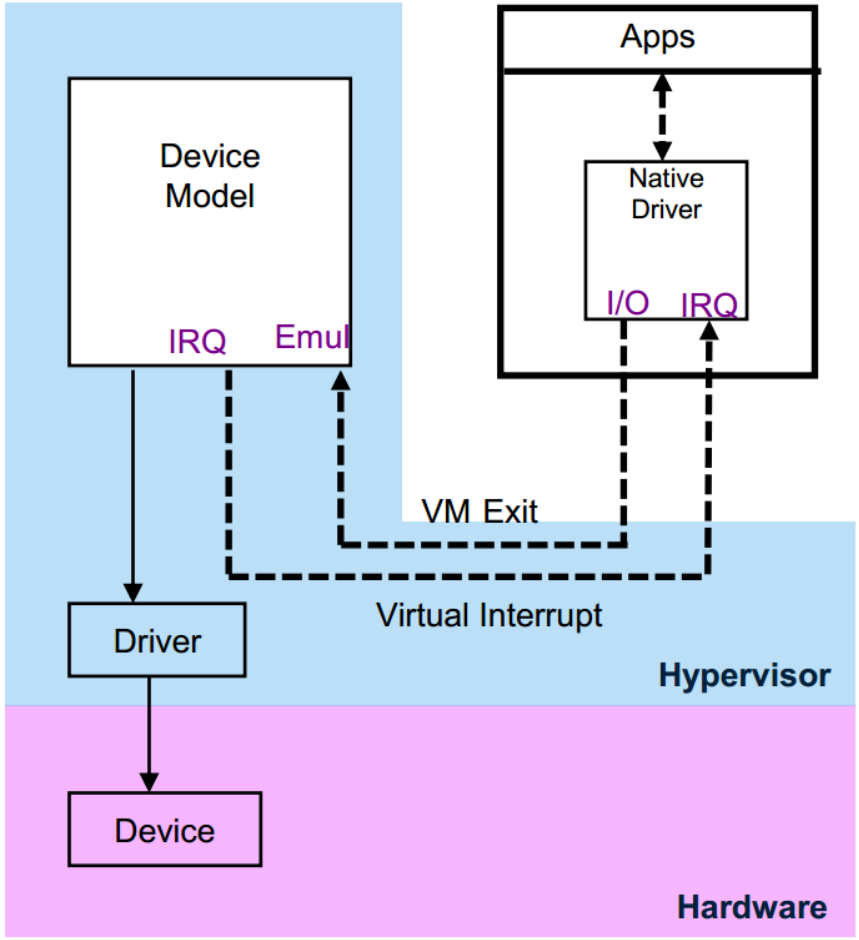
Technologie / Nested Virtualisierung

Peripherie 1 - Emulation

Windows Guest OS

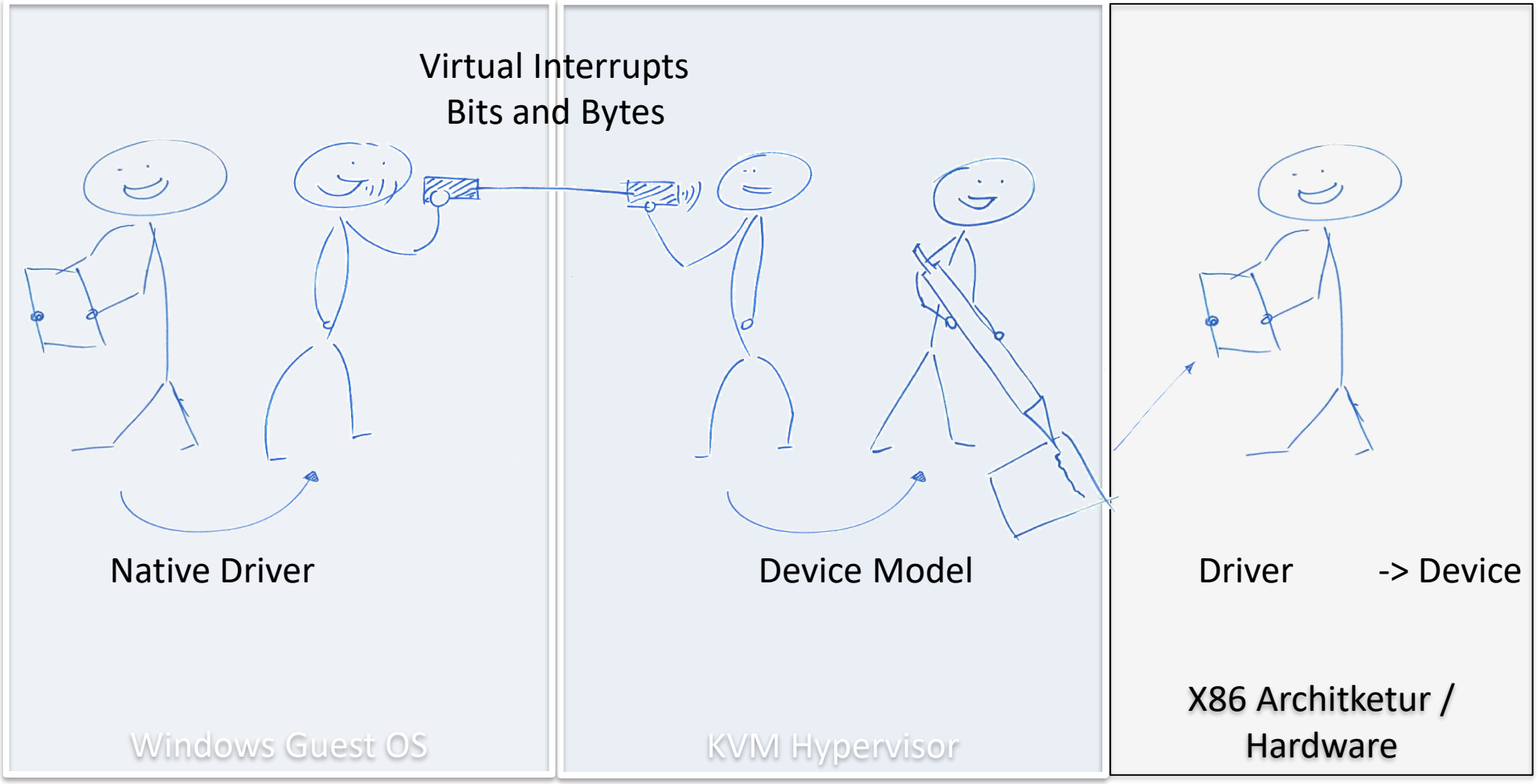
KVM Hypervisor

X86 Architketur / Hardware



Technologie / Optimale Virtualisierung

Peripherie 1 - Emulation

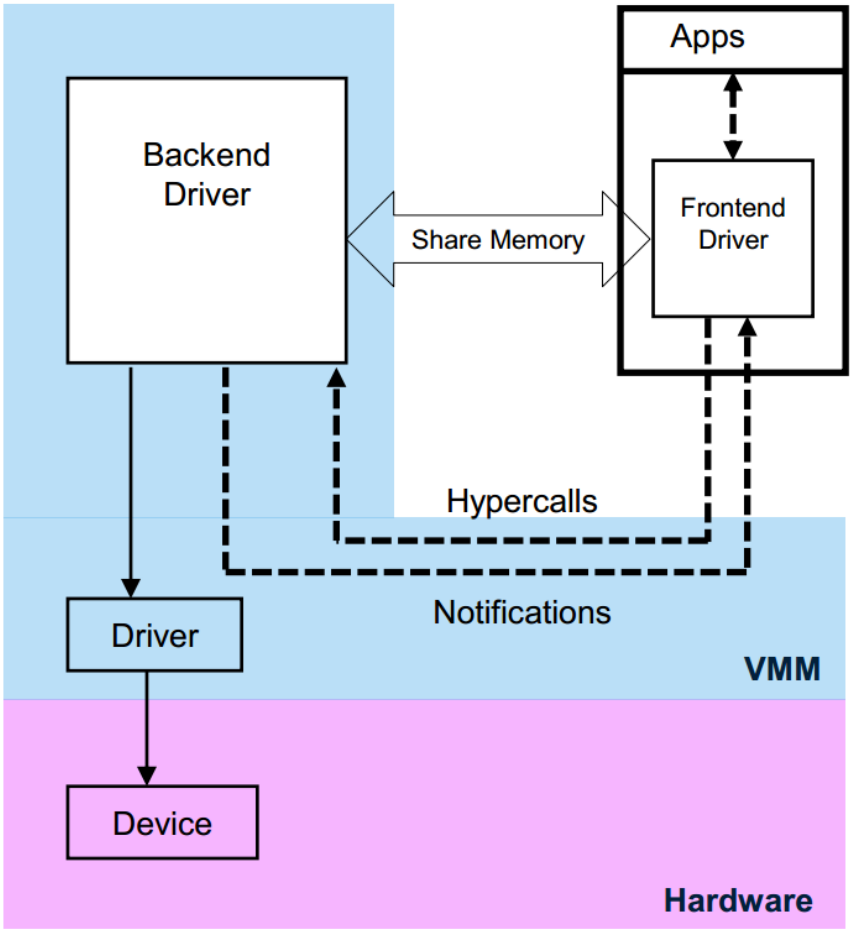


Peripherie 2 - Paravirtualisierung

Windows Guest OS

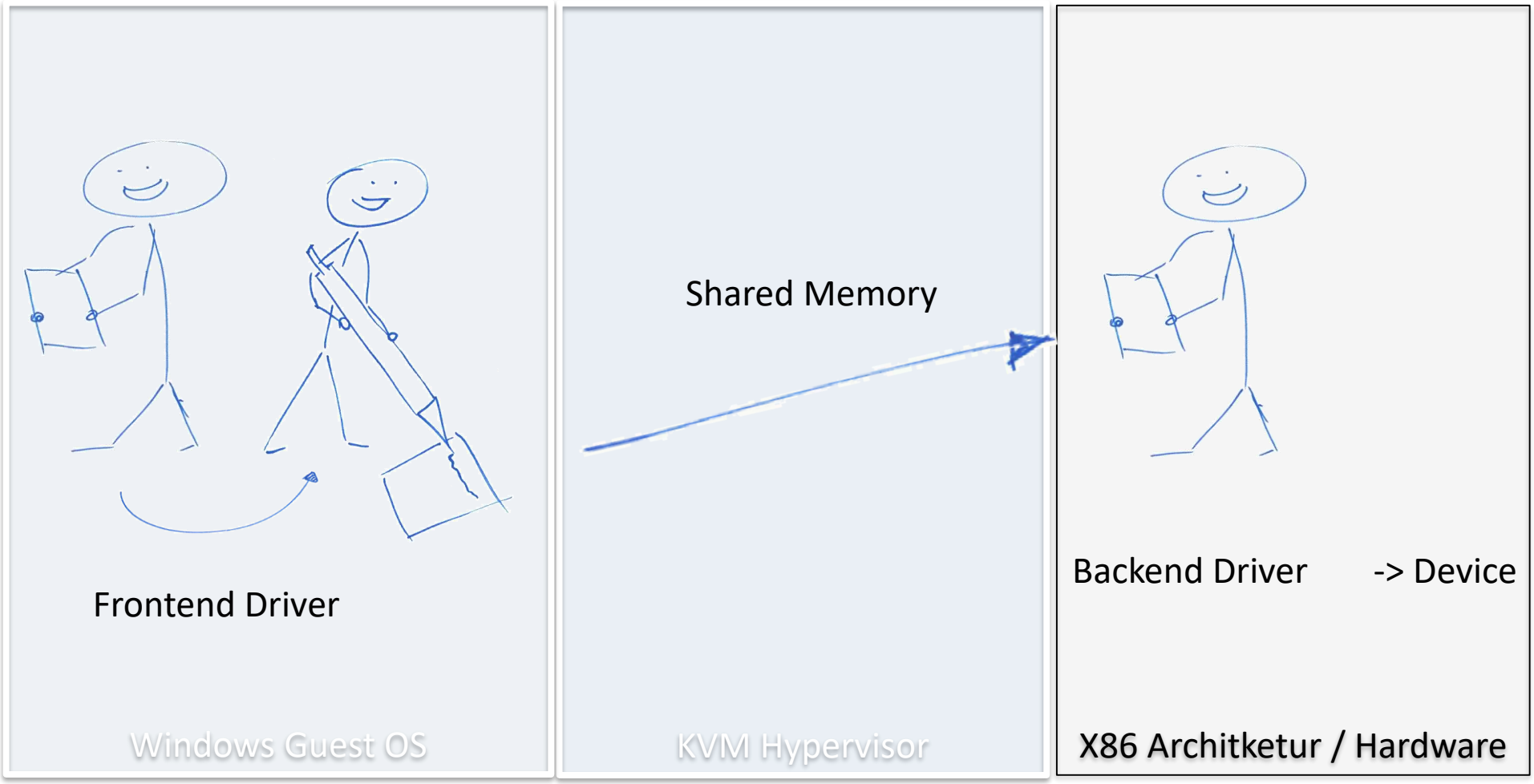
KVM Hypervisor

X86 Architketur / Hardware



Technologie / Optimale Virtualisierung

Peripherie 2 - Paravirtualisierung



Technologie / Optimale Virtualisierung

Formal Requirements for Virtualizable Third Generation Architectures

- Treue (Fidelity): ✓
- Isolation oder Sicherheit: ✓
- Performance: ✓

Fragen

Hands On – Cluster / Optimale VM's

Cluster

- Stellen Sie sicher, dass der DNS-Server auf 192.168.1.10 zeigt
- Ersetzen Sie die Fix eingestellte IP-Adresse im Management-Netz durch dhcp
- Bauen Sie auf einem Blade («Master») mit folgendem Befehl einen Cluster
  - `pvecm create <<ihr favorisierter clustername>>`
  - Weitere Infos zum Proxmox Virtual Environment Cluster Manager erhalten Sie mit `pvecm help` oder im Wik: https://pve.proxmox.com/wiki/Cluster_Manager
- Die anderen 2 Blades ihrer Gruppe fügen Sie auf der entsprechenden Konsole zum Cluster hinzu
  - `pvecm add <<fqdn des masters>>`
  - Überprüfen mit `pvecm nodes`

Hands On – Aufsetzen Laborumgebung

Storage

- `pvcreate /dev/sdb` kreiert ein Physical Volume PV
  - Achtung, muss nicht in jedem Fall sdb sein `lsscsi` hilft weiter
  - auf jedem Node ausführen
- `vgcreate -s 4M clusterVolume /dev/sdb` auf einem beliebigen Clustermember ausgeführt kreiert schlussendlich die Volumegroup

Hands On – Cluster / Optimale VM's

VM's

- Installieren Sie pro Cluster mindestens
  - 1 Windows-Guest
  - 1 Linux Guest mit GUI
  - Verwenden Sie bei beiden Installationen paravirtalisierte Ethernet und Storage-Treiber (Stichwort virtio)
  - Lassen Sie Memory dynamisch zuweisen. Wie viel Memory sieht ihr Guest-system? Wie wird dieser Effekt erzielt?
- Erstellen Sie mit Sysprep ein generelles Windows Template https://technet.microsoft.com/de-de/library/cc721940(v=ws.10).aspx