Bibliotheks-basierte Virtualisierung

Dr.-Ing. Volkmar Sieh

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

WS 2024/2025

Bibliotheks-basierte Virtualisierung

Idee:

Will man eine bestimmte Applikation laufen lassen, muss „nur" das **ABI** für das Programm korrekt vorhanden sein.

Application Binary Interface („**ABI**"):

- CPU-Instruktionen
- Memory-Layout
- (Shared-) Library-Interface
- Betriebssystem-Interface

Werden System-Calls über die Library abgewickelt, entfällt der letzte Punkt.

Bibliotheks-basierte Virtualisierung – CPU-Instruktionen

CPU-Instruktionen:

Es müssen die CPU-Instruktionen vorhanden sein, die das Programm nutzt.

=> CPU muss „stimmen"

Beispiel: Windows-Programm auf einer Sparc-Prozessor-basierten Workstation auszuführen funktioniert daher mit dieser Methode *nicht*!

Bibliotheks-basierte Virtualisierung – Memory-Layout

Memory-Layout:

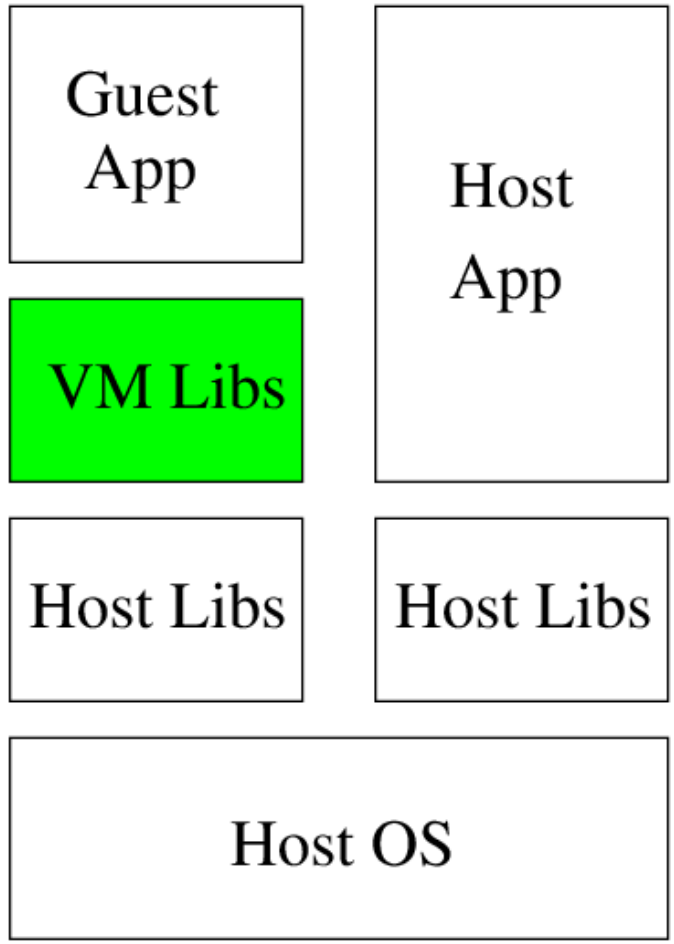
- Da, wo das zu startende Programm seinen Programm-Code erwartet, muss das Code-Segment liegen.
- Da, wo das zu startende Programm sein Daten-Segment erwartet, muss Read/Write-Memory sein.
- Da, wo das zu startende Programm sein Stack-Segment erwartet (i.A. dort, wo %esp hinzeigt), muss Read/Write-Memory sein.

Verhindert es das Host-OS, an die entsprechende Stelle Speicher zu mappen, ist diese Virtualisierungs-Methode nicht anwendbar!

Beispiele:

- Windows verhindert Mappings unterhalb von 1MB und oberhalb von 2GB.
- Linux verhindert Mappings oberhalb von 3GB.

Bibliotheks-basierte Virtualisierung



Bibliotheks-basierte Virtualisierung

- Vorteile:
  - hohe Performance (nahe 100%)
  - geringe Ressourcen-Anforderungen (z.B. kein zweites OS)
  - Gast-Applikation läuft im Fenster parallel zu anderen Native-Applikationen
- Nachteile:
  - ein Update der Original-Libraries erfordert ein Update der nachprogrammierten Libraries
  - neue Original-Libraries erfordern neue nachprogrammierte Libraries
  - Nachprogrammieren der Libraries u.U. schwierig (keine Dokumentation, kein Source-Code)
  - nachprogrammierte Libraries u.U./i.A. fehlerbehaftet

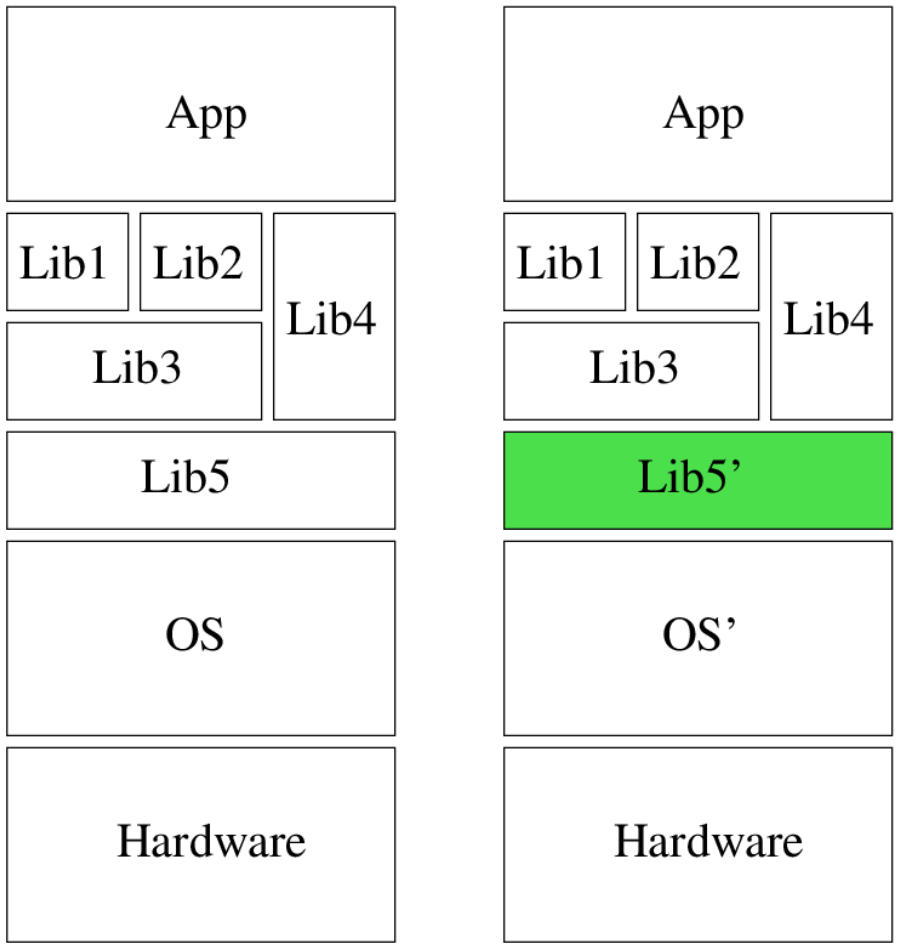
Bibliotheks-basierte Virtualisierung

I.A. existieren viele Libraries (Beispiel: Linux/Windows je ca. 100).

- höhere Schichten können ggf. unverändert bleiben
- niedrigere Schichten werden ersetzt

Z.B. würde es reichen, die `libc` zu ersetzen, wenn man Linux-Programme auf anderen Plattformen ablaufen lassen wollte. (Beispiel: `cygwin`)

Bibliotheks-basierte Virtualisierung



Bibliotheks-basierte Virtualisierung

Kennt das Betriebssystem den Applikation-Typ nicht, kann es ihn nicht laden und mit den Libraries linken.

Möglichkeiten:

- Betriebssystem erweitern (Beispiel Linux: `binfmt_*`-Module laden, linken und starten entsprechende Applikationen)
- Betriebssystem startet Extra-Programm, dass die eigentliche Applikation lädt, linkt und startet (Beispiel: `wine`-Programm lädt, linkt und startet Windows-Applikation unter Linux)

Entspricht Funktionalität eines Shared-Library-Linkers.

Bibliotheks-basierte Virtualisierung

Problem:

Programme, die System-Calls enthalten (z.B. statisch gebundene Programme)

Lösung:

Betriebssystem muss „fremde" System-Calls bereitstellen („**Personality**", „**Execution Domain**").

Beispiel: FreeBSD-Kernel kennt „Linux Personality" => Linux-Programme unter FreeBSD lauffähig.

Bibliotheks-basierte Virtualisierung