Virtualisierung

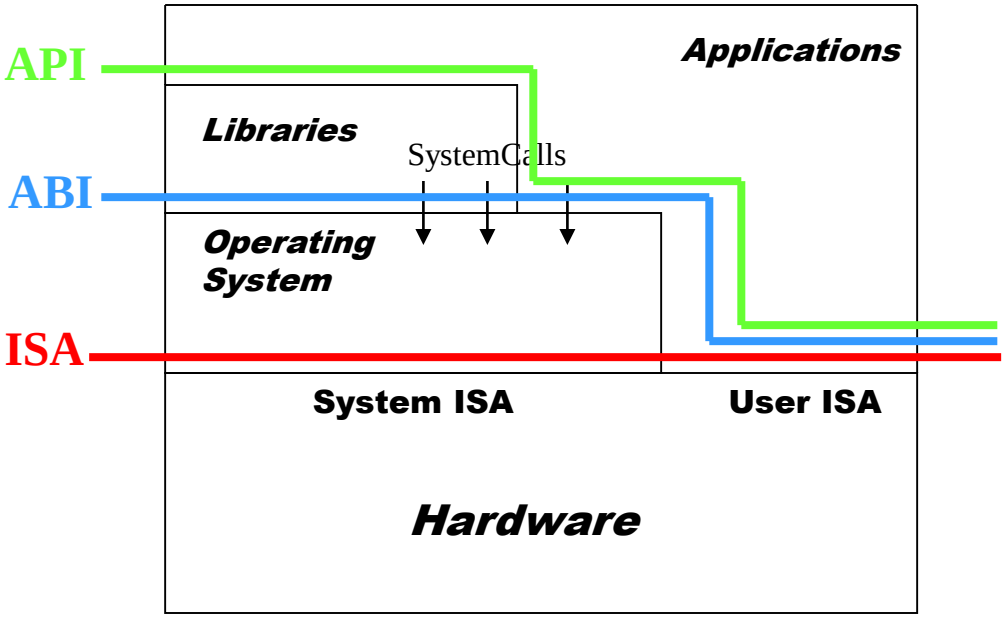
- Definition: „*In computing, virtualization (or virtualisation) is the creation of a virtual (rather than actual) version of something, such as a hardware platform, operating system (OS), storage device, or network resources.*" (Wikipedia)
- Beispiele
  - Virtueller Hauptspeicher (z.B. Auslagerungsdatei unter Windows)
  - Virtuelles Laufwerk (z.B. „Virtual Clone Drive")
  - Verteiltes Dateisystem
  - VPN (Virtual Private Network)
  - VM (Virtuelle Maschine)

IaaS und Virtualisierung

- Flexible Bereitstellung von Ressourcen durch virtuelle Maschinen (VMs)
  - Starten vorgefertigte (pre-compiled) VMs, die z.B. als Datei bereitstehen
  - Elastizität: Starten „beliebig vieler“ VMs auf „beliebig vielen“ Servern
- Unterschiedliche Arten von VMs
  - Vorausgesetzte Hardware-Eigenschaften (CPU, RAM, HDD, …)
  - Zusätzlicher Speicher kann integriert werden
  - Gast-Betriebssystem (Linux, Windows, …)
- Bezahlung nach VM-Typ und Nutzungsdauer (#Stunden)
- Herausforderungen
  - Isolated Execution (z.B. Speicher-Management)
  - Faire Verteilung der Ressourcen
  - Performanz im Vergleich Nicht-Virtualisierung

Architektur und Schnittstellen

- Maschinen-Architektur

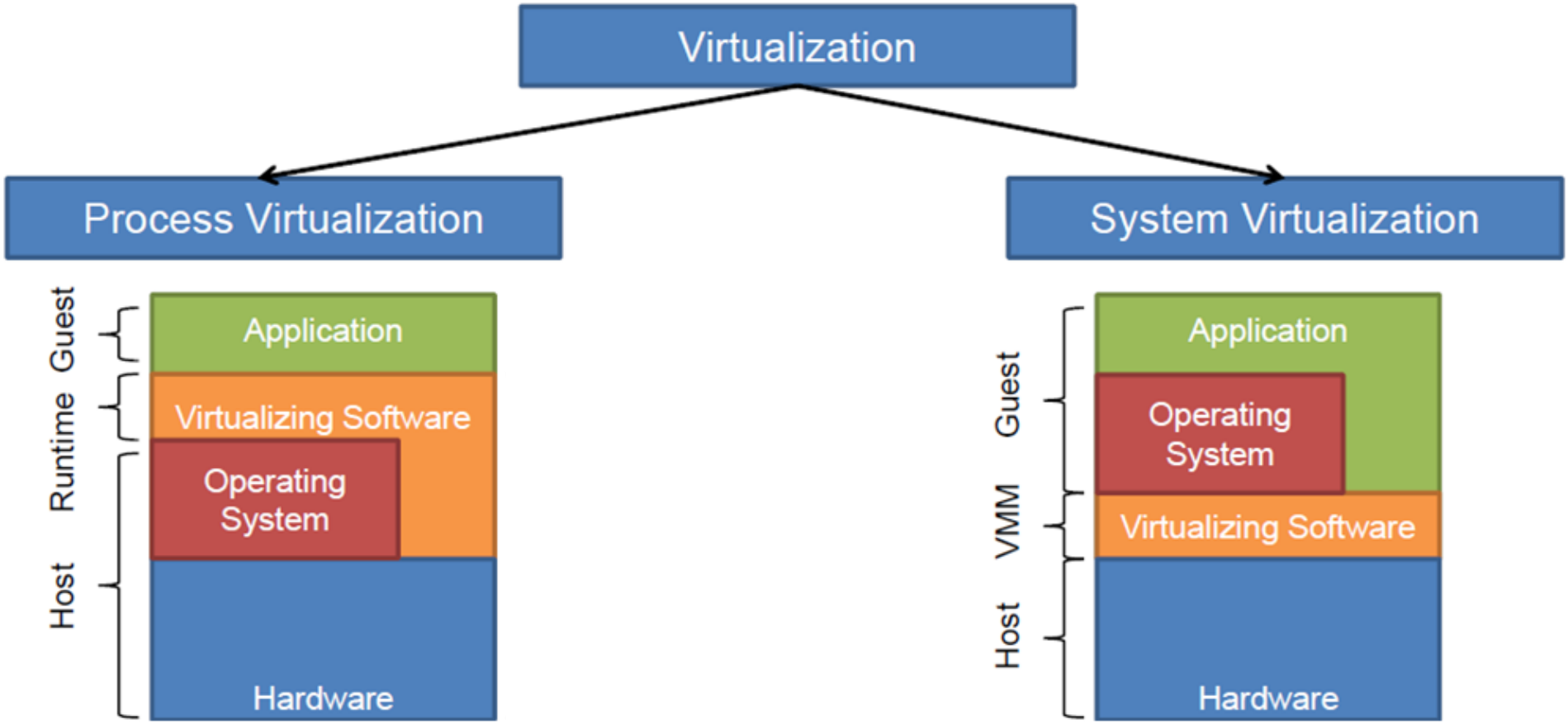


Quelle: CS5204 – Operating Systems @ Virginia Tech

  - API = Application Programming Interface
  - ABI = Application Binary Interface
  - ISA = Instruction Set Architecture

- Virtualisierung simuliert Schnittstellen

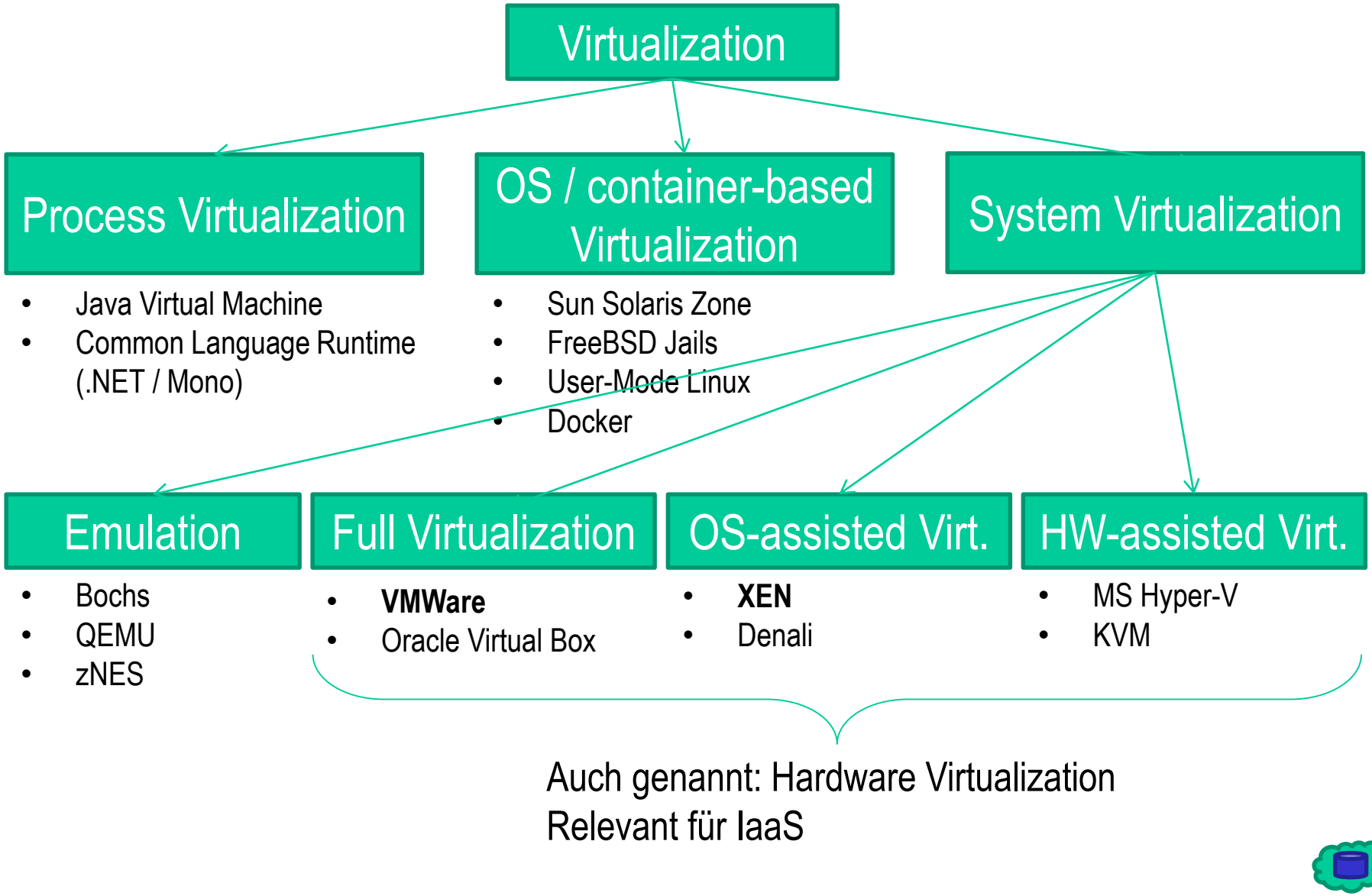
Arten der Virtualisierung (1)



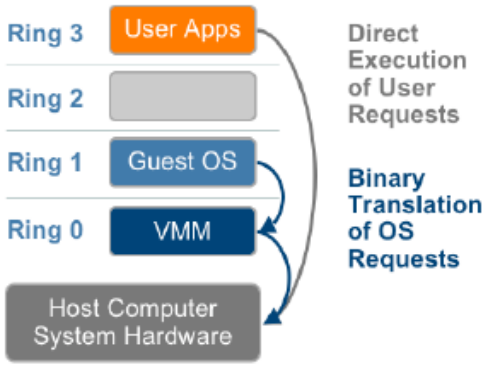
- Host = echtes System, Guest = virtuelle(s) System/Anwendung
- VMM = Virtual Machine Manager

Quelle Bild und folgende Folien: „Cloud Computing“-Vorlesung 2012, Daniel Warneke, TU Berlin

Arten der Virtualisierung (2)

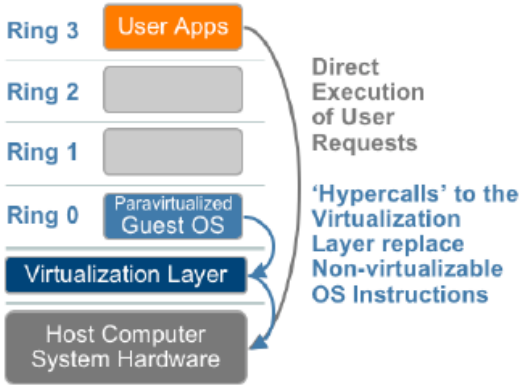


Full-Virtualization



- Gast-OS „weiß nicht“, dass es in einer VM läuft
  - Keine Anpassung des OS nötig, erfordert „virtualisierbare“ HW
  - Bekannter Vertreter: VMWare
- Handling von Privileged Instructions
  - Teil der Processor Instruction Sets, die nur in System Mode ausgeführt werden können (z.B. direkter Zugriff auf HS und CPU)
  - VMM emuliert Effekt von PrivInstr. auf Hardware (Ressourcen) („Trap and emulate“)
- Speicher-Management
  - Gast-OS hat keinen direkten Zugriff auf Machine Memory
  - Virtualisierung der MMU(Memory managment unit): Separierung von Speicher für Gast-OS und VMM (page tables, „shadow copies“)
  - Synchronisierung mittels Shadow Page Tables (Mapping Gast phys. Memory auf eigentl. Machine Memory)
- I/O-Management
  - i.Allg. Device-Nutzung durch mehrere VMs
  - Dedicated (z.B. Tastatur), Partitioned (z.B. HDD), Shared (z.B. Netzwerkadapter), Spooled (z.B. Drucker), Virtuell (z.B. Netzwerk-Interface)

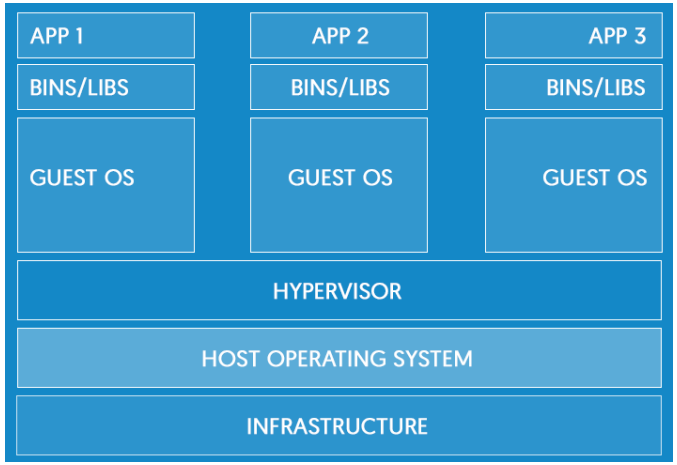
OS-assisted (auch: Paravirtualisierung)

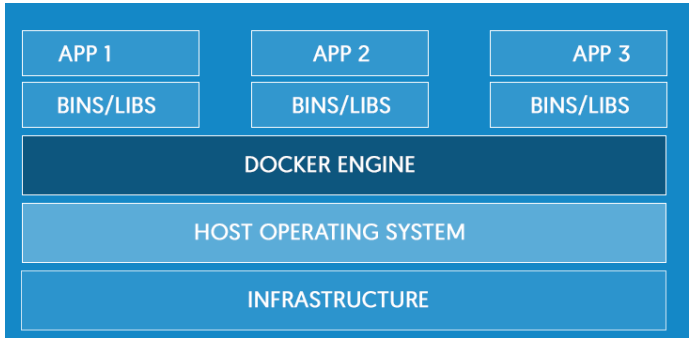


- Häufig genutzte Form von Virtualisierungs-Plattformen
- Gast-OS „weiß“, dass es in einer VM läuft
  - Anpassung des OS-Quellcodes nötig
  - OS unterhält Application Binary Interface (ABI)
- OS vermeidet Privileged Instructions (und damit Emulierung)
  - PrivInstr werden mit Hypercalls, die direkt mit VMM kommunizieren, ersetzt
- Gast-OS kann eigene Page Tables führen
  - Performanz-Steigerung bei Lesen
- I/O-Handling durch spezielle Device-Treiber
- Bekannter Vertreter: XEN
  - Kommerzielle und akademische Verbreitung (u.a. von Amazon EC2)
  - Unterstützung durch OS-Distributoren: openSuse, Debian, Ubuntu, …

Containerbasierte / Betriebssystem-Virtualisierung

- Virtualisierungsebene läuft als Anwendung innerhalb des Betriebssystems
- Bsp. containerbasierte Virt.: Docker
  - VMs/Container teilen sich Kernel des Host-OS
    - Kein Gast-OS, „leichtgewichtige VM“
    - Container enthält nur Anwendung und Abhängigkeiten (App + Bins/Libs)
  - Nutzen von Linux Kernel features wie cgroups, namespaces
    - Namespaces: isolieren Prozesse wie user lists, process lists, filesystem
    - CGroups: Isolation und Limitierung von Ressourcennutzung (CPU, memory, I/O)
  - Container-Management: z.B. Google Kubernetes, Apache Mesos





https://www.docker.com/what-docker

Virtualisierung: Vergleich

	Full	OS-assisted (para)	HW-assisted	Container
Modifiziertes Gast OS	Nein	Ja	Nein	Nein
Hardware Support nötig	Nein	Nein	Ja	Nein
Performanz	Gute Performanz für compute-intensive Apps (unprivileged instructions directly on CPU)	Verbesserte Performance (i.Vgl. zu full) durch Kooperation mit Gast-OS	Sehr gute Performanz (insbesondere für nicht-modifizierte OS)	Weniger speicherintensiv, kein Overhead durch durch Gast-OS, Portabilität, Effizienz
Nachteile	Schlechte Performanz für data-intensive Apps (IO erfordert syscalls → privileged instructions → context switches)	eingeschränkte Kompatibilität; Management-Overhead für OS-Versionen	Eingeschränkte Flexibilität wegen Hardware-Einschränkungen	Geringere Isolation, Ggf. Sicherheits-probleme (Zugriff auf Kernel-OS leichter)

Zusammenfassung

- Hardware-Infrastruktur
  - Data Center sind Basis einer effiziente Cloud-Infrastruktur
  - “Scale out” statt “Scale up”
- Software-Infrastruktur
  - Software/Platform/Infrastructure as a Service
  - Techniken für Performance & Availability
- Flexible Kombination von Cloud-Diensten
  - Beispiel AWS
- Virtualisierung als Grundlage für IaaS

Quellen und Literatur

- [BH09] Barroso and Hölzle: The datacenter as a computer: An introduction to the design of warehouse-scale machines. Morgan & Claypool, 2009
- [De09] Dean: Designs, Lessons and Advice from Building Large Distributed Systems. Keynote LADIS 2009
- [DGLS99] Devlin, Gray, Laing, Spix: Scalability Terminology, MS Tech Report, Dec. 1999
- [1] http://www.slideshare.net/gwendal/cloud-engineering
- [2] http://www.slideshare.net/Clogeny/cloud-computing-making-the-right-choices
- [3] http://www.slideshare.net/AmazonWebServices/building-powerful-web-applications-in-the-aws-cloud-a-love-story-jinesh-varia
- [4] http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/