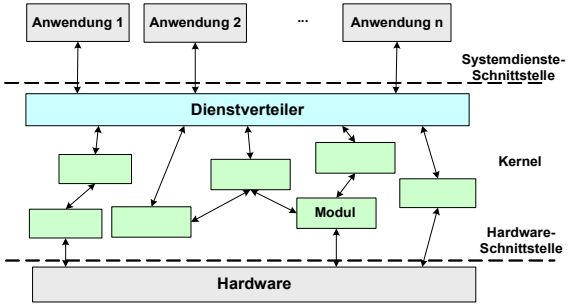
MAS: Betriebssysteme

Betriebssystemvirtualisierung

T. Pospíšek

Gesamtüberblick

1. Einführung in Computersysteme
2. Entwicklung von Betriebssystemen
3. Architekturansätze
4. Interruptverarbeitung in Betriebssystemen
5. Prozesse und Threads
6. CPU-Scheduling
7. Synchronisation und Kommunikation
8. Speicherverwaltung
9. Geräte- und Dateiverwaltung
10. **Betriebssystemvirtualisierung**



Zielsetzung

- Der Studierende soll die Grundlagen, Ziele und Konzepte der Betriebssystemvirtualisierung erläutern können
- Der Studierende soll wichtige Aspekte der Arbeitsweise von Virtualisierungstechniken verstehen

Überblick

1. **Grundbegriffe**
2. Virtualisierbarkeit von Hardware
3. Varianten der Virtualisierung
4. Betriebsmittelverwaltung bei Virtualisierung

Was ist Virtualisierung (Wiederholung)?

- Allgemeine Definition:
  - Unter Virtualisierung versteht man Methoden zur Abstraktion von Ressourcen mit Hilfe von Software
- Virtuelle Maschine verhält sich wie die reale Maschine
- Diverse Varianten:
  - **Virtuelle Computer: Server- und Desktopvirtualisierung (= Betriebssystem- bzw. Plattformvirtualisierung)**
  - Storage Virtualisierung
  - Anwendungsvirtualisierung
  - Virtuelle Prozessumgebungen (Prozessmodell und virtueller Speicher)
  - Virtuelle Prozessoren: Java Virtual Machine (JVM)
  - Netzwerkvirtualisierung (VLAN)

Terminologie zur Betriebssystemvirtualisierung

- Reale Maschine
- Virtuelle Maschine (VM)
- Hostbetriebssystem
  - Synonyme: Wirt, Host, Gastgeberbetriebssystem oder Hostsystem
- Gastbetriebssystem
  - Synonyme: Gast, Guest oder Gastsystem
- Virtual Machine Monitor (VMM)
  - Synonym: Hypervisor

Abgrenzung zur Emulation

- Unterscheidung Emulation – Virtualisierung
  - Emulation: **Komplette** Nachbildung der Hardware in Software
  - Virtualisierung: **Geringer Teil** der Befehle müssen nachgebildet werden, die meisten Befehle laufen direkt auf der Hardware (direkter Aufruf aus VM aus)

Partitionierung

- Bekannter Ansatz aus dem Mainframe-Umfeld:
  - Gesamtsystem wird bei der Partitionierung in Teilsysteme mit lauffähigen Betriebssysteminstanzen (VMs) partitioniert
- Ressourcen (Prozessor, Hauptspeicher, I/O-System) werden über die Hardware/Firmware den VMs zugeordnet
  - Erstes Betriebssystem dieser Art war CTSS (von MIT entwickelt)
  - Später wurde CTSS von IBM weiterentwickelt zu VM/CMS, heute z/VM (zSeries)

Nachteile der Betriebssystemvirtualisierung

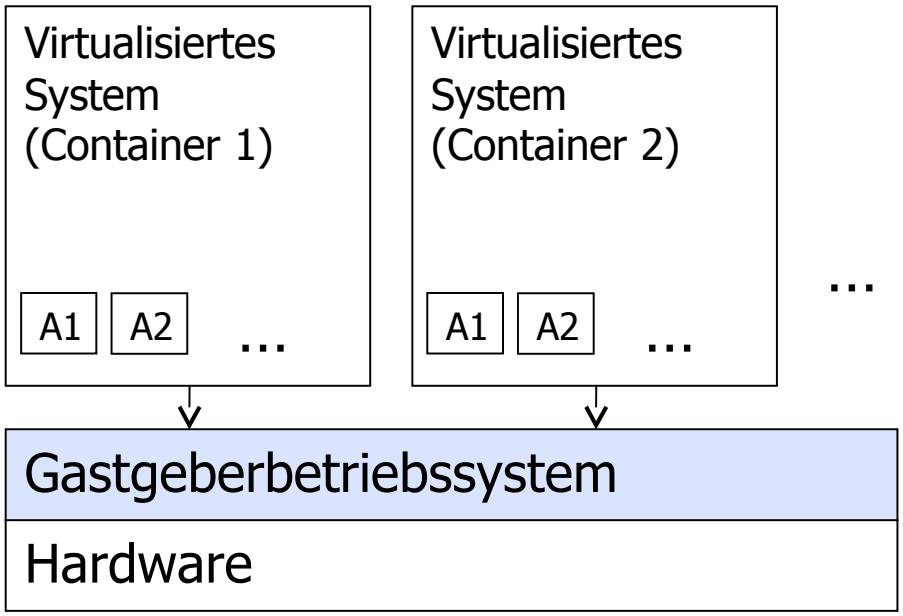
- Nachteile
  - Geringere Leistung als reale Hardware, Overhead von 5 bis 10 %
  - Schwierig bei spezieller Hardwareunterstützung
    - z.B. Hardware-Dongles, spezielle Grafikkarten
  - Bei Ausfall eines Serverrechners fallen gleich mehrere virtuelle Rechner aus
    - ☾ hohe Anforderungen an Ausfallkonzepte und Redundanz

Nutzen der Betriebssystemvirtualisierung

- Vorteile
  - Weniger Hardware notwendig, bessere Hardwareauslastung durch Serverkonsolidierung
    - Heutige Server sind meist bei weitem nicht ausgelastet
  - Weniger Leistungsaufnahme für Rechner und Klimatisierung (Stromverbrauch)
  - Flexibilität bei Aufbau einer Infrastruktur, schnelle Bereitstellung wird unterstützt, VMs beliebig vervielfältigbar und archivierbar
  - Vereinfachte Wartung, Life-Migration, unterbrechungsfrei, auch Technologiewechsel ohne Betriebsunterbrechung
  - Unterstützt Verfügbarkeits- und Ausfallsicherheitskonzepte
  - Unterstützung auch historischer Anwendungen

Abgrenzung: Anwendungsvirtualisierung (1)

- Kein zusätzliches Gastbetriebssystem, alles läuft im Host-Betriebssystem
- Isolierte Laufzeitumgebungen virtuell in geschlossenem Container
- Im Container laufen die Anwendungen



Ax: Anwendungen

Abgrenzung: Anwendungsvirtualisierung (2)

- Vorteile
  - Geringer Ressourcenbedarf und hohe Leistung
  - Gut für Internet Service Provider zur Skalierung von gehosteten Servern oder Webdiensten ☾ im Cloud Computing wird unterstützt
- Nachteile
  - Alle Umgebungen müssen den gleichen in einheitlicher Version BS-Kern nutzen
  - Verschiedene Betriebssysteme können nicht gleichzeitig verwendet werden
- Beispiele
  - Containertechnologie von Sun Solaris
  - OpenVZ für Linux
  - FreeBSD Jails
  - Linux-VServer

Virtualisierungsansätze

- Vollvirtualisierung:
  - Hardware wird vollständig virtualisiert
  - Keine Anpassung der Gastbetriebssysteme notwendig
- Paravirtualisierung:
  - Anpassung der Gastbetriebssysteme notwendig

Überblick

1. Grundbegriffe
2. **Virtualisierbarkeit von Hardware**
3. Varianten der Virtualisierung
4. Betriebsmittelverwaltung bei Virtualisierung

Das Problem mit den sensitiven Befehlen (1)

- Es gibt **privilegierte** und **nicht privilegierte** Befehle im Befehlssatz von Prozessoren
- Der Aufruf von privilegierten Befehlen löst einen Sprung ins Betriebssystem (Trap, Unterbrechung) aus
- Es gibt privilegierte Befehle, die dürfen nur im Kernelmodus ausgeführt werden
  - Trap ins Betriebssystem und Ausführung dort
- Es gibt in Prozessoren noch **die Kategorie der sog. sensitiven Befehle**,
  - zustandsverändernd (z.B. Zugriff auf I/O-Geräte oder die MMU oder auf Statusregister)
  - dürfen nur im Kernmodus ausgeführt werden

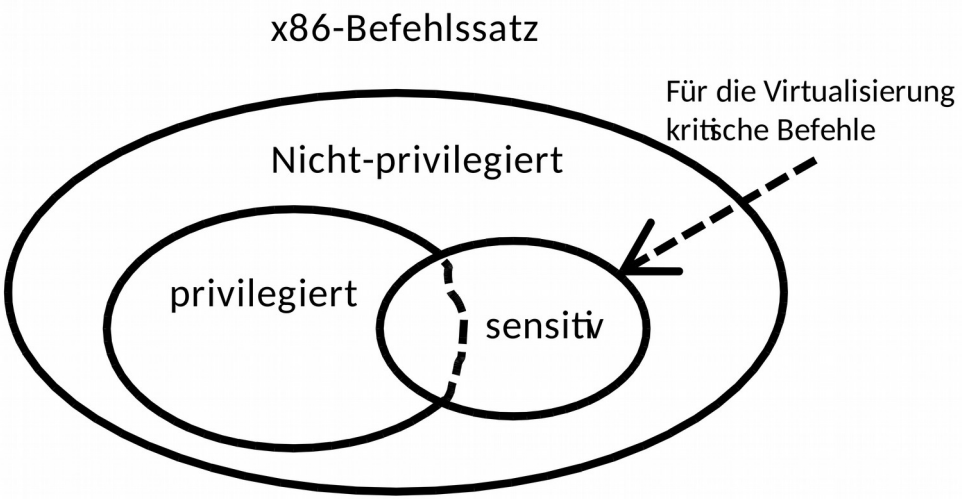
Das Problem mit den sensitiven Befehlen (2)

- Virtualisierbarkeit nach **dem Popek und Goldberg Theorem** (1974):
  - Nach Popek und Goldberg ist ein Prozessor virtualisierbar, wenn alle privilegierten Maschinenbefehle eine Unterbrechung (Trap) erzeugen, wenn sie in einem unprivilegierten Prozessormodus ausgeführt werden
  - Alle sensitiven Befehle sind auch privilegierte Befehle

Popek G. J.; Goldberg R. P. : Formal Requirements for Virtualizable Third Generation Architectures *Communications of the ACM* 17 (7). 1974. SS. 412 - 421

Virtualisierung bei x86-Prozessoren (1)

- **Kritische** Befehle sind privilegierte Befehle, die im Usermodus keinen Trap (Ausnahme) auslösen
- 17 Befehle von ca. 250 sind kritisch
  - GDT (Store Global Descriptior Table)
  - Stackzugriff mit PUSHF und POPF, usw.



Virtualisierung bei x86-Prozessoren (2)

- Lösung dafür nennt sich **Code Patching** oder ***Binärübersetzung*** (*Binary Translation)*
  - Kritische Befehle werden vor der Ausführung, also z.B. beim Starten eine Programms durch den Hypervisor erkannt und ausgetaucht
  - Typisch für Typ-2-Hypervisors (siehe unten)
  - Diese ausgetauschten Befehle führen einen Sprung in den Hypervisor aus, wo eine Emulation der kritischen Befehle erfolgen kann
- Damit kann trotz der Schwächen der x86-Architektur eine vollständige Virtualisierung erreicht werden

Prozessoren mit Virtualisierungsunterstützung (1)

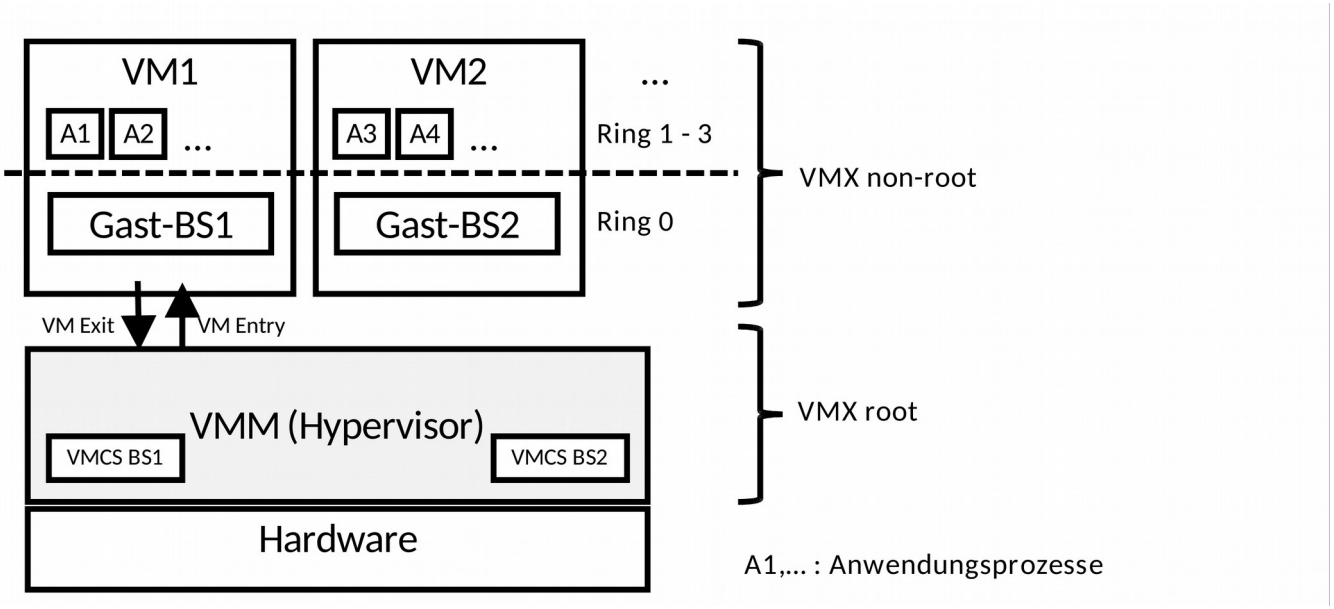
- Sensitive Befehle sind **vollständig** in den privilegierten Befehlen enthalten
- Intel und AMD bieten z.B. erweiterte Befehlssätze:
  - AMD: Pazifica oder ADM-V = SVM = Secure-Virtual-Machine-Befehlssatz
  - Intel: VMX = Virtualization Instruction Set Architecture (ISA) Extension, früher Vanderpool

Prozessoren mit Virtualisierungsunterstützung (2)

- Hypervisor läuft in einem neuen **Root- oder Wirt-Betriebsmodus** mit mehr Privilegien als der Ring 0
  - Wird auch als *Ring -1* bezeichnet
  - Root-Betriebsmodus hat die Kontrolle über die CPU und andere Ressourcen
  - Bei Intel-Prozessoren heißt der Root-Betriebsmodus auch *VMX root*.
- Die Gastbetriebssysteme nutzen dagegen den sog. **Gast-Modus**
  - *Wird bei Intel-Prozessoren auch als VMX non-root genannt*
  - Im Gast-Modus wird bei allen sensitiven Befehlsaufrufen ein Trap in den Hypervisor veranlasst

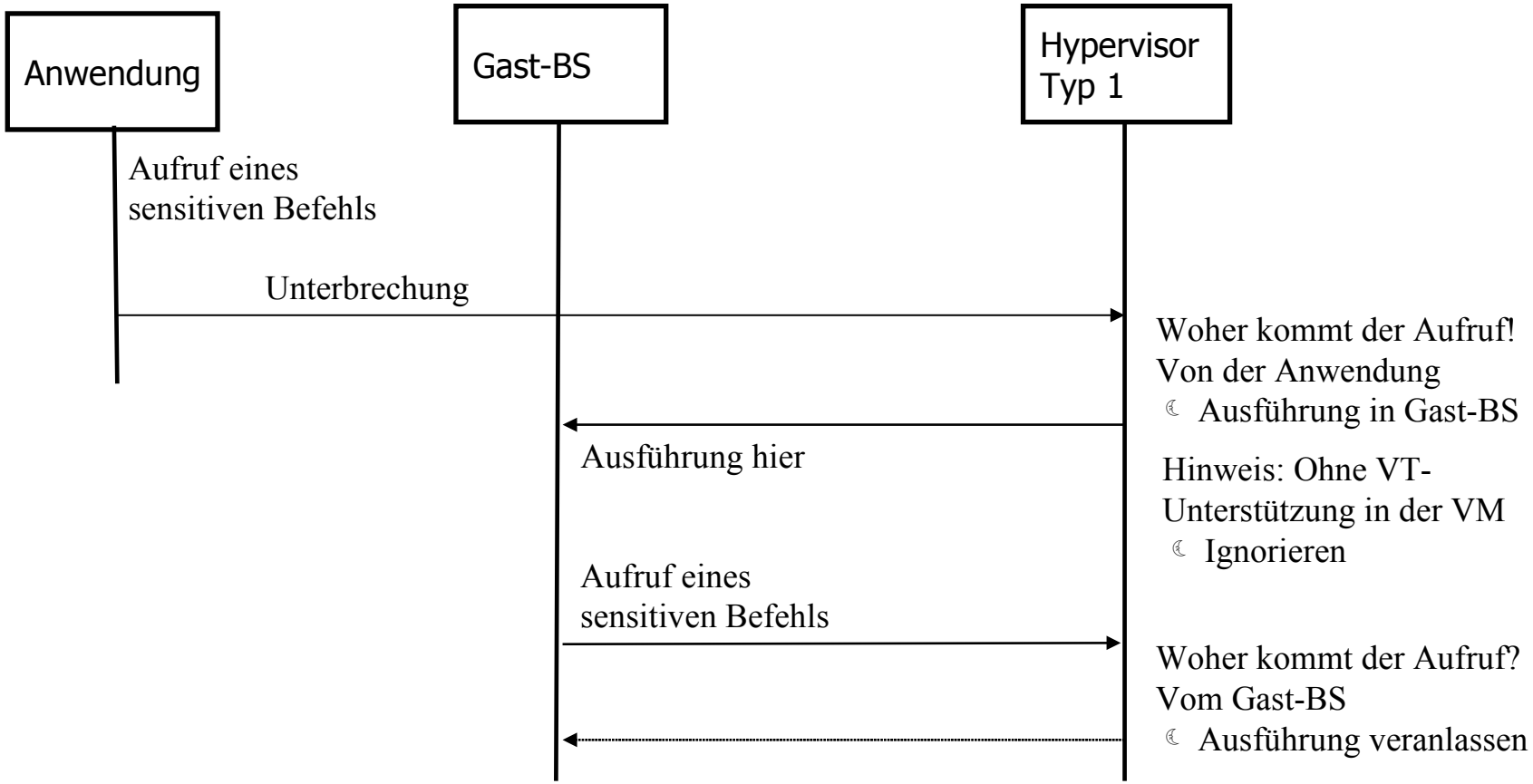
Prozessoren mit Virtualisierungsunterstützung (3)

- Hypervisor verwaltet je VM eine Datenstruktur – VMCS
- Mit **Hypercalls** springt VM in den Hypervisor
  VMCALL (bei Intel)
- Weitere Befehle: VMENTRY, VMEXIT, VMON, VMOFF, …



Das Problem mit den sensitiven Befehlen (5)

- Typischer Ablauf für die Befehlsausführung eines sensitiven Befehls mit Virtualisierungstechnik in der CPU



Überblick

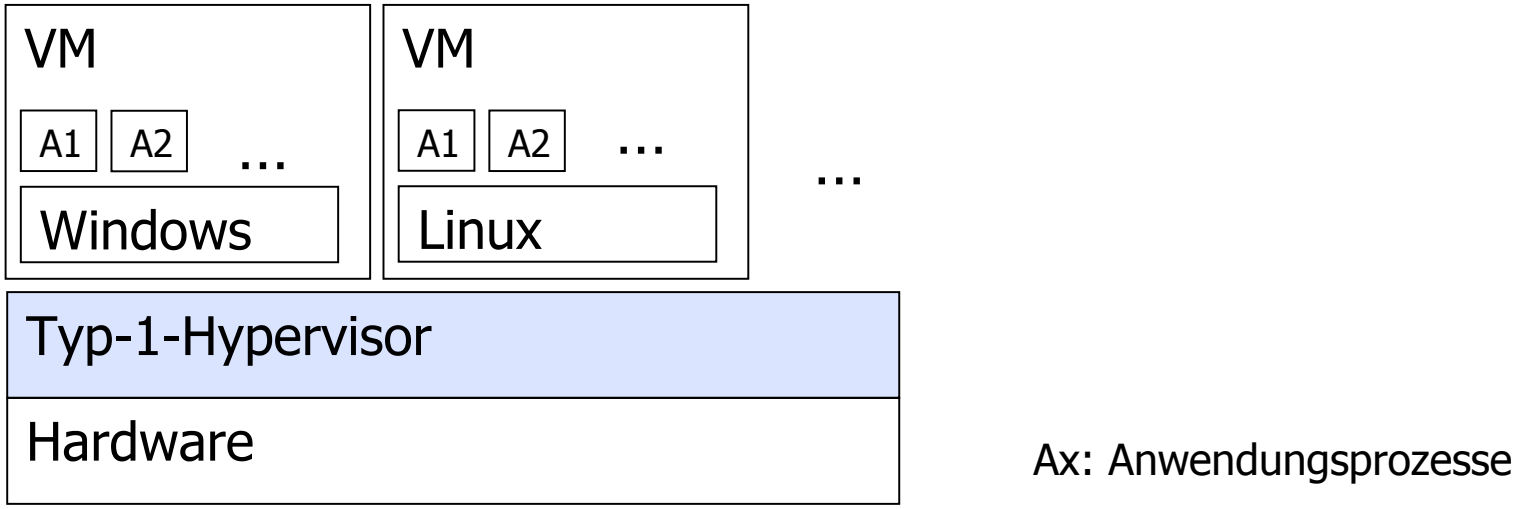
1. Grundbegriffe
2. Virtualisierbarkeit von Hardware
3. **Varianten der Virtualisierung**
4. Betriebsmittelverwaltung bei Virtualisierung

Überblick über Konzepte und Technologien

- Es gibt unterschiedlichste Virtualisierungskonzepte und -technologien
- Einige davon sollen diskutiert werden:
  - Vollständige Virtualisierung, Typ-1-Hypervisor
  - Vollständige Virtualisierung, Typ-2-Hypervisor
  - Paravirtualisierung, …

Typ-1-Hypervisor (Vollständige Virtualisierung) (1a)

- VMM = Typ-1-Hypervisor direkt über der Hardware als kleines **Minibetriebssystem**
- Variante benötigt VT im Prozessor, sonst nicht möglich
- Beispiele: XenServer von Citrix Systems, vSphere ESX von VMware, Hyper-V von Microsoft



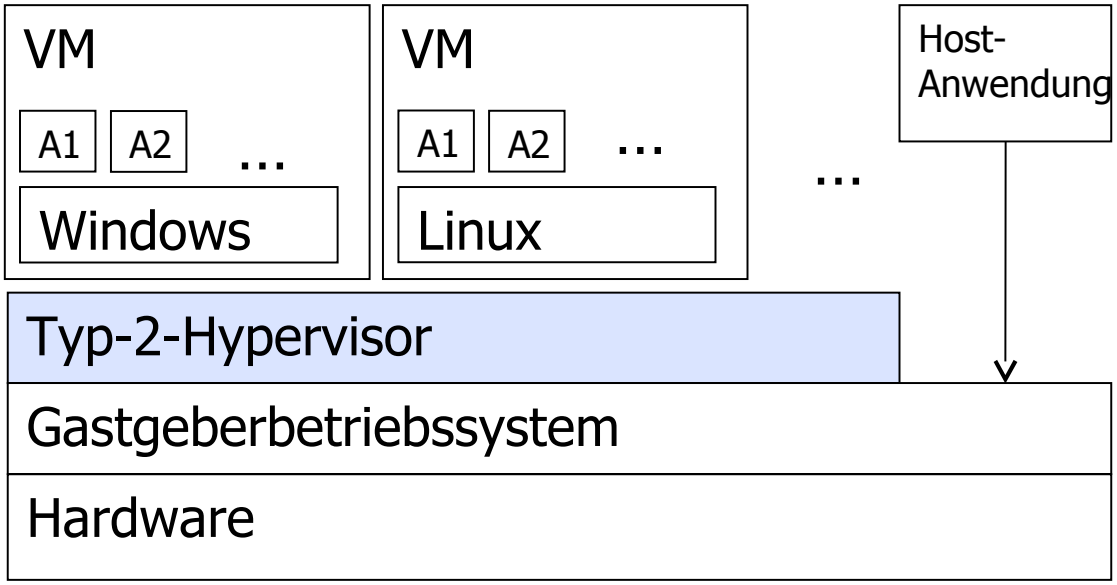
Ax: Anwendungsprozesse

Typ-2-Hypervisor (2)

- Auch Hosted-Ansatz, Vollvirtualisierung
- Wenn **keine** Virtualisierungsunterstützung durch Prozessoren vorhanden ist, ist Hypervisor vom Typ 2 sinnvoll
- Beim Start eines Anwendungsprogramms in einer VM wird zunächst eine Übersetzung sensitiver Befehle in spezielle Hypervisor-Prozeduren durchgeführt
  - *Binary Translation*
  - Hypervisor läuft als einfaches Benutzerprogramm über Gast-BS
- Gastbetriebssystem wandelt die Aufrufe in spezielle Aufrufe an den Hypervisor um, der die Befehle dann emuliert

Typ-2-Hypervisor (1)

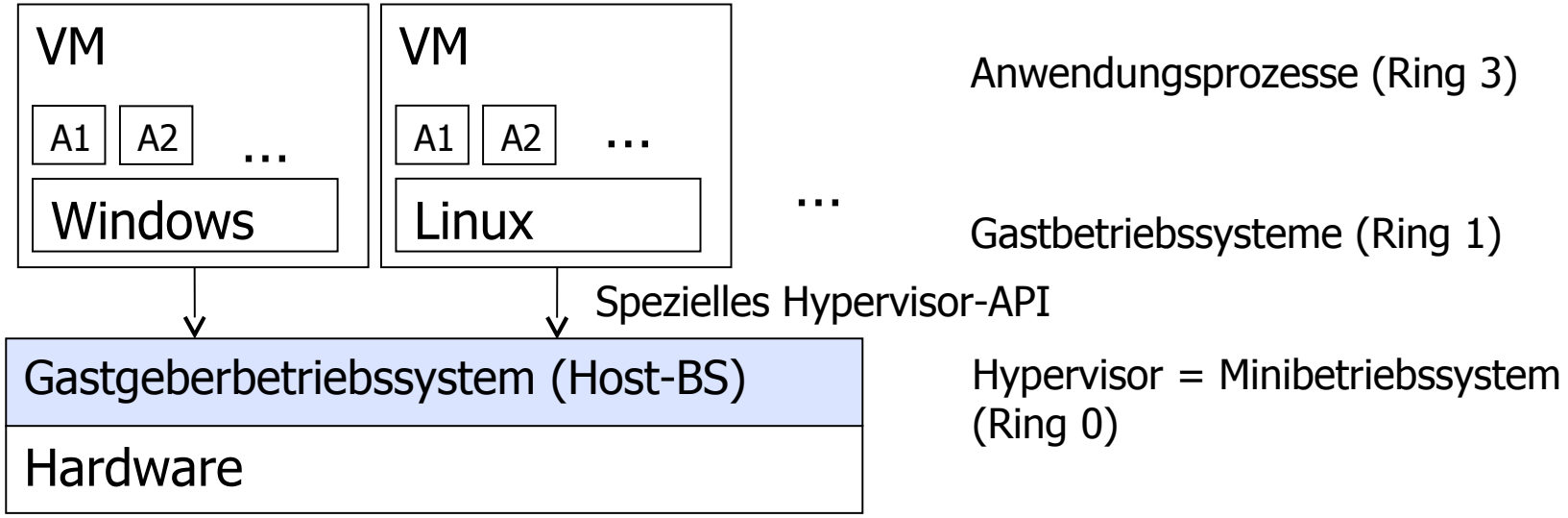
- Mehr Overhead als bei Typ-1-Hypervisor
- Nicht für Produktionseinsatz, eher für Test- und Entwicklungsumgebungen
- Beispiele: Virtual Server / PC von Microsoft, VMware Workstation



Ax: Anwendungsprozesse

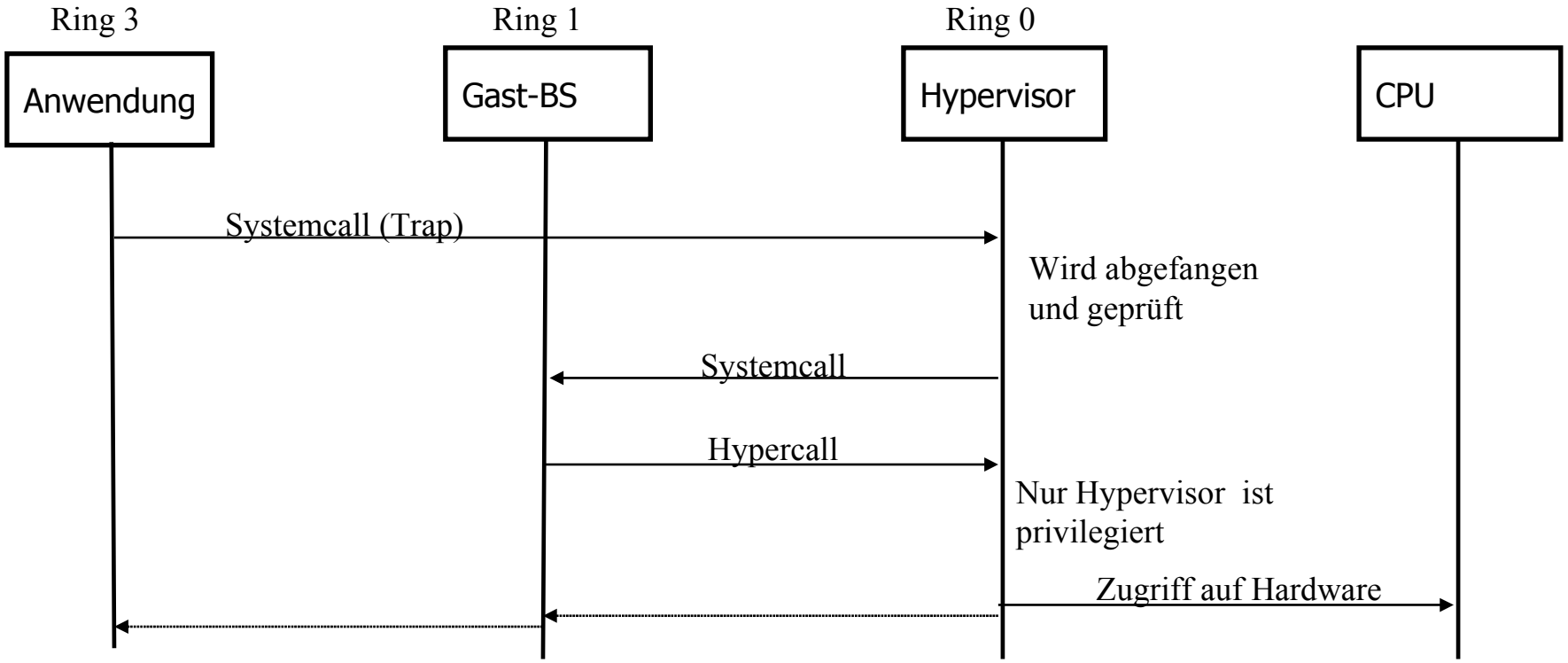
Paravirtualisierung (1)

- Hypervisor ist hier ein reduziertes Metabetriebssystem
- Paravirtualisierung arbeitet mit **modifiziertem** Gastbetriebssystem (paravirtualisieres Betriebssystem)
- Hypervisor-Aufrufe über spezielle Systemaufrufe (API)
- Beispiele: Das offene Xen



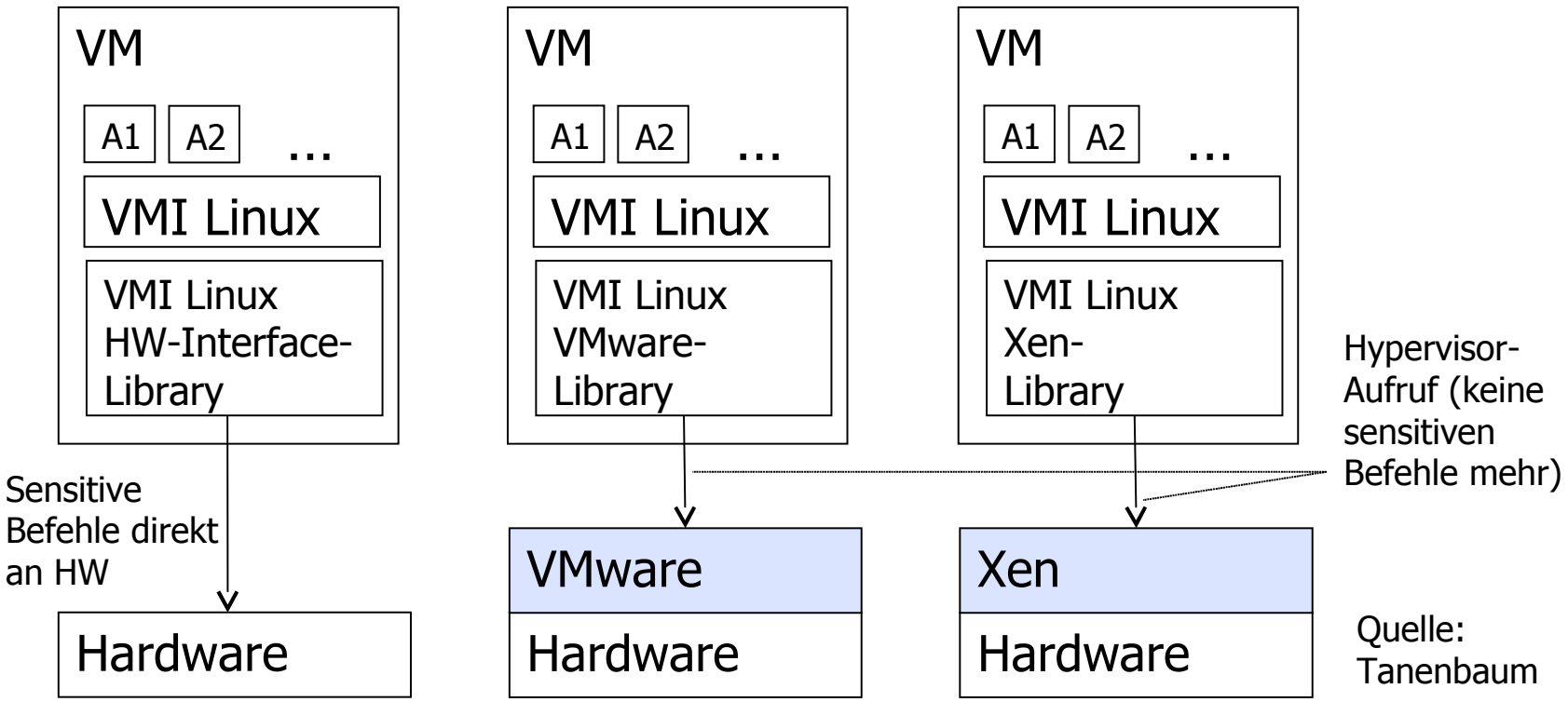
Paravirtualisierung (2)
Ablauf eines Hypercalls

- Bei x86-Architektur



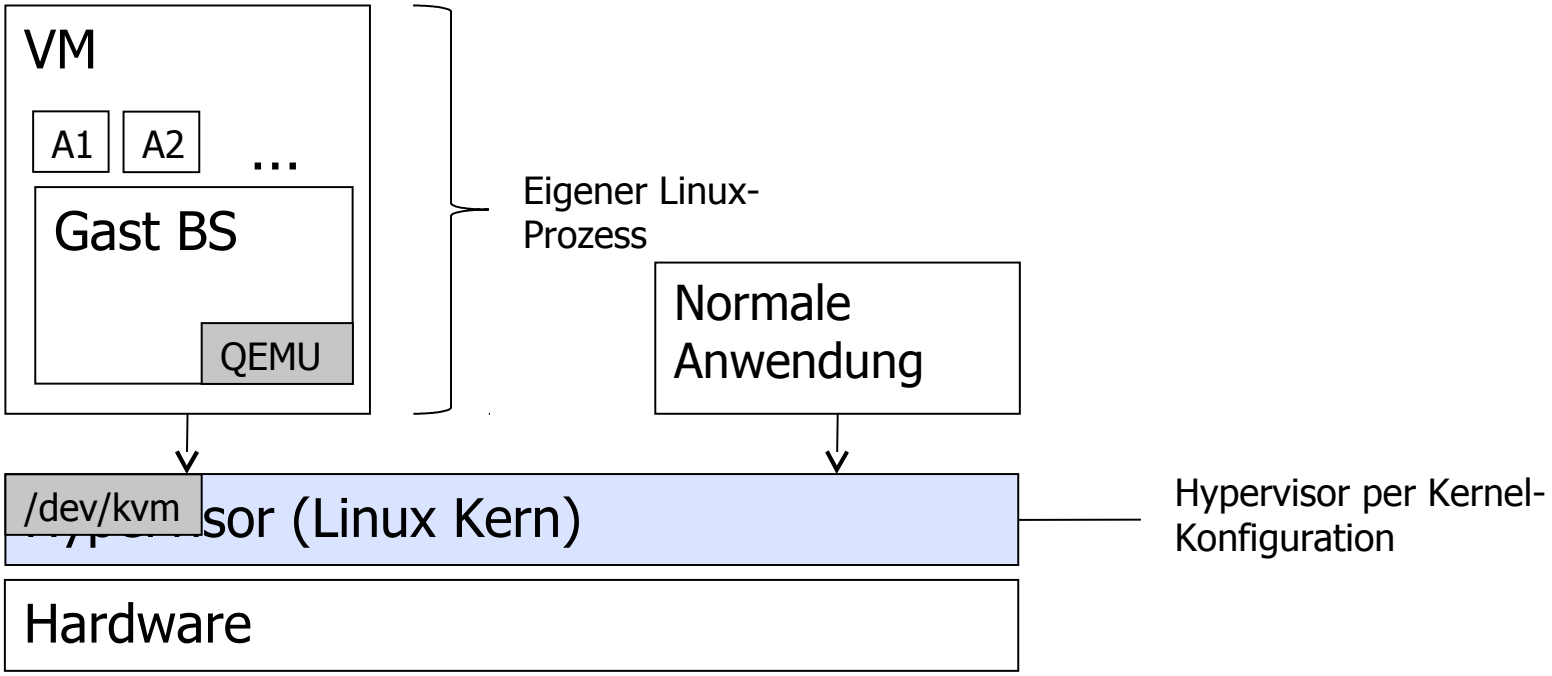
Weiterführung der Idee

- Idee eines VMI = Virtual Machine Interface als Schnittstelle zu einem Mikrokern (API mit Hypervisor-Befehlen)
- Hypervisor-Konzepte und Paravirtialisierung immer schwerer abzugrenzen



Fallbeispiel:
KVM – Kernel Virtual Machine unter Linux

- KVM ist ein Modul des Linux-Kerns ☾ **Linux-Kern ist Hypervisor**
- Voraussetzung: Prozessor unterstützt Virtualisierungstechnik
- Special Device **/dev/kvm** dient der Virtualisierung des Speichers
- **QEMU**-Prozess wickelt I/O-Befehle ab

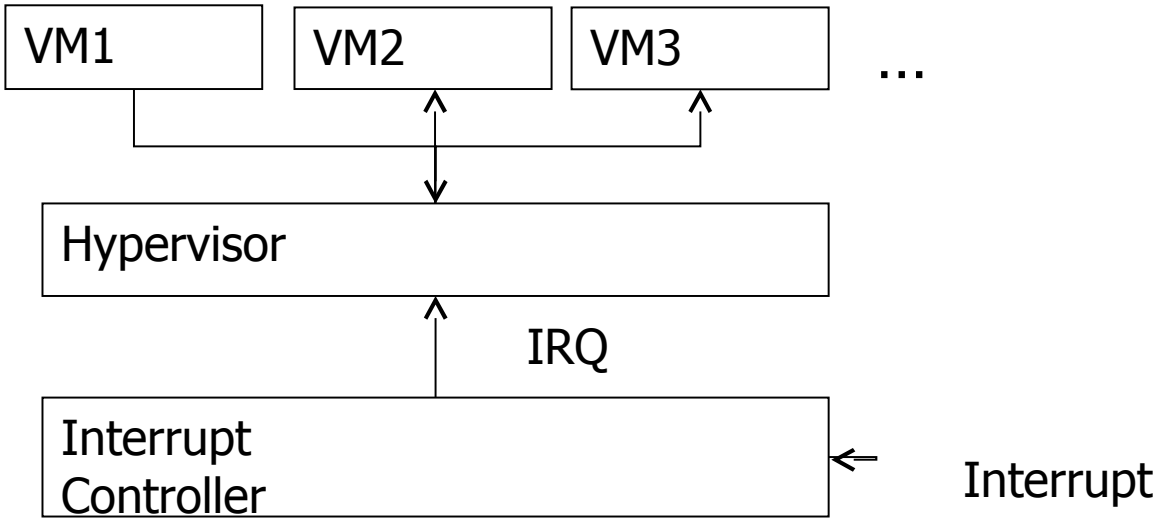


Überblick

1. Grundbegriffe
2. Virtualisierbarkeit von Hardware
3. Varianten der Virtualisierung
4. **Betriebsmittelverwaltung bei Virtualisierung**

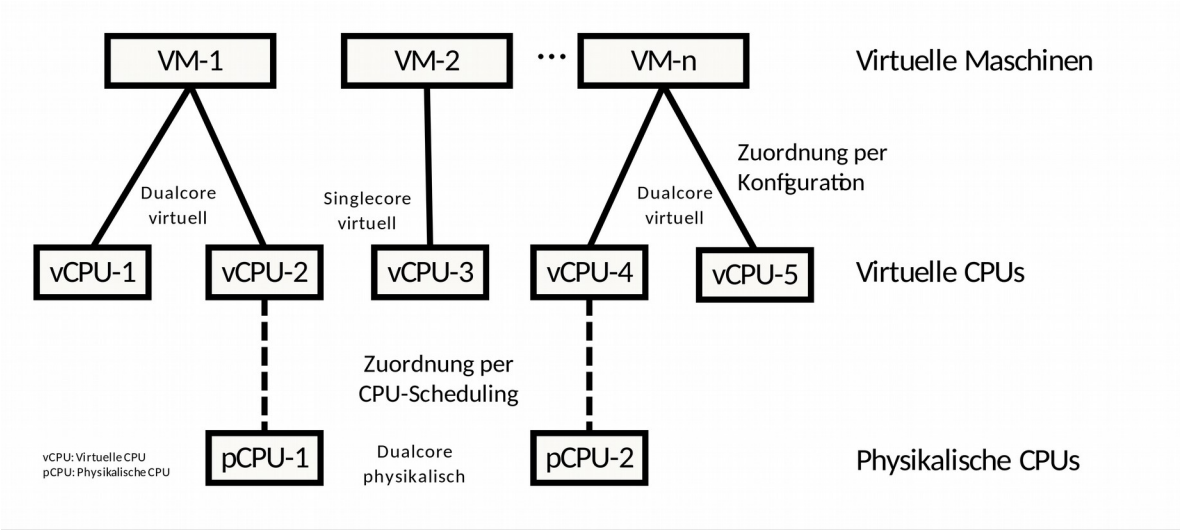
Interruptverarbeitung und Gerätesteuerung

- Hypervisor ist in der Rolle des **Multiplexers** für Interruptbearbeitung
- Xen Hypervisor: Privilegierte VM greift auf Geräte mit **realen Treibern** zu und kommuniziert mit virtuellen Treibern in den VMs



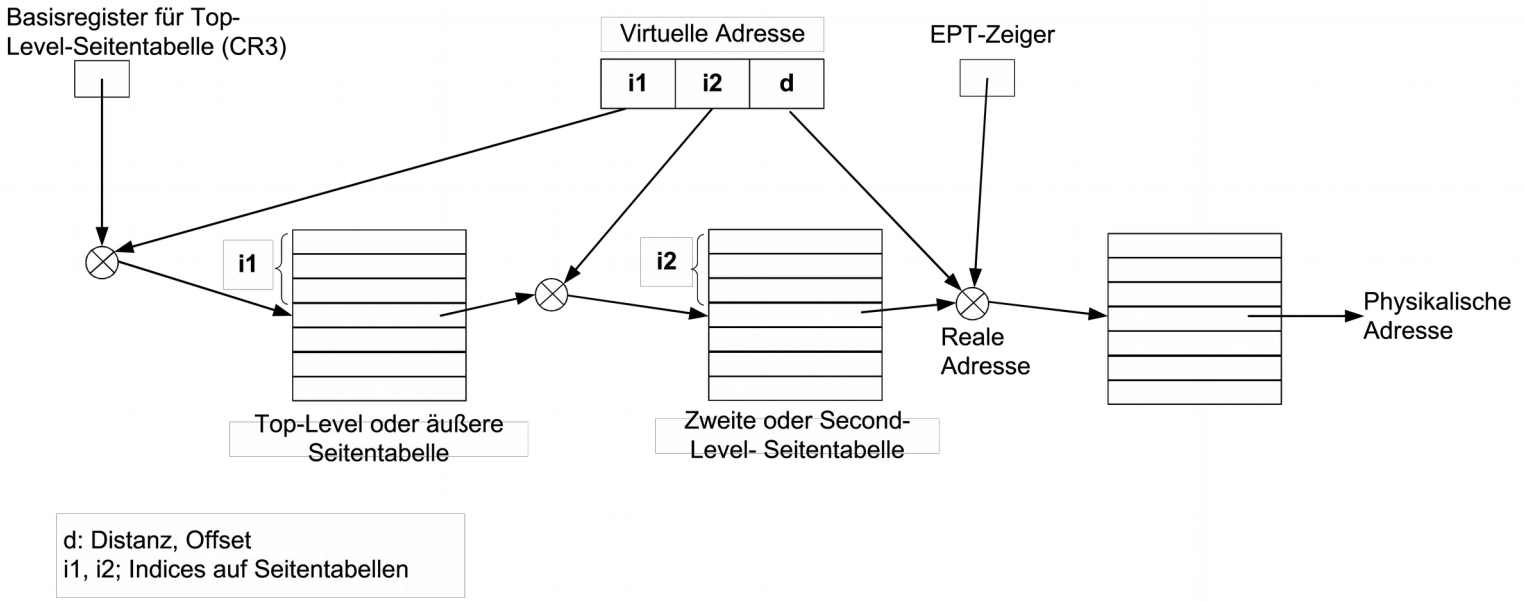
CPU-Scheduling

- **SEDF-Scheduler**: SEDF steht für *Simple Earliest Deadline First* ☾ alt
- **Credit-Scheduler:** Jeder vCPU wird initial ein Kredit gewährt, der zeitgesteuert reduziert wird
- **Co-Scheduling:** *für virtuelle Multiprozessoren*
  - streng
  - relaxed



Hauptspeicherverwaltung

- Abbildung: Virtuelle Adresse ☾ reale Adresse ☾ physikalische Adresse
- **Schattentabellen**
- **Extended Page Tables (**EPT) bei Intel in der MMU
- Weiterer Mechanismus: **Ballooning**



Zusammenfassung / Anmerkungen Virtualisierung

- Viele Varianten möglich, Grenzen verschwinden
- Paravirtualisierung und Hardwarevirtualisierung **gewinnen an Bedeutung**
- **Sicherheit ein Thema**
  - Z.B. VMware ESXi, kleine Virtualisierungs-Appliance (32 MB) zur Reduzierung von Angriffen
- Virtualisierung ist wichtig für Cloud Computing
  - Cloud Computing nutzt virtuelle Maschinen zur Abschottung untereinander und für die Skalierung
- Wichtig: Für den Betrieb sind **Management-instrumente** zur Administration erforderlich

Überblick

- ✓ Einführung in Computersysteme
- ✓ Entwicklung von Betriebssystemen
- ✓ Architekturansätze
- ✓ Interruptverarbeitung in Betriebssystemen
- ✓ Prozesse und Threads
- ✓ CPU-Scheduling
- ✓ Synchronisation und Kommunikation
- ✓ Speicherverwaltung
- ✓ Geräte- und Dateiverwaltung
- ✓ Betriebssystemvirtualisierung