

Betriebssystem-basierte Virtualisierung

Dr.-Ing. Volkmar Sieh

Department Informatik 4
Systemsoftware
Friedrich-Alexander-Universität Erlangen-Nürnberg

WS 2025/2026



Stellt ein Internet-Service-Provider viele (virtuelle) Maschinen zur Verfügung, werden viele Daten vervielfacht:

- Guest-OS
- Guest-Libraries
- Guest-Applikationen
- im Hauptspeicher
- auf der Platte



Web-Server (Linux mit Apache):

(Sehr grobe!) Rechnung:

	Hauptspeicher	Plattenspeicher
Linux-Kernel	4 MByte	80 MByte
Linux-Utilities	4 MByte	400 MByte
Linux-Libraries	50 MByte	100 MByte
Apache-Web-Server	10 MByte	20 MByte
Platten-Cache	82 MByte	–
Web-Daten	1 MByte	100 MByte
Summe	150 MByte	700 MByte



Für 100 Web-Server:

- 15 GByte Hauptspeicher
- 70 GByte Plattenplatz

Zusätzlich: Speicher für virtuelle Maschinen

- jeweils gleiche Daten:
 - Linux-Kernel
 - Linux-Utilities
 - Linux-Libraries
 - Apache-Web-Server
- jeweils unterschiedliche Daten:
 - Web-Daten
- dynamisch
 - Platten-Cache



- Internet-Service-Provider möchten viele Server bereitstellen.
- Server sollen voneinander unabhängig sein (Intrusion, Admin-Fehler, ...)
- OS auf den Servern in vielen Fällen gleich.

Idee:

Ein Betriebssystem nehmen, aber Applikationen, Daemons, User und Admins gegeneinander abschotten.

„Virtual Private Server“



Jeder Admin soll Betriebssystem-Dateien installieren, löschen und ändern können.

=>

Möglichkeiten:

- Jeder virtuelle Rechner hat eigene Dateien.
(Ähnlich: Unix-chroot-Umgebung)
- System-Dateien und -Directories sind read-only.
(„immutable“-Files)
- System-Dateien und -Directories sind copy-on-write.



Jeder Rechner kennt Methoden zur lokalen Inter-Prozess-Kommunikation (Unix: Shared-Memory, Semaphoren, Messages, Pipes, Named-Pipes). Über diese Methoden darf nur „Rechner-lokal“ kommuniziert werden.

=>

Jeder virtuelle Rechner braucht eigene Inter-Prozess-Kommunikations-Objekte.



Auf jedem virtuellen Rechner sollen Dienste laufen können. Jeder Admin soll nur seine Dienste starten/stoppen können.

=>

Jeder virtuelle Rechner braucht eigene Prozess-Liste.

(Unix: jeder virtuelle Rechner braucht eigenen `init`-Prozess.)



Jeder Admin soll neue Accounts anlegen/löschen können.

=>

Jeder virtuelle Rechner kennt andere User/Admins.



Wenn jeder virtuelle Rechner Internet-Dienste anbieten soll, braucht er eigene Internet-Ports.

=>

Jeder virtuelle Rechner braucht eine eigene Internet-Adresse und einen eigenen Satz von Internet-Ports.

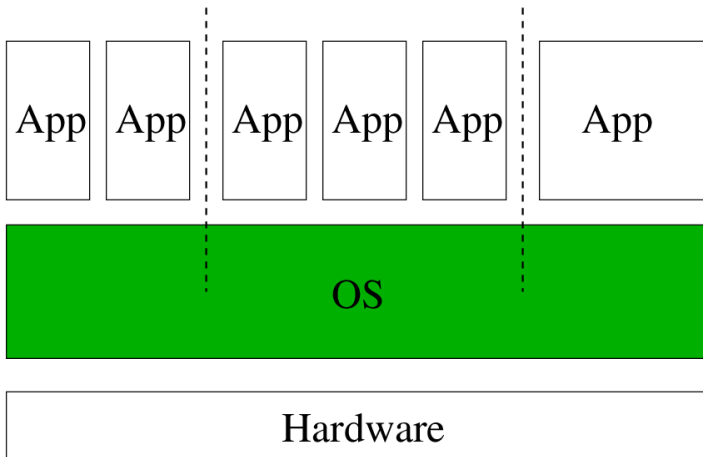


Ein übergeordneter System-Administrator muss virtuelle Rechner erzeugen, starten, stoppen und löschen können.

=>

Ausgezeichnete virtuelle Maschine mit neuem System-Call notwendig.





Änderungen notwendig in

arch: neuer System-Call, Clone von Kernel-Threads

drivers: „vroot“-Device

fs: neue Rechte-Überprüfung

include: neue xid-Einträge (Nummer der VM)

ipc: jede VM braucht eigene IPC

kernel: jede VM braucht eigene Prozesse, eigene Timer, einen eigenen Namen, neue Rechte-Überprüfung, ...

mm: Ressourcen-Checks

net: jede VM hat eigene IP(s)



Virtuelle Rechner können sich ggf. durch übermässigen Ressourcen-Konsum gegenseitig stören.

Daher muss dafür gesorgt werden können, dass jeder virtuelle Rechner

- nicht zu viel CPU-Zeit verbraucht,
- nicht zu viel Memory braucht,
- nicht zu viel Dateisystem-Platz belegt.

Normale System erlauben es, den Ressourcen-Verbrauch *einzelner* Prozesse zu beschränken. Hier muss der Verbrauch *aller* Prozesse einer virtuellen Maschine eingeschränkt werden.



- Vorteile:
 - sehr geringer Overhead (i.A. $< 1\%$)
 - Ressourcen (CPU, Speicher, Platte) können einfach geshared werden
 - jeder Server braucht nur ca. 100MB Plattenplatz
 - jeder Server braucht ggf. $< 1MB$ Hauptspeicher
- Nachteile:
 - nur jeweils gleiches OS möglich
 - Beschränkung des Ressourcen-Verbrauchs kompliziert
 - Admins können nur eingeschränkt booten
 - für ein Kernel-Update müssen *alle* virtuellen Maschinen neu booten
 - Abschottungen im Kernel u.U. unsicher
 - Abschottungen müssen für jede Kernel-Version neu an den Kernel angepasst werden

