Geschichte der Virtualisierung

20. Juni 1959: erste Idee von der Virtualisierung mit der Publikation „Time Sharing in Large Fast Computers" von Christopher Strachey, (ein logischer Prozessor, auf dem Programme wie auf einem realen laufen). Ein Scheduler ordnet den logischen dem physikalischen Prozessor zu.

7. Dez. 1962: der ATLAS Computer beinhaltete einen einstufigen virtuellen Speicher mit Demand Paging

60er Jahre: eine IBM 7044 dient als Hauptrechner, auf dem mehrere virtuelle Maschinen (VM) des Typs IBM 7044 ausgeführt wurden.

1971: praktische Anwendungen auf Mainframe-Ebene (IBM 370) mit dem CP/CMS-40 und dem VM/370 = erste virtuelle Maschine (Differenzierung in privilegierte und unprivilegierte Instruktionen)

Febr. 1972: erste theoretische Arbeiten zur Virtualisierung von Goldberg mit seiner Doktorarbeit „Architectural Principles of Virtual Machines“

Geschichte der Virtualisierung

Bis dahin war eine virtuelle Maschine eine Kopie einer realen Maschine.

1977: Kenneth Bowles entwickelte eine Pseudo-Maschine (Emulator, der die Funktionalität anderer Rechenmaschinen nachbildet)

….

1991: die von Sun Microsystems Inc. entwickelte und sehr verbreitete „Java Virtual Machine“ agiert ähnlich der P-Maschine. (Die JVM wird auf einem realen Rechner emuliert)

1993: mit der von Sun Microsystems Inc. entwickelten Software „Wabi“ (Windows Application Binary Interface) konnte erstmalig Software ohne Veränderungen auf einem nicht proprietären Betriebssystem verwendet werden

1993: mit Wine (WINE Is Not an Emulator) konnte man Windows 3.1 Programme unter Linux ausführen

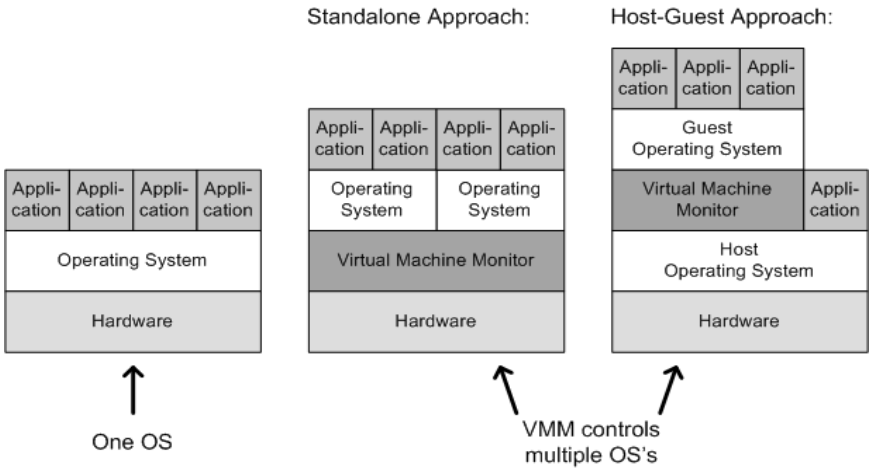
Geschichte der Virtualisierung

1999: Mit einem modifizierten Kernel von Linux, User Mode Linux (UML) genannt, war es Jeff Dike gelungen, mehrere Instanzen eines Linux-Betriebssystems gleichzeitig auf derselben Architektur auszuführen

1999: VMware präsentiert die „VMware Workstation“, die erstmalig einen vollständigen x86-Computer auf einem anderen x86-System virtualisieren konnte

Heute: Mit den Mehrkern-Prozessoren erobern Virtualisierungen den Mainstream-Bereich

Übersicht



Was ist Virtualisierung?

- Unter Virtualisierung versteht man das Ersetzen einer direkten Schnittstelle zwischen einer Ressource (Hardware) und ihrem Nutzer (Betriebsystem) durch eine indirekte, Software-vermittelte Verbindung, der **Virtualisierungsschicht**.
- die Abstraktionsschicht suggeriert dem Gastsystem, dass es der alleinige Nutzer einer Ressource sei. Das in einer virtuellen Maschine befindliche Programm soll sich ebenso verhalten wie in einem äquivalenten realen System
- Virtualisierung bezeichnet Methoden, die es erlauben, Ressourcen eines Computers zusammenzufassen oder aufzuteilen und damit besser auszulasten [Wiki]
- Der auf dem realen System eingesetzte **Hypervisor**, auch **Virtual Machine Monitor** genannt, besitzt die Kontrolle über die virtuelle Maschine. Er fängt kritische Befehle ab, bei denen die Betriebssysteme direkt auf die Hardware zugreifen und regelt die Verwaltung von I/O-Ressourcen, wie Netzwerk, Festplatten usw..

Was ist Virtualisierung?

- die tatsächlich vorhandene Hardware eines physischen Systems wird (auch mehrfach) nachgebildet
- mehrere (heterogene) Hardwareressourcen werden zu einer homogenen Umgebung zusammengefügt.
- Eine virtuelle Maschine bildet nicht alle Befehle nach, sondern nur kritische, die Konflikte zwischen den virtuellen Maschinen verursachen können. Das sind i.a. Befehle, die im sog. **Kernel Modus** ablaufen. (Zugriffe auf nur einmal vorhandene Hardware wie Timer, MMU, Int,..)
- bei der Nachbildung eines architekturgleichen Rechners werden die meisten Befehle direkt an die Hardware weitergereicht und mit nativer Performance abgearbeitet
- Virtualisierung erlaubt die Koexistenz mehrerer, unterschiedlicher Betriebssysteme parallel und gleichzeitig auf einer isolierten Maschine. Dabei können sich inkompatible Betriebssysteme eine Hardware teilen
- keine VM kann auf den Speicherbereich der anderen zugreifen.

Arten der Virtualisierung

- Anwendungsvirtualisierung
- Emulation
- Host-gestützte Virtualisierung
- Vollvirtualisierung
- Virtualisierung auf Hardware-Ebene
- Paravirtualisierung
- Partitionierung
- Windows-Echtzeit-Erweiterungen
- Rekursive Virtualisierung

Arten der Virtualisierung

- Speichervirtualisierung
- Desktopvirtualisierung
- Servervirtualisierung

Anwendungsvirtualisierung

- Anwendungsvirtualisierung (application virtualization) ist das lokale Ausführen von Anwendungen, ohne dass diese installiert werden müssen
- Bei der Anwendungsvirtualisierung befindet sich zwischen der auszuführenden Anwendung und dem Betriebssystem eine Abstractionlayer
- Diese stellt die von der Anwendung benötigten Ressourcen, Registry, Softwareschnittstellen, usw. bereit
- Die Anwendungen werden lokal in einer virtuellen Umgebung ausgeführt

Anwendungsvirtualisierung

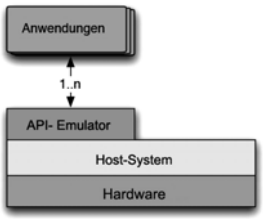
- Anwendungsvirtualisierung ist eine gute Möglichkeit Kompatibilitätsprobleme bei Betriebssystemmigrationen zu beheben
- Der gesamte Installationsumfang wird in einer Datei und die benutzerspezifischen Daten in einem separaten Verzeichnis gespeichert
- Gleichzeitiger Betrieb verschiedener Versionen einer Anwendung durch Isolation auf einem Betriebssystem möglich
- Nachteile: geringere Performance (Abstractionlayer), aufwendiger
- Beispiele: VMware ThinApp; MS Application Virtualization; Citrix XenApp

Emulation

Zwei Arten der Emulation:

Emulation einer API ←→ Hardware-Emulation

1. **Emulation einer API** (Application Programming Interface) → für Anwendungsprogramme, unter einer artfremden Architektur zum Einsatz kommen

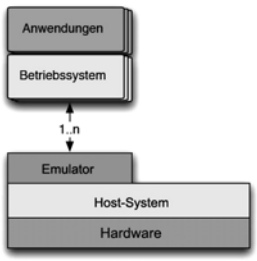


- Das Verfahren der Emulation einer API ist sehr performant, da nur Systemrufe übersetzt werden und eventuell erforderliche Bibliotheken verfügbar sind.
- Nachteil: eine API-Übersetzung ist für jede Kombination von Host- und Gastsystemen erforderlich
- Beispiel: WINE (Win32 API unter Linux)

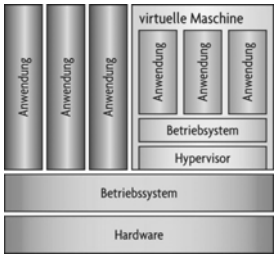
Emulation

2. **Hardware-Emulation** (Emulation eines kompletten Systems)

- Lediglich Ein- und Ausgaben greifen auf die Ressourcen des Host Systems zurück
- große Performance-Einbußen, aber sehr flexibel
- entspr. Virtualisierung
- Beispiel: Java

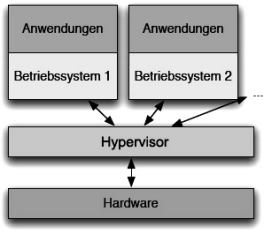


Host-gestützte Virtualisierung



- Eine host-gestützte Virtualisierung (Gast-System) läuft als Anwendung unter einem Host-Betriebssystem
- Damit ist das virtuelle System von der Funktion des Wirts abhängig.
- das Host-System kontrolliert sämtliche Ressourcen
- Vorteil: diese Art der Virtualisierung ist leicht zu installieren und die Virtualisierungssoftware ist größtenteils kostenlos.
- Beispiele: VMWare Workstation, Sun VirtualBox, Microsoft VirtualPC

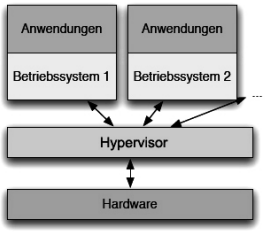
Vollvirtualisierung



- Bei der Vollvirtualisierung, auch Bare Metal-Virtualisierung oder native Virtualisierung genannt, läuft der Hypervisor direkt auf der Hardware.
- Keine Modifikationen an bestehenden Betriebssystemen erforderlich
- Sämtliche virtuellen Maschinen laufen völlig unabhängig voneinander. Es ist nicht erkennbar, dass sie nicht direkt auf der Hardware, sondern in einer virtuellen Maschine laufen.
- Keine der virtuellen Maschinen kann die gesamte Rechenzeit okkupieren und dadurch die anderen virtuellen Maschinen an der Ausführung hindern.
- Die gute Performance resultiert daraus, dass fast alle Befehle direkt auf der Hardware ausgeführt werden. Verluste entstehen nur durch den Kontextwechsel zwischen den virtuellen Maschinen.

Vollvirtualisierung

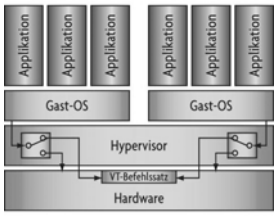
- Die x86-Architektur weist allerdings 17 kritische Befehle auf, die nicht virtualisierbar sind. Diese Befehle fängt der Hypervisor ab und emuliert sie.
- Bei einer neuen Hardware-Plattform müssen Modifikationen am Hypervisor durchgeführt werden



- Ein signifikanter Nachteil der Vollvirtualisierung ist die zu erwartende Performance-Einbuße, die sich durch die Erkennung, Auswertung von Instruktionen sowie dem Generieren und Einfügen von Emulationscode zur Laufzeit ergibt
- Beispiele: VMware ESX-Server, LynuxWorks Secure Hypervisor, Virtuallogix VLX

Virtualisierung auf Hardware-Ebene

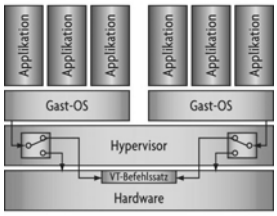
- Die Virtualisierung von Prozessoren bildet den wichtigsten Bestandteil der Hardware-Virtualisierung
- Eine VT-x (Vanderpool) Befehlssatzerweiterung (Intel), bzw. Pacifica (AMD) bewirkt, dass diese Befehle nicht vom Hypervisor emuliert werden müssen, dadurch wird die Programmierung des Hypervisors einfacher



- Die privilegierten Befehle müssen in der CPU im Nutzermodus abgefangen und im Supervisormodus bearbeitet werden
- die Hardware-Erweiterungen beschleunigen die Umschaltung zwischen den virtuellen Maschinen
- Vorteil: immense Effizienzsteigerung der Hardware-Virtualisierung auf Prozessorebene gegenüber der Software-Virtualisierung

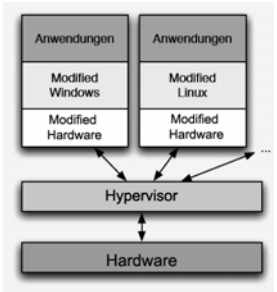
Virtualisierung auf Hardware-Ebene

- die Betriebssysteme befinden sich im Supervisormodus und isoliert voneinander
- Nachteil: die Hypervisoren sind hardwareabhängig, weil AMD- und Intel-Befehle nicht binärkompatibel sind
- Beispiel: Xen für Intel VT/AMD Pacifica



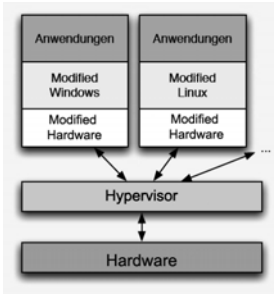
Paravirtualisierung

- Bei der Paravirtualisierung (para = parallel) wird das Gast-Betriebssystem so modifiziert, dass es im user mode ablaufen und den zugehörigen Emulations-Code des Hypervisors aufrufen kann
- Die Hardware wird weder virtualisiert, noch emuliert
- Über eine abstrakte Verwaltungsschicht (VMI) fängt das Gast-Betriebssystem kritische Befehle, die nur im Kernel mode ausgeführt werden können, selbst ab
- Dazu werden bestimmte Teile des Hypervisors in den Adressraum des Gast-Betriebssystems verlagert
- Die Programmierung des Hypervisors wird dadurch einfacher und der performance beanspruchende Wechsel von Gast zum Hypervisor wird reduziert



Paravirtualisierung

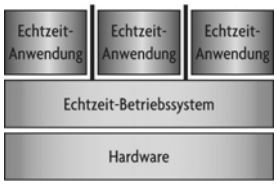
- **Vorteil:** sehr performante Abarbeitung der virtuellen Maschinen
- **Nachteil:** Vom Gast-Betriebssystem muss der Quellcode verfügbar sein, damit die Modifikation erfolgen kann. Mit jeder neuen Version des Gast-Betriebssystems muss eine neue Anpassung erstellt werden. (sehr zeit- und kostenintensiv)



- **Beispiele:** VMware ESX 3.5; Parallels Workstation; Red Hat Fedora Core 5 mit Xen 3.0; SUSE Linux Enterprise Server mit Xen; UserModeLinux; Citrix XenServer; KVM; IBM z/VM; Denali

Partitionierung

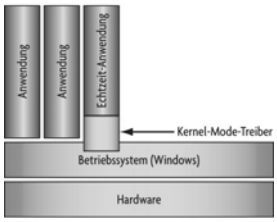
- Bei der Partitionierung werden Ressourcen, wie Speicherbereiche, Rechenzeit, I/O-Ressourcen usw., in kleinere Anzahl von Systemen des gleichen Typs aufgeteilt und streng voneinander getrennt



- Wie bei der Virtualisierung laufen mehrere unterschiedliche Betriebssysteme oder Instanzen desselben Betriebssystems parallel nebeneinander
- Im Unterschied zur Virtualisierung müssen Peripherieeinheiten einer bestimmten Partition zugewiesen werden. Dadurch können die Betriebssysteme direkt auf die ihnen exklusiv zugewiesenen Ressourcen zugreifen.
- Beispiele: Kuka/Acontis VxWin/QWin/CeWin, Tenasys InTime, Green Hills Padded Cell

Windows-Echtzeit-Erweiterungen

- Echtzeit-Erweiterungen stellen für den Anwender eine relativ einfache Möglichkeit dar, einzelnen Anwendungen in einem Windows-System zu deterministischem Verhalten zu verhelfen.
- Um Windows echtzeitfähig zu machen, greift ein Kernel-Mode-Treiber direkt auf die Hardware zu z.B. der Grafiktreiber
- Beispiele: Kithara Real-Time Suite, IntervalZero RTX (früher: Verturcom/Ardence), Sybera Real-Time Studio; für Linux: z.B. Wind River RT-Linux (früher: FSM-Linux)

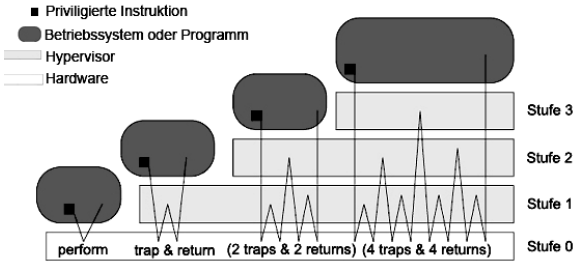


Rekursive Virtualisierung

- Die rekursive Virtualisierung beschreibt das Abbilden einer virtuellen Maschine auf sich selbst.
- Die Rekursion einer Maschine kann auch als Virtualisierung einer virtuellen Maschine aufgefasst werden.
- Die Abbildung der Systemressourcen sollte so trivial wie möglich konzipiert sein, um eine effiziente Ausführung bei n-facher Abbildung von virtuellen Maschinen zu gewährleisten
- In einer konventionellen virtuellen Maschine gibt es nur Vater- und Sohnprozesse, die miteinander kommunizieren können.
- Im Unterschied dazu ist bei der rekursiven Virtualisierung die Möglichkeit der Kommunikation zwischen Großvater- und Enkelprozessen gegeben, indem in jeder Stufe der rekursiven Virtualisierung die vollständige Umgebung einer übergeordneten Stufe simuliert wird.

Rekursive Virtualisierung

- Die Verringerung der Performance stellt sich in jeder Stufe der Rekursion ein, da sämtliche privilegierten Befehle aus der höheren Schicht empfangen und gleichzeitig emuliert werden.



- Dieser Slowdown steigt exponentiell zur Anzahl der verwendeten Stufen an. So muss z.B. auf der n-ten Stufe der Virtualisierung der Befehl 2^n-1 mal iteriert werden.
- Beispiel: von der Universität Utah entwickeltes Betriebssystem „Flux“

Hardware-basierende Vorteile

- Plattformunabhängigkeit, Hardware-unabhängiges Testen von Betriebssystemen möglich
- effizientere Ausnutzung der vorhandenen Ressourcen
- Kosten können minimiert und Hardwareressourcen geschont werden (durch geringeren Einsatz von Hardware)
- geringere Ausfallquoten
- bessere Administration der existierenden Systeme durch zentrales Monitoring erreichbar
- Geringere Wartungskosten
- Bei entsprechender Überwachung der virtualisierten Systeme können Kapazitätsengpässe vermieden werden.
- Einsatz von noch nicht entwickelter Hardware möglich
- Platzeinsparung

Hardware-basierende Vorteile

- Dienste auf den Gast-System arbeiten unabhängig von allen Prozessen auf dem Hosts-System
- Der Wirt stellt dem Gast ein normierte Hardware zur Verfügung (Gast muss sich keine Gedanken zu den Treiber machen)
- Ein Installation auf einer physischen Hardware kann jederzeit als VM-Image herangezogen werden
- „Hardware“ kann beliebig getauscht werden
- Hardware-Konsolidierung bringt Kosteneinsparung

Software-basierende Vorteile

- zusätzliche und unterschiedliche Betriebssysteme können auf dem selben Computer zur gleichen Zeit laufen
- Reduktion von Entwicklungs- und Testiterationen
- Momentaufnahmen sowie die Realisierung von Backups sind zu jeder Zeit möglich; es entfällt deren ständige Neuaufsetzung (Snapshot)
- Gegenüber nativen Systemen ist eine höhere Automatisierbarkeit und wesentlich schnellere Datensicherung durchführbar
- mannigfaltige Systeme können migriert werden. Es sind reale auf virtualisierte Maschinen überführbar und angelegte Images können in virtualisierte Systeme migriert werden
- abgekapseltes System (Viren, unsichere Systeme)
- Verwendung von Legacy-Applikationen (älterer Software) möglich

Software-basierende Vorteile

- Es besteht die Möglichkeit, die virtualisierten Systeme in einer einzigen dynamischen Datenstruktur abzulegen. Dadurch besitzen die Maschinen den Vorteil der Portabilität
- Das Klonen von virtuellen Maschinen ermöglicht ein Konzipieren von kompletten Rechnerlandschaften; verschiedenste Bedingungen sind gefahrenfrei simulierbar
- Man kann durch die Bereitstellung von eigenen Laufzeitumgebungen verhindern, dass Programme sich gegenseitig beeinflussen. (Workload Isolation)

Hardware-basierende Nachteile

- Die Performance eines virtualisierten Systems ist bei Verwendung einer Virtualisierungsschicht essentiell geringer als die auf einer realen Maschine.
- Der Energieverbrauch der Virtualisierungslösungen ist zum derzeitigen Entwicklungstand exorbitant hoch. Die Parzellierung der Energie in den virtualisierten Betriebssystemen ist nur ungenügend.
- In einem System, in dem viele I/O-Anforderungen erzeugt werden, ist die Architektur einer „Hosted-Virtuellen-Maschine“ ungeeignet

Software-basierende Nachteile

- die Lizenzproblematik ist noch ungeklärt, wie ist eine Software bei Verwendung von mehreren/Multicore- Prozessoren zu lizenzieren
- Bei der Architektur einer „Hosted-Virtuellen-Maschine“ kann die virtuelle Maschine einem schlechten Scheduling des Host-Betriebssystems ausgesetzt sein
- Weiterhin kann die Verwendung von zu vielen konzipierten virtuellen Maschinen zu einer Erhöhung der Anzahl der zu überwachenden Systeme führen
- Beim Accounting müssen neue Verfahren zur Messung verursachungs-gerechter Zurechnungen eingesetzt werden

Kommerzielle Virtualisierungslösungen

- Microsoft Virtual PC [Win → Win , DOS, OS/2]
- Microsoft Virtual PC Mac [OSX → Win, OS/2, Linux]
- Microsoft Hyper-V [Win → Win]
- Virtuozzo [Win, Linux → Win, Linux]
- VMware
- Parallels
- Citrix Xen Server
- SandBoxIE
- PikeOS für eingebettete Systeme

Freie Virtualisierungslösungen

- Xen [Netbsd, Linux → Linux, FreeBSD, (Win), …]
- Sun VirtualBox [Win, Linux → Win, Linux]
- KVM [Linux → Win, Linux]
- Uml [Linux → Linux]
- Linux-VServer [Linux → Linux]
- BSD Jails [FreeBSD → FreeBSD]
- lguest
- OpenVZ
- Proxmox VE