TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Betriebssysteme

BETRIEBSSYSTEME UND SICHERHEIT

mit Material von Olaf Spinczyk, Universität Osnabrück

Cloud-Computing und Virtualisierung

https://tud.de/inf/os/studium/vorlesungen/bs

HORST SCHIRMEIER

Inhalt

- Wiederholung
- *Cloud-Computing*-Modelle
- *Cloud*-Systemsoftware
- Grundlagen der Virtualisierung
- CPU-Virtualisierung
- Speichervirtualisierung
- E/A-Virtualisierung
- Zusammenfassung

Literatur

Silberschatz:
--- :-(

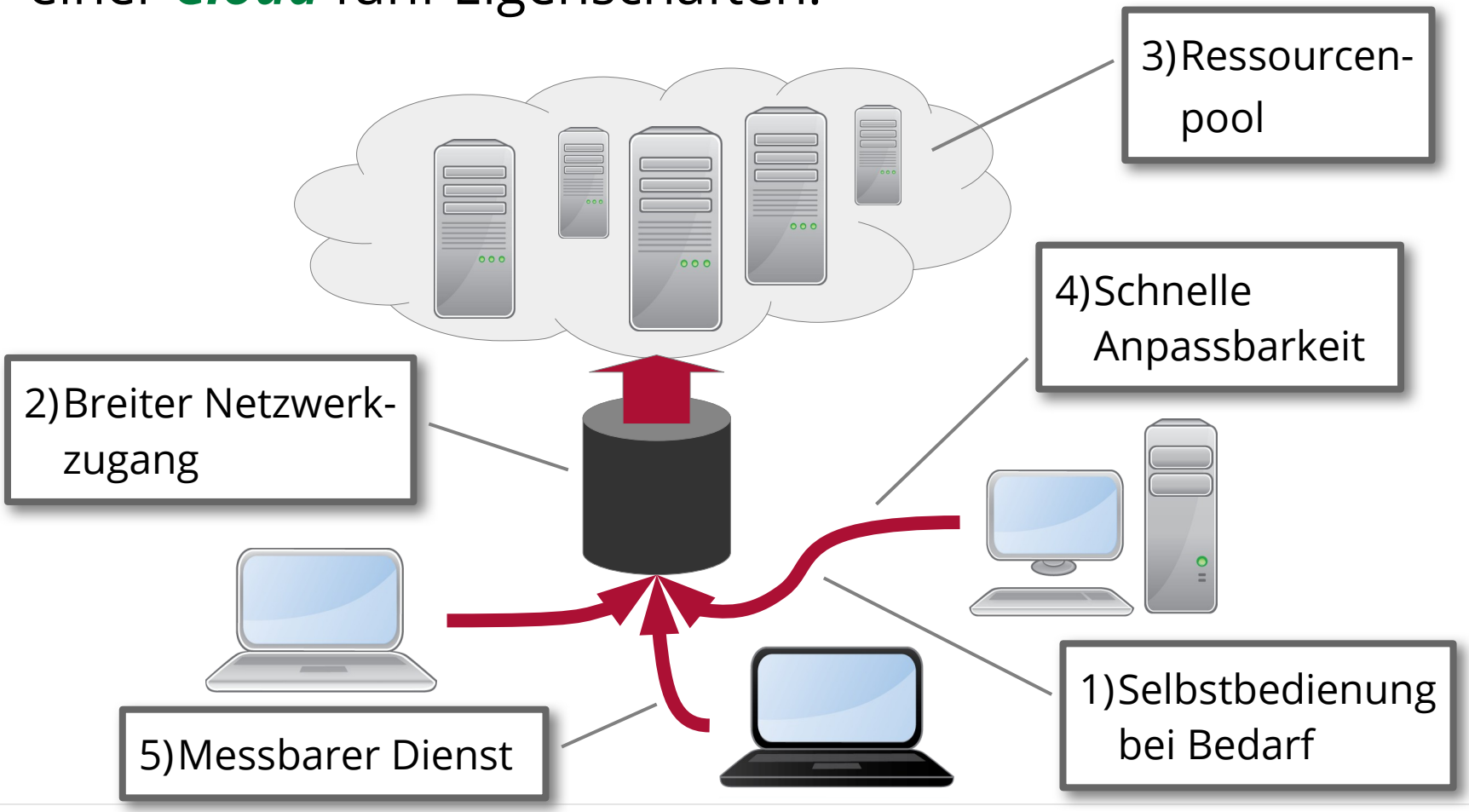
Tanenbaum: Kap. 7 ...
„Virtualisierung und die *Cloud*“

Inhalt

- **Wiederholung**
- *Cloud-Computing*-Modelle
- *Cloud*-Systemsoftware
- Grundlagen der Virtualisierung
- CPU-Virtualisierung
- Speichervirtualisierung
- E/A-Virtualisierung
- Zusammenfassung

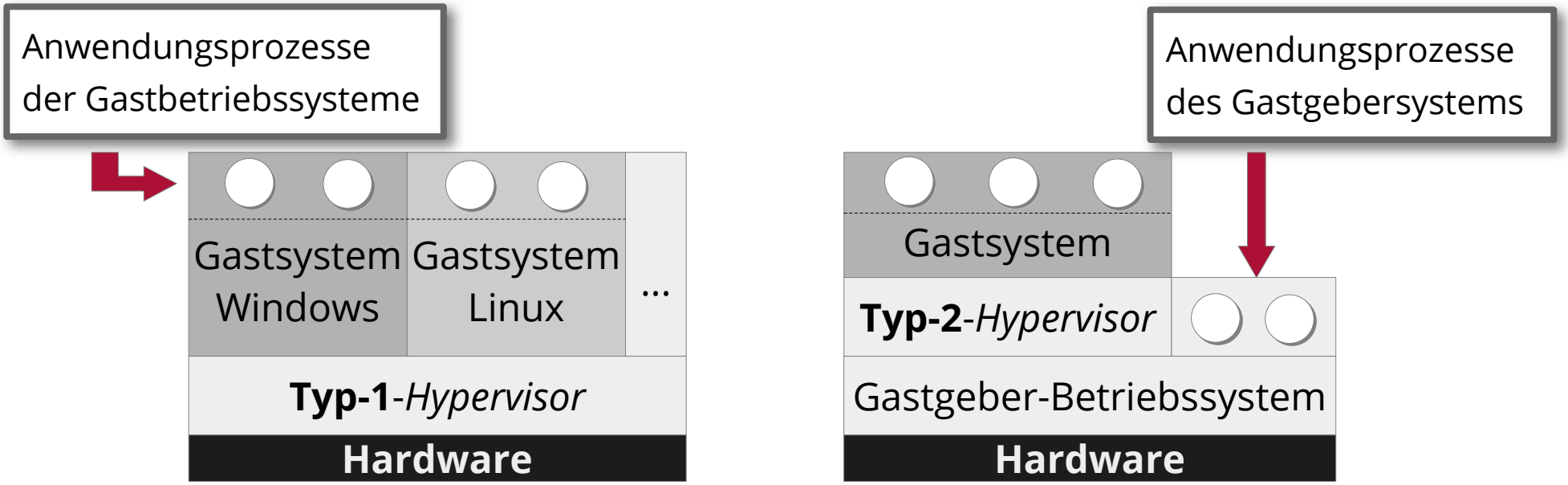
Cloud Computing

- Laut *National Institute of Standards and Technology* gehören zu einer ***Cloud*** fünf Eigenschaften:



Hardware-Virtualisierung

- … ermöglicht es in einer realen Maschine mehrere virtuelle Maschinen zu schaffen, die alle ihr eigenes Betriebssystem haben können.
  - Wichtige Grundlage für *Cloud-Computing* und *Server*-Konsolidierung.
- Technische Basis: ***Hypervisor*** / ***Virtual Machine Monitor***

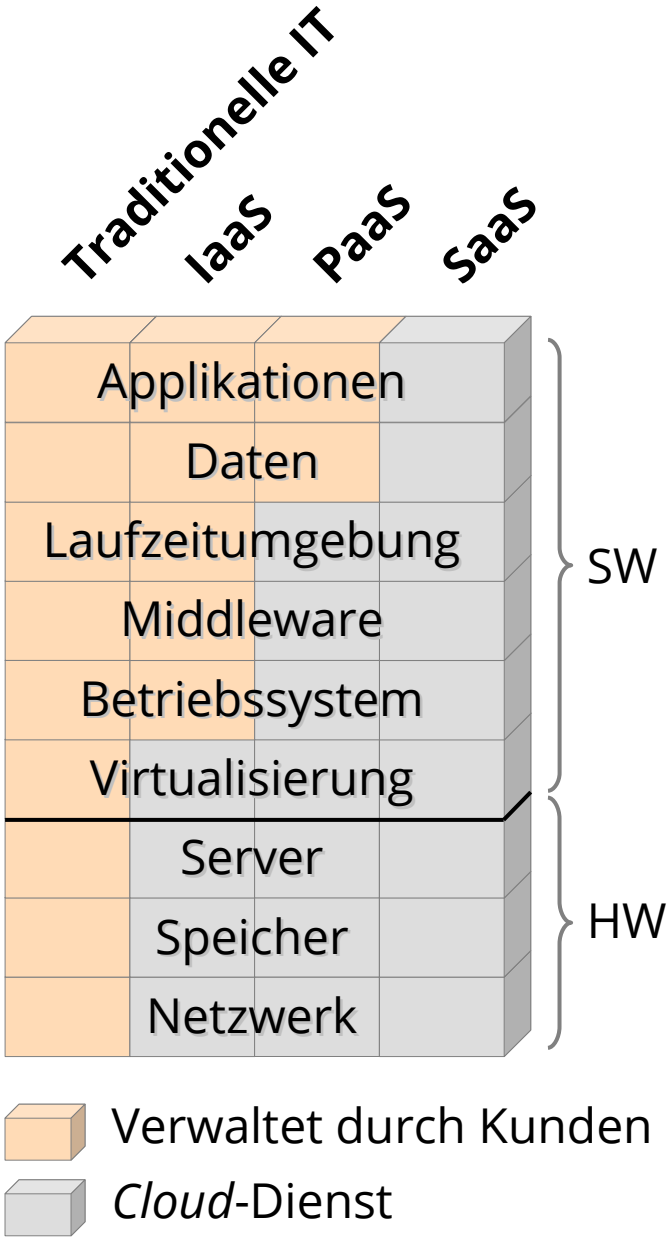


Inhalt

- Wiederholung
- ***Cloud-Computing*-Modelle**
- *Cloud*-Systemsoftware
- Grundlagen der Virtualisierung
- CPU-Virtualisierung
- Speichervirtualisierung
- E/A-Virtualisierung
- Zusammenfassung

Cloud-Dienstmodelle

- **SaaS – *Software-as-a-Service***
  - Cloud-Dienstleister stellt **fertige Anwendung** zur Verfügung.
  - z.B. Office365, Gmail, Zoom
- **PaaS – *Platform-as-a-Service***
  - **Ausführungsumgebung** für Anwendungen inkl. Betriebssystem und Laufzeitumgebung (je nach Prog.sprache)
  - z.B. Engine Yard, Google App Engine
- **IaaS – *Infrastructure-as-a-Service***
  - (Virtuelle) **Hardwareplattform**
  - z.B. Amazon EC2, Microsoft Azure



Quelle: Idee aus Stallings "*Operating Systems*"

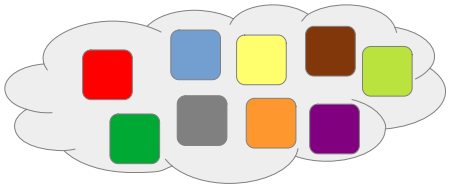
Diskussion: Schattenseiten der *Cloud*

Cloud-Computing bietet viele Vorteile, schafft aber auch ggf. neue Probleme, die bei der Planung berücksichtigt werden müssen:

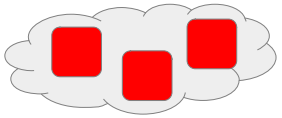
- **Datenschutz**
  - **Wo liegen** die Daten meiner Anwender/Kunden? Welche Datenschutzrichtlinien gelten in dem entsprechenden Land?
  - Ist der *Cloud*-Dienstleister vertrauenswürdig?
- ***Vendor-Lock-in***
  - Komme ich an meine Daten, falls ich den **Anbieter wechseln** möchte? Wenn ja, in welchem Format?
- **Dienstgüte**
  - Welche **Garantien** gibt mir der Dienstleister?

Bereitstellungsmodelle

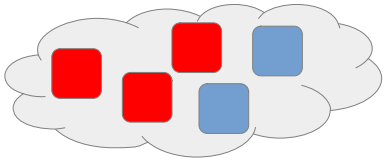
- **Öffentliche *Cloud*** (engl. *Public Cloud*)
  - *Cloud*-Dienstleister (engl. ***Cloud Service Provider – CSP***) arbeitet für beliebige Kunden.



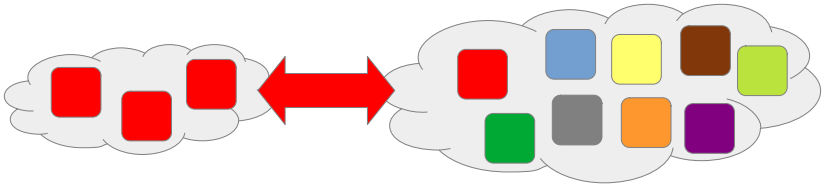
- **Private *Cloud*** (engl. *Private Cloud*)
  - Eine *Cloud*-Infrastruktur für ein (großes) Unternehmen. Dafür können eigene oder fremde Ressourcen genutzt werden. Man hat mehr Kontrolle.



- **Gemeinschafts-*Cloud*** (engl. *Community Cloud*)
  - Mehrere Kunden mit gleichen Anforderungen teilen sich eine *Cloud*-Infrastruktur.



- **Hybride *Cloud*** (engl. *Hybrid Cloud*)
  - Gemischtes Konzept.



Bereitstellungsmodelle – Vergleich

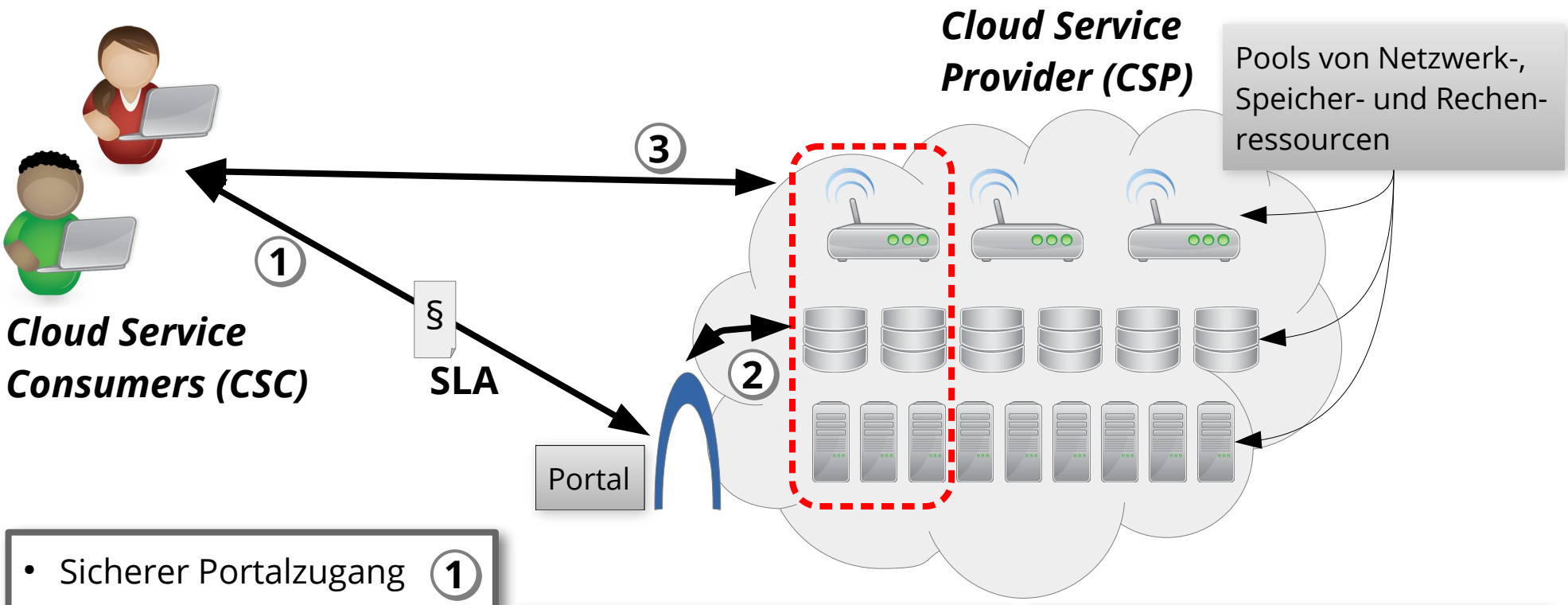
- Stallings: „*Operating Systems: Internals and Design Principles*“

	Private *Cloud*	**Gemeinschafts-*Cloud***	**Öffentliche *Cloud***	**Hybride *Cloud***
Skalierbarkeit	eingeschränkt	eingeschränkt	sehr hoch	sehr hoch
Datenschutz/ Sicherheit	sicherste Option	sehr sicher	moderate Sicherheit	sehr sicher
Leistung	sehr gut	sehr gut	niedrig bis mittel	gut
Zuverlässigkeit	sehr hoch	sehr hoch	mittel	mittel bis hoch
Kosten	hoch	mittel	niedrig	mittel

Inhalt

- Wiederholung
- *Cloud-Computing*-Modelle
- ***Cloud*-Systemsoftware**
- Grundlagen der Virtualisierung
- CPU-Virtualisierung
- Speichervirtualisierung
- E/A-Virtualisierung
- Zusammenfassung

Anwendungsfall / Anforderungen



- Sicherer Portalzugang
- Funktionsabfrage (z.B. VM-*Templates*)
- Auswahl des Dienstes
- Beobachtung und Anpassung

(1)

- Bereitstellung der benötigten Ressourcen
- Bestätigung des ***Service-Level Agreements*** **(SLA)** und der Kosten

(2)

- Nutzung (CSC) und Beobachtung (CSP)
- ***Management*** (CSP) (Migration, Redundanz, Energiesparen, Erweitern, ...)

(3)

Generelle Architektur eines *Cloud*-BS

Alle Ressourcen werden virtualisiert → IaaS Basis aller Dienste

Cloud-BS

API und GUI

Datenbanken und Objektspeicher
- Blockbasierter Speicher
- Dateibasierter Speicher (NFS, SMB, ...)
- Objektspeicher (Verteil. *Key/Value-Store*)

Management und Orchestrierung
- Verknüpfung aller Funktionen
- Steuerung der VM-Lebenszyklen
- Authorisation

Server-Verwaltung
- Virt. CPUs/Hauptsp.
- Beschleuniger (GPU, ...)
- Int./ext. Speicher

Massensp.-Verwaltung
- VM-Images
- *Backup*, *Snapshots*
- Pers. Speicher für VMs

Netzwerkverwaltung
- Bandbreitenreservier.
- Virtuelle Netze
- Adressverwaltung

Hypervisor-Technologie

Physische Infrastruktur


Standard *Rack-Server*


Standard Massenspeicher (DAS, NAS, SAN)


Standard Netzwerk-*Switches*

Strategische Fragen

- Wo platziert man VMs? Wann sollte man migrieren?
- Wie minimiert man SLA-Brüche? Wie viel **Überbuchung**?
- Lohnt es, Rechner frei zu machen und abzuschalten?

Diverse Strategien sind möglich:

Scheduler	Migrationen	Brüche			Strafen	Kosten	Gewinn	Marge (in %)
		CPU	RAM	UT				
FF	28.326	12	370	3.350	87.792,85	31.537,03	-58.523,54	-96,2
HPGBF	5.355	401	212	60	90.558,46	30.381,79	-60.133,91	-98,9
HPGOP	2.372	114	101	33	30.942,97	31.093,17	-1.229,80	-2,0
HPGWF	3.705	49	141	0	24.760,00	30.781,68	5.264,65	8,7
MMBF	1.111	6	69	40	10.516,92	31.350,98	18.938,43	31,1
MMOP	825	2	34	7	4.934,46	32.066,84	23.805,03	39,1
MMWF	890	2	34	0	4.400,00	31.736,97	24.669,36	40,6
MMWF + LB	871	3	26	0	3.600,00	32.326,96	24.879,37	41,0

Tabelle 10.10: Zusammenfassung der Ergebnisse der Intra-DC-Scheduler inklusive EE-Erweiterung und initialer VM-Verteilung nach RAM-Ressourcen für den Bitbrains RnD Trace (Monat 1) mit 500 VMs (In jeder Simulation wurden 60.806,34 Umsatz erzielt)

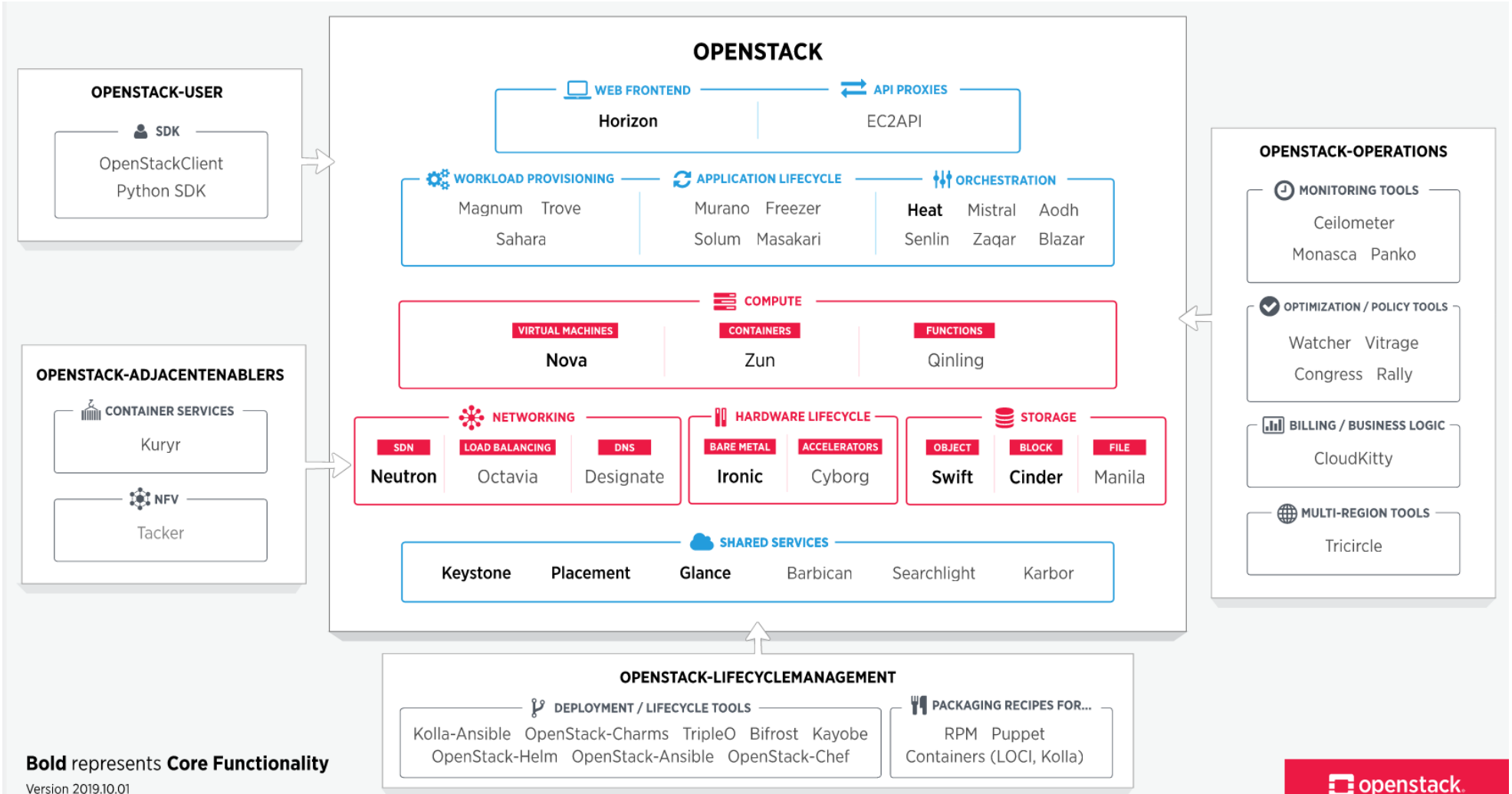
Es werden mehr Ressourcen verkauft als tatsächlich vorhanden sind.

Details zu den Strategien werden hier nicht weiter betrachtet.

Quelle:
Dissertation von A. Kohne,
„SLA-basierte VM-Scheduling-Verfahren für Cloud-Föderationen“

Beispiel: OpenStack

- *Open Source Cloud*-BS: www.openstack.org

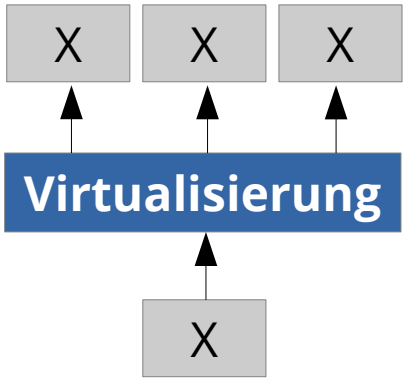


Inhalt

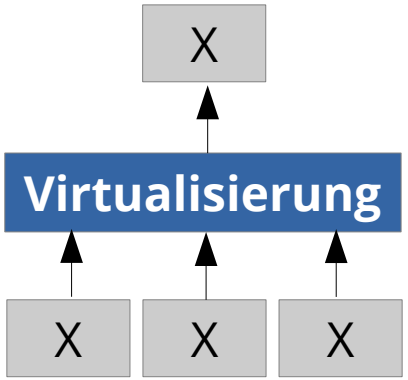
- Wiederholung
- *Cloud-Computing*-Modelle
- *Cloud*-Systemsoftware
- **Grundlagen der Virtualisierung**
- CPU-Virtualisierung
- Speichervirtualisierung
- E/A-Virtualisierung
- Zusammenfassung

Bedeutung/Nutzen der Virtualisierung

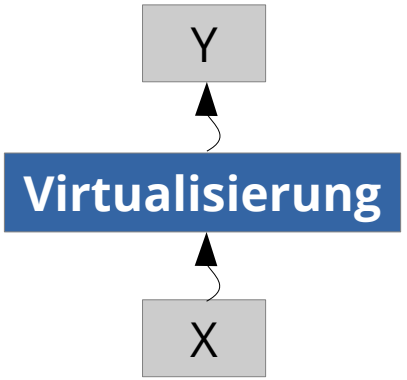
- Erzwingt die strikte Einhaltung einer Schichtenstruktur durch eine **Kontroll-/Eingriffsmöglichkeit** beim Ressourcenzugriff
- Grundlage für …



Multiplexing
z.B. „Virtueller Speicher“



Aggregation
z.B. *„Logical Volumes“*



Emulation
z.B. „C64-Emulator“

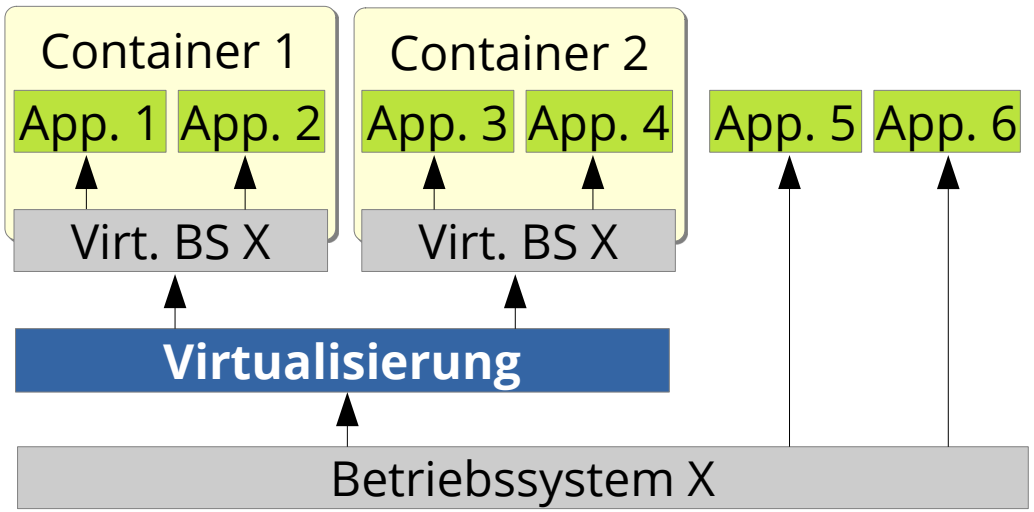
X u. Y sind Arten von Ressourcen, z.B. RAM, Platten, CPU, E/A-Geräte.

Quelle: [1]

- Konstruktionsprinzip kann auf verschiedenen Ebenen und für unterschiedliche Ressourcen wiederholt angewendet werden.

Container-basierte Virtualisierung

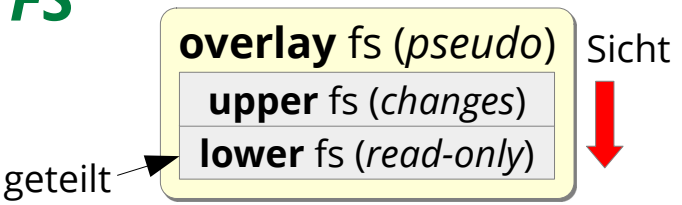
- kurz ***Container***
- Virtualisiert wird der Betriebssystem**kern**
  - Container teilen sich Kern
  - Bibliotheken/Systemprozesse können variieren



- Virtualisierungskomponente sorgt für ...
  - **Getrennte Sichten**, z.B. jeder Container “sieht” nur seine Prozesse
  - **Ressourcenpartitionierung**, z.B. bzgl. CPU-Zeit
  - **Effizientes *Sharing***, z.B. Duplizierung von Dateien vermeiden

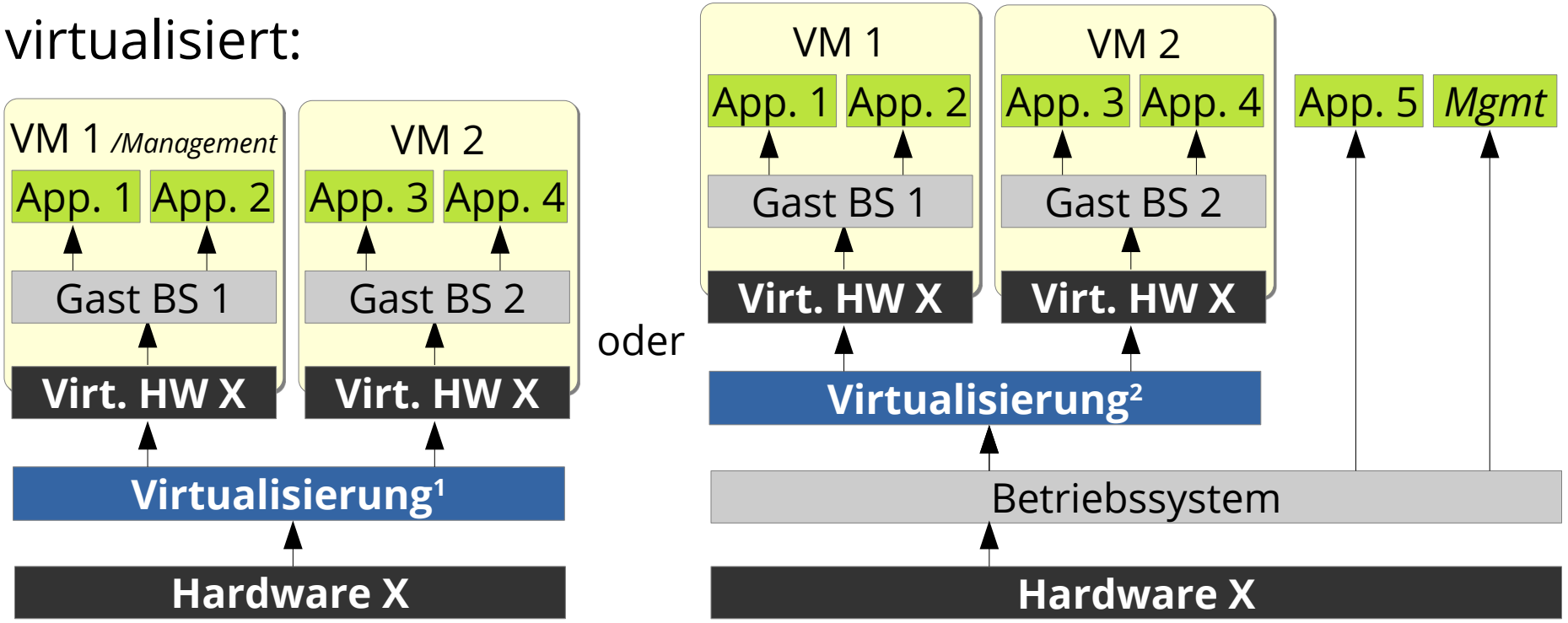
Beispiel: Linux-*Container*-Unterstützung

- Im Linux-Kern integriert
  - Container-Lösungen (z.B. Docker) haben lediglich Managementaufgaben
- Getrennte Sichten: ***Namespaces*** (dt. Namensräume) pro Task
  - … für Rechnernamen („UTS“), Prozesse („PID“), *Mount*-Punkte („Mount“), Netzwerkgeräte und -konfig. („Network“), IPC-Objekte („IPC“), *Control Groups* („Cgroup“, s.u.) sowie Systemzeit („Time“)
- Ressourcenpartionierung: ***Control Groups*** (cgroups)
  - Anteile der Container an CPU-Zeit, Speicher und E/A-Bandbreite
  - Schnittstelle zum Konfigurieren: Pseudo-Dateisystem **cgroupfs**
- Effizientes *Sharing* (von Dateien): ***Overlay FS***
  - Überlagerung von Verzeichnisbäumen



Hardware-Virtualisierung

- Eine komplette Maschine (CPU, Speicher, E/A-Geräte) wird virtualisiert:



[1]**Typ-1-Hypervisor** stellen virtuelle Maschinen ohne Hilfe eines Betriebssystems zur Verfügung (direkt auf der Hardware).

[2]**Typ-2-Hypervisor** arbeiten oberhalb eines „Host-Betriebssystems“. Sie können dessen Fähigkeiten nutzen, z.B. virtuellen Speicher.

Inhalt

- Wiederholung
- *Cloud-Computing*-Modelle
- *Cloud*-Systemsoftware
- Grundlagen der Virtualisierung
- **CPU-Virtualisierung**
- Speichervirtualisierung
- E/A-Virtualisierung
- Zusammenfassung

CPU-Virtualisierung (1)

- Einfachste Möglichkeit: **CPU-Emulation** (+ Multiplexing)
  - **Interpretation** oder ***Just-in-Time*-Übersetzung** der Instruktionen des emulierten Prozessors
  - Beispiele: Bochs, QEMU, MAME
- Nachbildung beliebiger CPU *Y* mit Hilfe von CPU *X*
- Problem: Ausführungsgeschwindigkeit

Execution Mode	T1FAST.EXE time	T1SLOW.EXE time
Native	0.26	0.26
QEMU 0.9.0	10.5	12
Bochs 2.3.5	25	31
Bochs 2.3.7	8	10

Table 3.2: Execution time in seconds of Win32 test program

FAST/SLOW: mit/ohne Code-Optimierung

```
static int foo(int i) {
  return(i+1);
}
int main(void) {
  … <start timer>
  for(i=0; i<100000000; i++)
    t += foo(i);
  … <stop timer>
}
```

Fazit: CPU-*Emulation* wenn möglich vermeiden

Quelle: [2]

CPU-Virtualisierung (2)

- Effiziente Möglichkeit: **CPU-Multiplexing** (CPU X_1, ..., X_N auf X)
- Gewünschte Eigenschaften („Virtualisierungskriterien“)
  - **Äquivalenz**: Eine VM verhält sich wie die reale Maschine.
  - **Sicherheit**: Eine VM ist isoliert. Der *Hypervisor* hat volle Kontrolle.
  - **Leistung**: Virtuelle CPUs sind nicht nennenswert langsamer als echte.
- Frage: Welche Architekturen sind so „virtualisierbar“?
- Antwort (Popek und Goldberg, 1974 [3]):
  - Es gibt „sensitive“ Instruktionen, die vom privilegierten Zustand der CPU (*User-/Supervisormode*, Speicherabbildung, ...) abhängen oder ihn verändern.
  - Alle sensitiven Instruktion müssen bei Ausführung im *User-Mode* zu einem *Trap* führen. Der *Hypervisor* kann damit die Instruktion emulieren.
- Alles Weitere wie in einem BS: VM-*Scheduling*

Inhalt

- Wiederholung
- *Cloud-Computing*-Modelle
- *Cloud*-Systemsoftware
- Grundlagen der Virtualisierung
- CPU-Virtualisierung
- **Speichervirtualisierung**
- E/A-Virtualisierung
- Zusammenfassung

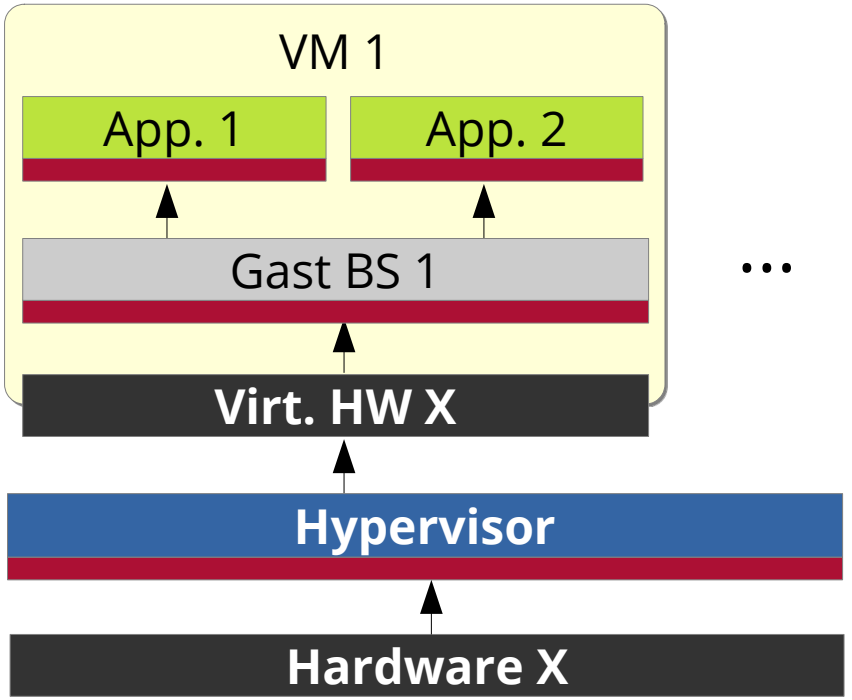
Speichervirtualisierung (1)

- Problem: Zusätzliche Adressabbildungsebene

Virtual Memory
(Virtueller (Gast-)Speicher)

Guest-Physical Memory
(Physischer Gast-Speicher)

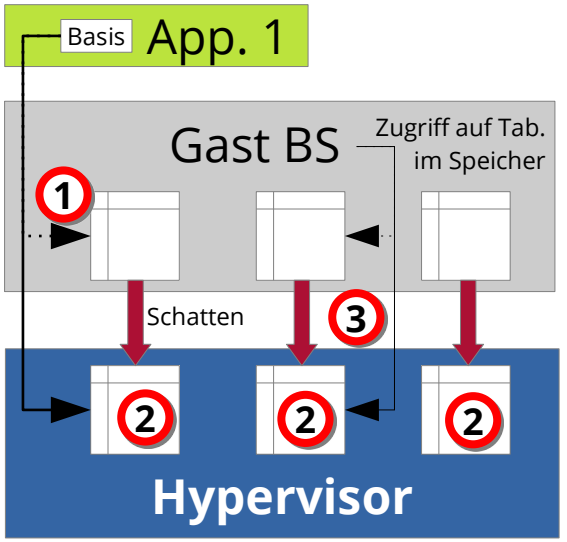
Host-Physical Memory
(Physischer (Gastgeber-/Host-)Speicher)



Gastbetriebssysteme *glauben* die Hardware voll unter Kontrolle zu haben. Sie nutzen Kacheln nach Belieben. Ohne zusätzliche Abbildung wären Überschneidungen mit anderen Gast-BS möglich!

Speichervirtualisierung (2)

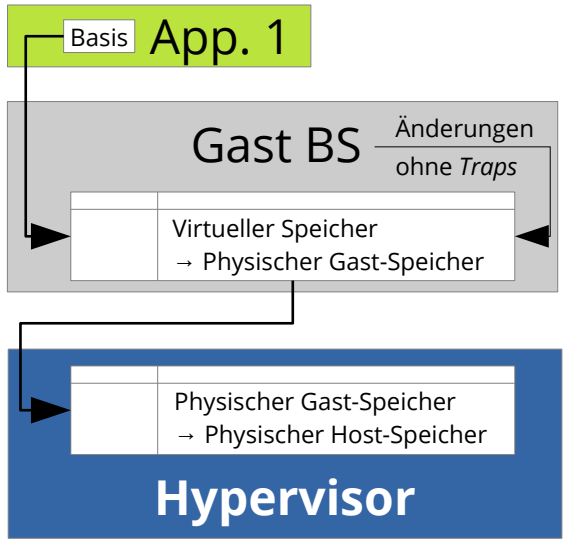
- **Lösung 1**: **Schattenseitentabellen** (***Shadow Page Tables***)
  - Erfordern keine spezielle Virtualisierungunterstützung der Hardware
- **Ansatz**:
  1. Seitentabellen der Gast-BS nicht benutzt
  2. *Hypervisor*: Schattentab. für jede Seitentab.
  3. Schattentab. muss aktuell gehalten werden!
     - Variante 1: Jeder Schreibzugriff auf Speicher, in dem eine Seitentabelle liegt, muss abgefangen und interpretiert werden.
     - Variante 2: Änderungen ignorieren; bei Seitenfehler Tabellen abgleichen
     - Beide Varianten führen zu vielen *Traps* in den *Hypervisor* → *Overhead*

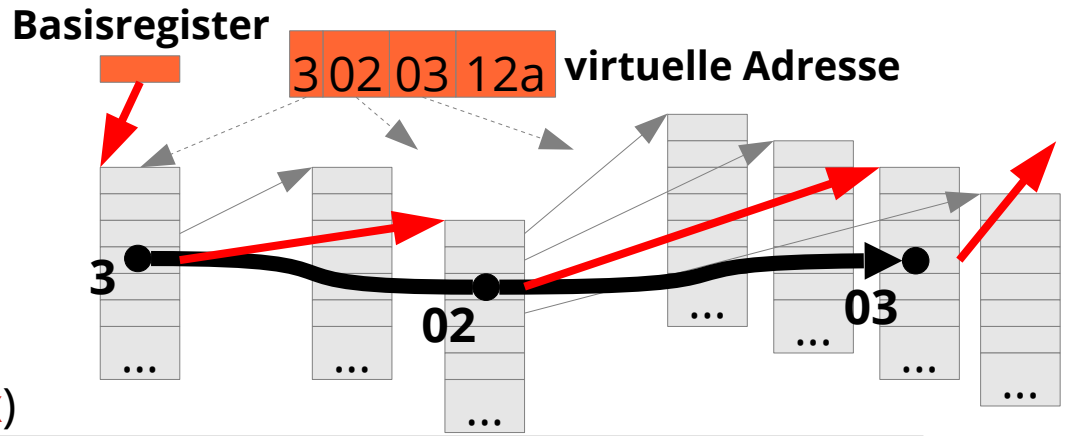


Schattenseitentabellen sind teuer. Abhilfe schafft nur Paravirtualisierung oder Hardwareunterstützung

Speichervirtualisierung (3)

- **Lösung 2**: **Verschachtelte Seitentab.** (***Nested Page Tables***)
  - AMD-Bezeichung; Intel: *Extended Page Tables*
- **Ansatz**:
  - Hardware übernimmt komplette Adressabbdg.
  - Gast-BS kann „seine“ Seitentab. beliebig ändern
  - Seitentabellensuche (engl. ***page table walk***) wird jedoch teurer → TLB wichtiger
    - Seitenkacheltabellen haben Baumstruktur
    - Verweise auf Tabellen sind physische **Gast-**Adressen
    - Umwandlung in physische *Host*-Adressen nötig (hier **4x**)





Speichervirtualisierung (4)

Was sonst noch alles geht …

- ***Ballooning***: Trick für dynamische Speicherzuordnung an VMs
  - Kleines Treibermodul kommuniziert mit Hypervisor,
  - bei Bedarf kann es Speicher des BS-Kerns reservieren,
  - dieser kann dann an andere VMs gegeben werden.
- **Deduplikation**: Erkennung und Vermeidung von Seitenduplikaten zwischen VMs. Spart Speicherplatz, z.B. bei demselben Gast-BS.
- **VM-Migration**
  - Kompletter VM-Speicherzustand wird anderem Host übergeben
  - Optimierung: Transfer der Seiten, während VM noch läuft
    - Letzte Änderungen werden mittels *Dirty*-Bits in *Page Table* erfasst.
- **VM-Replikation**
  - Zustandsänderungen im Speicher werden periodisch an *Backup-Host* übertragen. *Backup-VM* springt bei Rechnerausfall ggf. schnell ein.

Inhalt

- Wiederholung
- *Cloud-Computing*-Modelle
- *Cloud*-Systemsoftware
- Grundlagen der Virtualisierung
- CPU-Virtualisierung
- Speichervirtualisierung
- **E/A-Virtualisierung**
- Zusammenfassung

E/A-Virtualisierung (1)

- Einfachste Möglichkeit: **E/A-Emulation** (+ Multiplexing)
  - Zugriffe auf E/A-Register sind privilegierte Befehle oder können per MMU vom *Hypervisor* abgefangen werden (***„trap and emulate“***)
- Nachbildung beliebiger E/A-Geräte *Y* mit Hilfe von E/A-Gerät *X*
  - Beispiel Oracle VirtualBox: PS/2-**Mouse**/**Keyboard**; IDE, SATA, SCSI, … **Harddisk**; SVGA-**Graphik**; mehrere AMD- und Intel-**Netzwerkkarten**; **USB** Host Controller; AC‘97, Intel HD oder Soundblaster 16 **Audio**
- Problem: E/A-Durchsatz
  - Selbst einfache E/A-Operationen erfordern hunderte oder tausende von E/A-Register-Zugriffen!

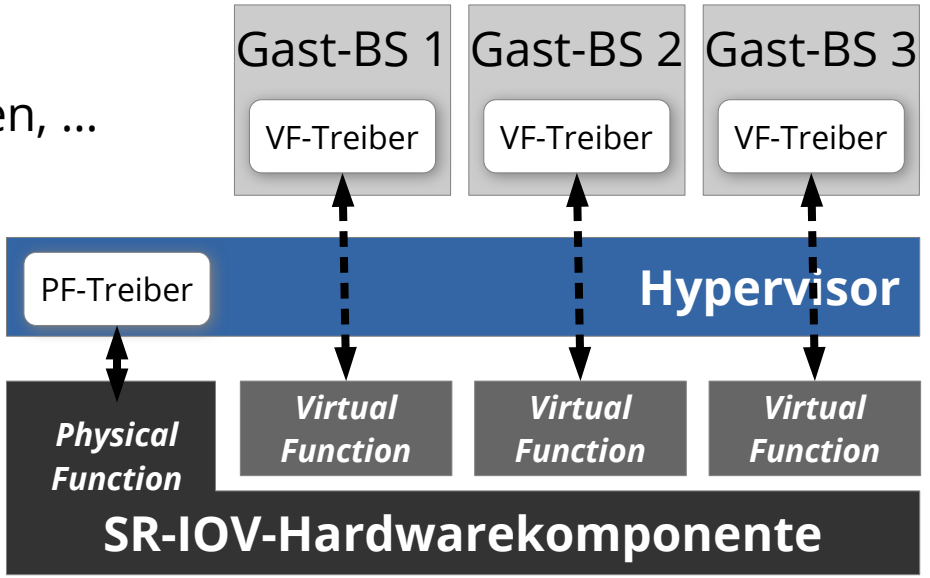
E/A-Emulation ist teuer. Abhilfe schafft (schon wieder) Paravirtualisierung oder Hardwareunterstützung.

E/A-Virtualisierung (2)

- Alternative: Verzicht auf Multiplexing – **Geräte-*Passthrough***
  - Gerät wird exklusiv genau einer VM zugeordnet
  - Beliebige Registerzugriffe werden erlaubt (ohne *Trap*)
- Probleme:
  - DMA-Adressen sind physische *Host*-Adressen, die die VM nicht kennt
    - Isolation könnte durchbrochen werden
  - *Interrupts* könnten auf „falscher“ CPU ausgelöst werden.
- Lösung: **I/O-MMU**
  - Hardware-Erweiterung: macht CPU bzw. Mainboard-Chipsatz
  - Bei DMA erfolgt eine **Adressabbildung** mittels Tabellen
    - Beschleunigung durch eigene TLBs
  - ***Interrupt-Remapping*** kann Interrupt-Nummer und Ziel-CPU ändern

E/A-Virtualisierung (3)

- Alternative 2: PCIe ***Single-Root-I/O*-Virtualisierung** (SR-IOV)
  - **Hardware-Mechanismus**: *Ein* Gerät erscheint wie *mehrere*
    - Mehrere E/A-Registersätze, mehrere Interrupt-Konfigurationen, …
  - Hypervisor blendet eines dieser Geräte in der VM ein und wird nicht mehr gebraucht.



- Mögliches Problem:
  - Hardware kümmert sich selbst um die Priorisierung der VMs
    - Zum Beispiel *Round Robin*
  - Widersprüche zu den Prioritäten des Hypervisors möglich

Inhalt

- Wiederholung
- *Cloud-Computing*-Modelle
- *Cloud*-Systemsoftware
- Grundlagen der Virtualisierung
- CPU-Virtualisierung
- Speichervirtualisierung
- E/A-Virtualisierung
- **Zusammenfassung**

Zusammenfassung

- Virtualisierung ist ein wichtiges wiederkehrendes **Architekturkonzept** im Systemsoftwarestapel
  - Transparent: *Multiplexing*, Aggregation, Emulation
- Hardware-Virtualisierung (im Sinne von Popek/Goldberg)
  - hebt die starre Bindung zwischen Hardware und Software auf
    - z.B. VM-Migration und -Replikation zur Laufzeit
  - Technische Grundlage des *Cloud-Computing*
- Betriebssysteme für *Clouds*
  - Wie gehabt: Ressourcenverwaltung und Abstraktionen
  - Aber auf einer höheren Ebene

Literatur

[1] Edouard Bugnion, Jason Nieh, and Dan Tsafrir. 2017. *Hardware and Software Support for Virtualization*. Morgan & Claypool Publishers, 2017.

[2] Mihočka, Darek, Stanislav Shwartsman and Intel Corp. *Virtualization Without Direct Execution or Jitting: Designing a Portable Virtual Machine Infrastructure*.", 2008.

[3] Gerald J. Popek and Robert P. Goldberg. 1974. *Formal requirements for virtualizable third generation architectures*. Commun. ACM 17, 7 (July 1974), 412–421. DOI:https://doi.org/10.1145/361011.361073