Basistechnologie: Virtualisierung

Ein Vortrag zum Cloudseminar im WS09/10

Maximilian Hoecker

Fakultät für Informatik,
Hochschule Mannheim,
Paul-Wittsack-Straße 10,
68163 Mannheim
maximilian.hoecker@stud.hs-mannheim.de

20.10.2009

Agenda

Agenda

Eine wage Begriffsdefinition

Definition von Virtualisierung

Virtualization is a framework or methodology of dividing the resources of a computer into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others.

Eine wage Begriffsdefinition

. . . und nochmal auf Deutsch

Virtualisierung ist. . .

- ein Framework bzw. Methodik
- zur Aufteilung von Computerresourcen in viele Ausführungsumgebungen z.B.
  - Hardware-/Softwarepartitioning
  - Timesharing
  - Teil- oder Vollsimulierung von Maschinen
  - Emulation
  - Quality of Service
  - uvm.

Aber

Die Definition sollte man nicht zu genau nehmen

- Virtualisierung bedeutet nicht immer **Aufteilen** von Resourcen
- Man kann z.B. X Hardwarefestplatten zu einer logischen Partition **zusammenfassen** durch einen Virtualizationlayer
- Grid Computing virtualisiert durch das Nutzen verschiedener verteilter Resourcen

Agenda

Historisches

- Idee der Virtualisierung ist schon 50 Jahre alt
- *Christopher Strachey* veröffentlichte 1959 *Time Sharing in Large Fast Computers*
- Am MIT in den 1950er Jahren:
  - Konzept zur gleichzeitigen Bedienung von Rechnern
  - Ohne Leistungseinbußen
  - Name: Time Stealing
- Damit waren die ersten Konzepte zur CPU-Virtualisierung geboren.

Die erste Abstraktion

- Erster Großrechner *Atlas* (1962) mit Stracheys Time Sharing
- Neuer Architekturansatz im Betriebssystem
- Dort wurde separiert zwischen:
  - Prozessen des Betriebssystems (**Supervisor**)
  - Darunter **eine Komponente** die zuständig war für ausgeführte Programme
- Ebenfalls neu waren *Virtuelle Speicher/One Level Store* und *Paging*

Schritt zur ersten Emulation

- Mit der Veröffentlichung von *IBM OS/360* (1967) hatte IBM ein Problem
- Die Programme die aktuell beim Kunden liefen konnten nicht auf der neuen Architektur ausgeführt werden
- Zur Erleichterung des Umstiegs von den alten *IBM 7070* Rechnern wurde im OS/360 eine Erweiterung eingebaut
- Die Erweiterung übersetzte die 7070er Befehle in S/360 Befehle
- Der Erfinder der Erweiterung taufte diese "Emulator"
- Die Emulation der 7070 Maschine beanspruchte viel Rechenleistung (bis zu 25% Overhead)

Server Partitioning

- Bis heute wurde die Architektur der S/360 beibehalten und erweitert
- Aktuelle Systeme: IBM z10 Business/Enterpise 1-64 CPUs 64GB-1,5TB RAM
- Diese Systeme haben 2 verschiedene Modi mit denen gearbeitet werden kann
  - LPAR Mode (**L**ogical **Par**tition Mode)
  - VM Mode (**V**irtual **M**achine Mode)
- Diese Modi können auch gemischt werden, also auf einer Partition mehrere VMs

Zusammenfassung Hardwarevirtualisierung

- Lässt man CPU-Scheduling als Virtualisierung außen vor gibt es 3 Virtualisierungsmöglichkeiten von Hardware:
  - Domaining/Partitioning z.B. mit LPAR/PRISM von IBM in dem z-Systemen (ganz nativ ohne SW-Zusätze im Betriebssystem)
  - Virtualierung von Hardware mit Hypervisorn (nativ, aber mit Hypervisor(später mehr dazu))
  - Emulation von Hardware (pure SW-Lösung)

Der Unterschied zwischen Virtualisierung und Emulation von Hardware

- Bei der Virtualisierung von Hardware darf die abgebildete Architektur in der VM nicht abweichen von der des Host-Systems.
- Bei der Emulation ist dies nicht nötig, da jeder Aufruf von in die darunter liegende Architektur, von einen Programm übersetzt wird

Zusammenfassung Hardwarevirtualisierung

Vorteile von Hardware-Virtualisierung gegenüber Emulation

- Der Zugriff auf die Hardware kann direkt an die phy. Addresse durchgereicht werden
- durch das fehlende Übersetzen hat man weniger Overhead
- und dadurch ist ein solches System schneller als ein Emuliertes
- man hat dadurch auch weniger Kompatibilitätsprobleme

Vorteile von Emulation gegenüber Virtualisierung

- Man kann auf einer Hardware viele verschiedene andere Hardware emulieren (zur Konsolidierung)
- Falls ein Emulator existiert, kann die darunter liegende Hardware getauscht werden

Agenda

WABI

WABI

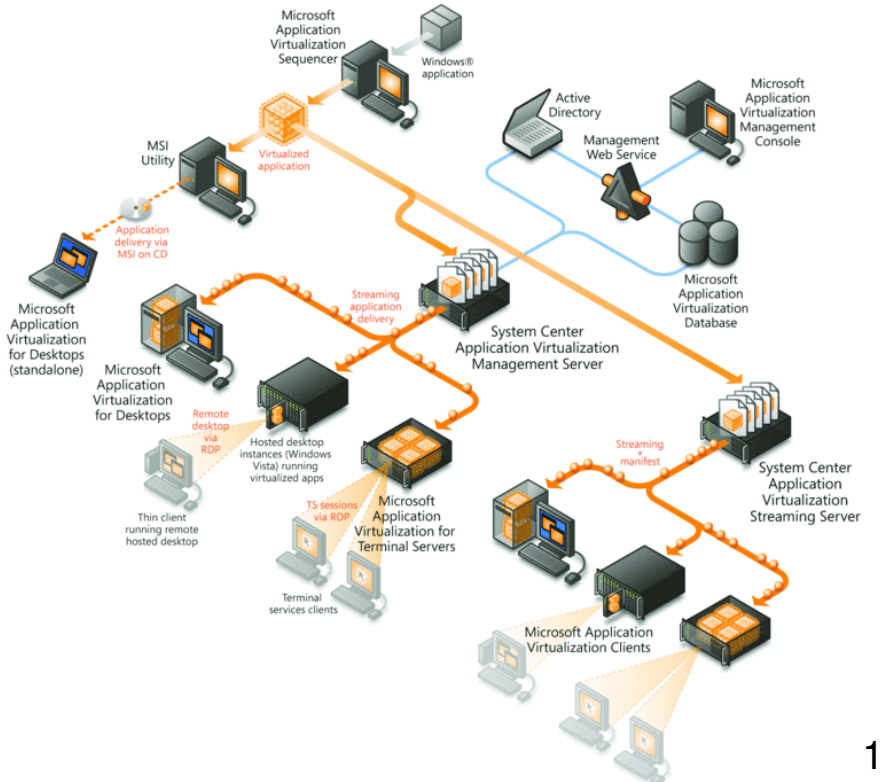
- Sun veröffentlichte 1993 das **W**indows **A**pplication **B**inary **I**nterface
- Windows Programme in Solaris (unverändert)
- Sowohl auf SPARC als auch auf x86-Architektur

WINE

WINE

- WINE Projekt startet 1993 (**W**ine **I**s **N**ot an **E**mulator)
- Keine Emulatorschicht
- x86 Architektur ist Vorraussetzung
- Aufrufe werden nur zu den entsprechenden Linux-Kernel Funktionen umgeleitet

Microsoft App-V



[1]

- Bei App-V werden Anwendungen in Packages virtualisiert und lokal auf den Clients/Terminal-Servern ausgeführt

[1] Quelle: $http://i.technet.microsoft.com/cc904189.fig01_L.gif$

Virtualisierung von Peripherie-Geräten

- Bei der Peripherie-Geräten wird im Betriebssystem ein Treiber/Modul geladen
- Dieses Modul simuliert ein Gerät, das am Rechner angeschlossen ist
- Bespiele:
  - Virtuelle CD-Laufwerke (Daemon Tools, Virtual CD)
  - Drucker Spooler
  - RAID/LVM
  - iSCSI
  - rCAPI
  - usw.

Virtualisierung auf Betriebssystemebene

Siehe nächstes Kapitel

Agenda

Allgemeine Überlegung

- Ein Betriebssystem versucht immer auf Hardware (egal ob virtualisiert oder nicht) zuzugreifen.
- Virtualisiert man also ein Betriebssystem, muss man dafür sorgen, dass das Betriebssystem denkt, es greift direkt auf Hardware zu.
- Es versucht ebenfalls immer auf die gleiche Weise auf Peripheriegeräte zuzugreifen.

Grundlagen der x86 Architektur

- Der x86 Prozessor hat ein Ringschutzkonzept
- Es gibt 4 Ringe, der innerste ist Ring 0, der äußerste Ring 3
- Je innerer der Ring desto mehr Privilegien hat der Prozess auf Hardware zuzugreifen
- In Ring 0 (Kernel-Space) läuft meistens der Kernel/Module/Treiber eines Betriebssystems
- In Ring 3 (User-Space) die Anwendungen
- Die meisten Betriebssysteme benutzen nur 2 Ringe (0 und 3)

- Ein Prozess kann zu einem Zeitpunkt nur auf einem Ring laufen
- Ein Prozess kann nicht selbst "den Ring wechseln"
- Möchte ein Prozess eine Operation aufrufen, die in einem inneren Ring liegt, erzeugt der Prozessor eine Exception, die behandelt werden muss
- Wird diese nicht behandelt: Schutzverletzung, Prozessabsturz

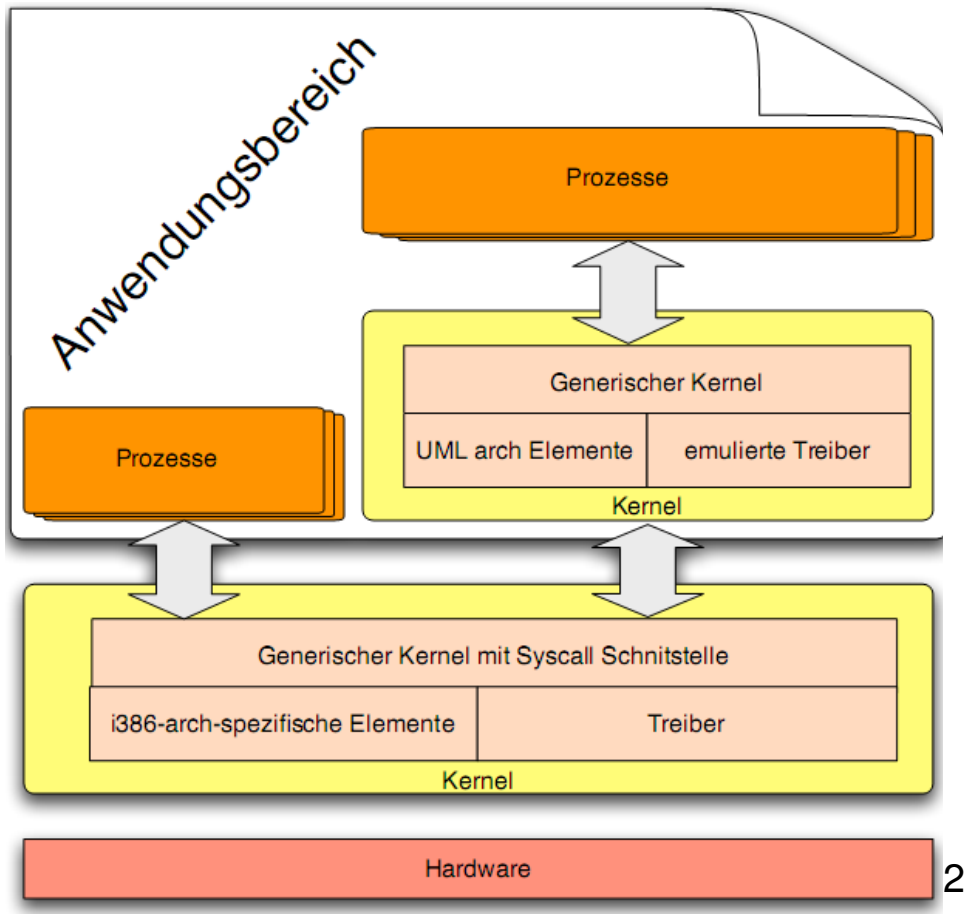
Agenda

Der Anfang von Betriebssystemvirtualisierung

User Mode Linux

- 1999 veröffentlichte Jeff Dike einen Patch mit dem sich der Linux-Kernel sich selbst als unprivilegierten Prozess aufrufen kann
- Folge: man konnte mehrere Instanzen des Betriebssystems auf dem selben Rechner fahren
- Implementierung (User Mode) wurde mit der arch-Schnittstelle im architekturunabhängigen Teil des Linux Kernels umgesetzt
- Startet man den Kernel also im User Mode, werden alle hardwareunabhängigen Aufrufe aus der VM vom Host Kernel angenommen und bedarfsweise an den Gast-Kernel durchgereicht (gleich mehr dazu)

Gesamtarchitektur eines UML Systems



[2]

[2] Quelle:
http://www.lrr.in.tum.de/ stodden/teaching/sem/virt/ss06/doc/virt06 – 07 – 20060531 – kern – doc%20 – %20Paravirtualisierung.pdf

Funktionsweise der Architektur

UML im TT-Mode

- Für jeden Gast betreibt der UML-Host einen Tracing Thread
- Für jeden Prozess eines Gasts wird ein Host Prozess angelegt
- d.h. jeder Prozess läuft im Prinzip auf dem Host Kernel (nur unterschieden durch den Tracing Thread)

Aufgabe des Tracing Threads

- Setzt ein Gast ein Signal ab, wird dieses für den Hostkernel, falls es ein HW-unabhängiges Signal ist, immer in einen *getpid()* Call umgewandelt/"nullifiziert"
- Der gleiche Call wird aber unmodifiziert an den Gastkernel weitergereicht.

Hää??Wieso mit getpid() überschrieben?

- Die Theorie eines solchen Systems ist oft ideell
- In der Realität existiert ein UML-Debug-Handler im Host Kernel
- Die Debugschnittstelle wurde extra für UML in den Kernel eingepflegt
- Dieser Handler springt vor jedem Systemcall an und greift diesen Call auf
- Interessanterweise gibts aber keine Möglichkeit den Handler zu deaktivieren
- Es gibt auch keine Möglichkeit den Systemaufruf im Hostkernel abzubrechen oder zu löschen
- Deswegen wird dieser Call mit einem *getpid()* Call überschrieben

Vorteile und Probleme von UML

Vorteile im Allgemeinen

- VMs laufen fast ohne Overhead (Im Vergleich zu emulierten Systemen)
- Keine eigenen Treiber auf der VM benötigt
- Keine Modifikation des Gast- oder Hostsystems nötig (da UM-Kernel wie ein normaler Prozess behandelt wird)

Probleme im TT-Mode

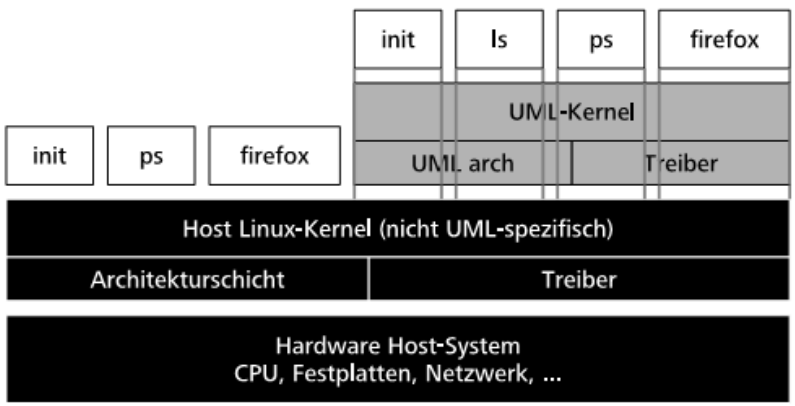
- Prozessverwaltung ist mit Host gekoppelt:
  - Für jeden Prozess im Gast wird ein entsprechender Prozess im Host angelegt. (Prozesskopplung)
  - Folge bei vielen Gästen und vielen Prozessen: Performanceprobleme
- Speichervirtualisierung kaum umgesetzt: Einem Gast ist es möglich den Kernelspeicher zu überschreiben und aus dem Gast-Speicherbereich auszubrechen
- Kontextwechsel im Gastsystem können zu Leistungseinbußen führen, da jeder Wechsel explizit genehmigt werden muss vom Hostkernel (4 Stück: Syscall <-> TracingThread <-> Hostkernel) [a]

[a] Quelle: *http : //www.uni − koblenz.de/ vnuml/docs/vnuml/uml.pdf*

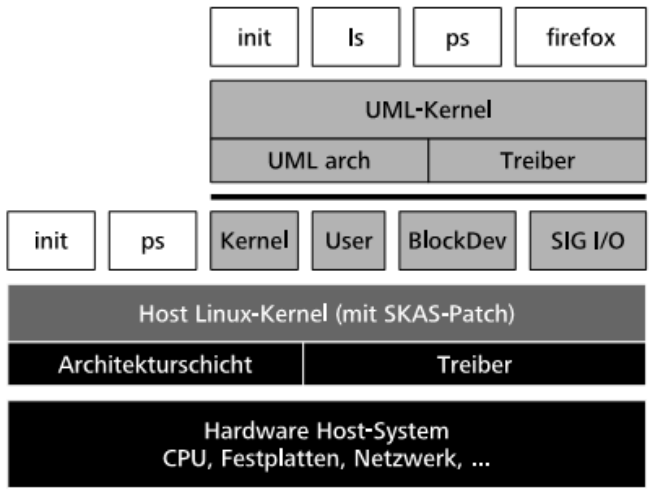
Problemlösung mit dem SKAS Patch

Seperate **K**ernel **A**ddress **S**pace Eigenschaften:

- Wird auf dem Kernel des Hosts implementiert
- Löst die Probleme mit Prozessverwaltung, fehlende Speichervirtualisierung und auch das Kontextwechselproblem



Prozessarchitektur eines UML-Systems
Prozesse im UML-Bereich sind für den Host-Kernel sichtbar



Prozessarchitektur eines UML-Systems mit SKAS-Patch
UML-Bereich wird hinter vier Prozessen verborgen:
Kernel- und User-Bereich, Blockgerätetreiber und
Signalvermittlung für Ein-/Ausgabe

[3] Quelle: *http://markus-gerstel.de/files/2005-Xen.pdf*

Folgen durch den SKAS Patch

- Durch die vorhandene Speichervirtualisierung wird pro Gast ein seperater Kernel-Datenspeicher verwendet
- Der Host-Kernel Speicher wird durch das nun vorhandene Memory Management geschützt
- Durch eine modifizierte *ptrace()* Funktion (Prozesstracing) im Kernel werden nun nur noch 2 Kontextswitches benötigt
- Insgesamt ist das System ca. doppelt so schnell (gemessen bei Kernel-Compiling) [4]
- Es gibt nun nur noch 4 Prozesse auf dem Host System für alle UML Gäste
  - UML Kernel Thread - führt Code im UML Kernelspace aus
  - UML Userspace Thread - führt Code im UML Userspace aus
  - Asyncroner I/O Treiber Thread - virtuelle Block Devices
  - Emulator-Thread für WRITE-SIGIO - Hilfsprozess, der Signale für *write()* calls

[4] *http* : *//user − mode − linux.sourceforge.net/old/skas.html*

Agenda

Virtual Machine Monitor (VMM)

- VMware veröffentlichte 1999 ihre Software VMware Workstation
- Virtualisierung eines kompletten x86 Rechners auf einem anderen x86er
- Neu: Eigenes BIOS, Eigene Hardware (begrenzt)
- Auch Neu: **V**irtual **M**achine **M**onitor, eine Anwendung überhalb des Betriebssystems, dass die vorhandene Hardwareresourcen an die VMs durch Emulation verteilt

VMM - Scan-before-Execution / PreScan

- Der VMM überprüft parallel zur Laufzeit den Programmcode der VM **bevor** der ausgeführt wird
- wird dabei ein abzufangender Befehl erkannt, wird dort ein Breakpoint gesetzt und die Suche endet erstmal
- Gibt es konditionale Sprünge, werden beide verfolgt (bis maximale Suchtiefe erreicht)
- Falls etwas nicht gescannt werden konnte, wird der Rest der Sprünge emuliert
- Dieses Verfahren wird **S**can-**b**efore-**E**xecution genannt

Weitere wichtige Teile in der VMware Architektur

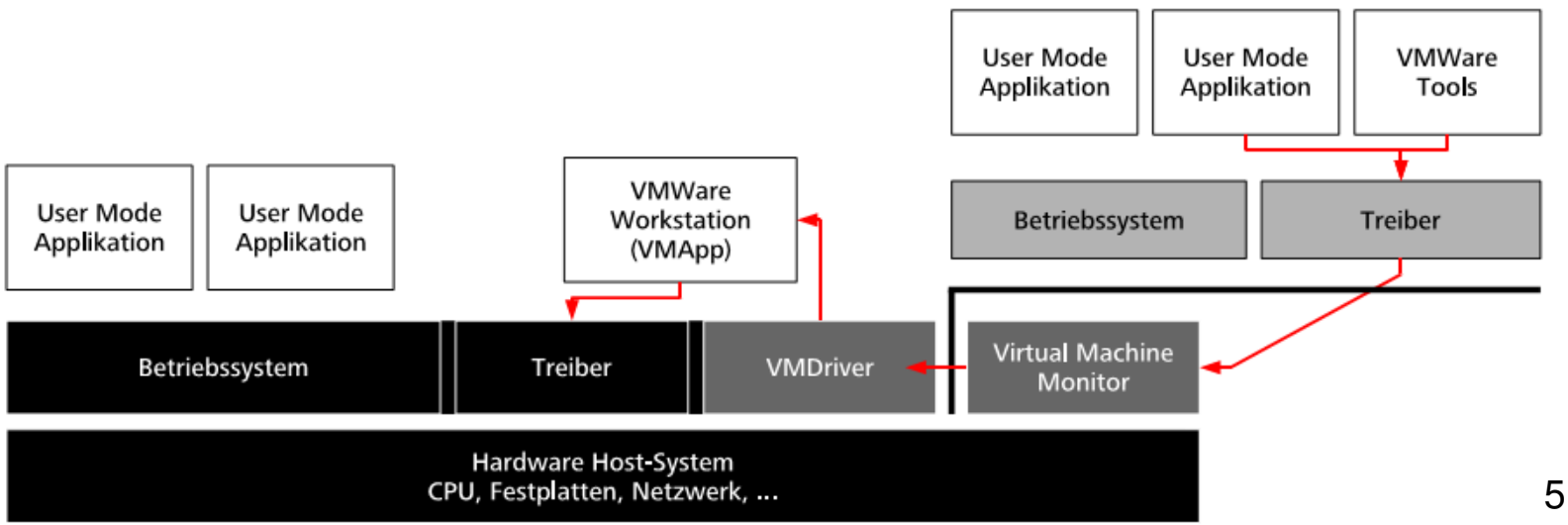
VMDriver / VMXDriver

- Modul im Hostsystem
- dient als Kommunikationsschnittstelle für I/O Verkehr und Gast <-> VMApp
- Speicherverwaltung
- Gerätemanagement

VMApp

- Oberfläche für den Endanwender
- Kommunikationsschnittstelle für Treiber
- Kommunikationsweg Hosttreiber <-> VMApp <-> VMXDriver

Architekturübersicht VMware Workstation



[5] Quelle: *http://wwwspies.informatik.tu-muenchen.de/lehre/seminare/WS0506/hauptsem/Ausarbeitung02.pdf*

Agenda

Para, was?

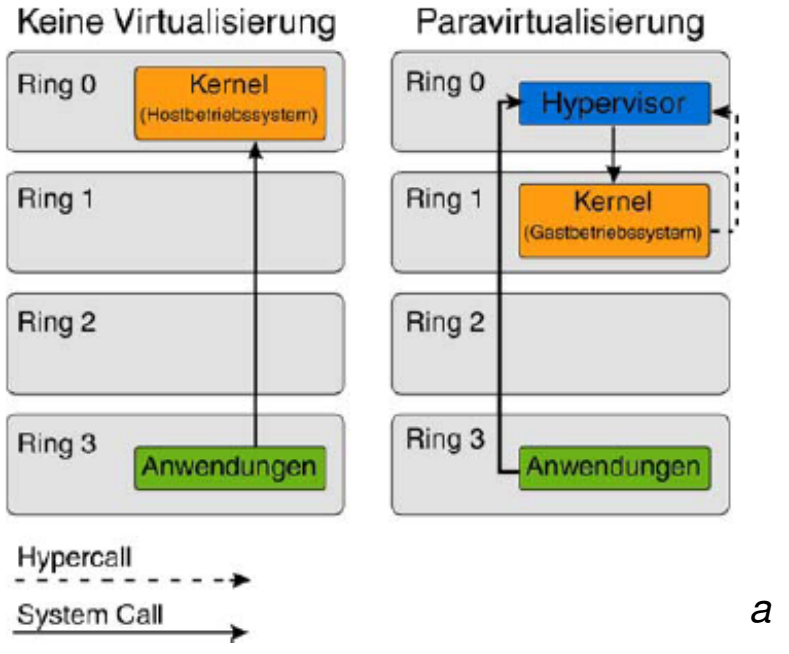
- Paravirtualisierung ist ein Verfahren zur effizienten Virtualisierung von x86 Architekturen mit einem Hypervisor und einem modifizierten Gast-Kernel
- Seit neustem auch modifizierte Kernel für Windows XP, 2003 und 2008 verfügbar

Hypervisor

- Ein Hypervisor ist im Grunde ein Virtual Machine Monitor
- Er besteht aus einem minimalen Betriebssystem, das im Ring 0 des CPUs arbeitet
- Die Aufgaben sind ebenfalls Speicherverwaltung, Scheduling, Geräteverwaltung
- Ein Hypervisor emuliert im Paravirtualisierungsmodus keine Resourcen, aber Geräte (Netzwerkkarte, Festplatten...)

Ringbelegung und Hypercalls

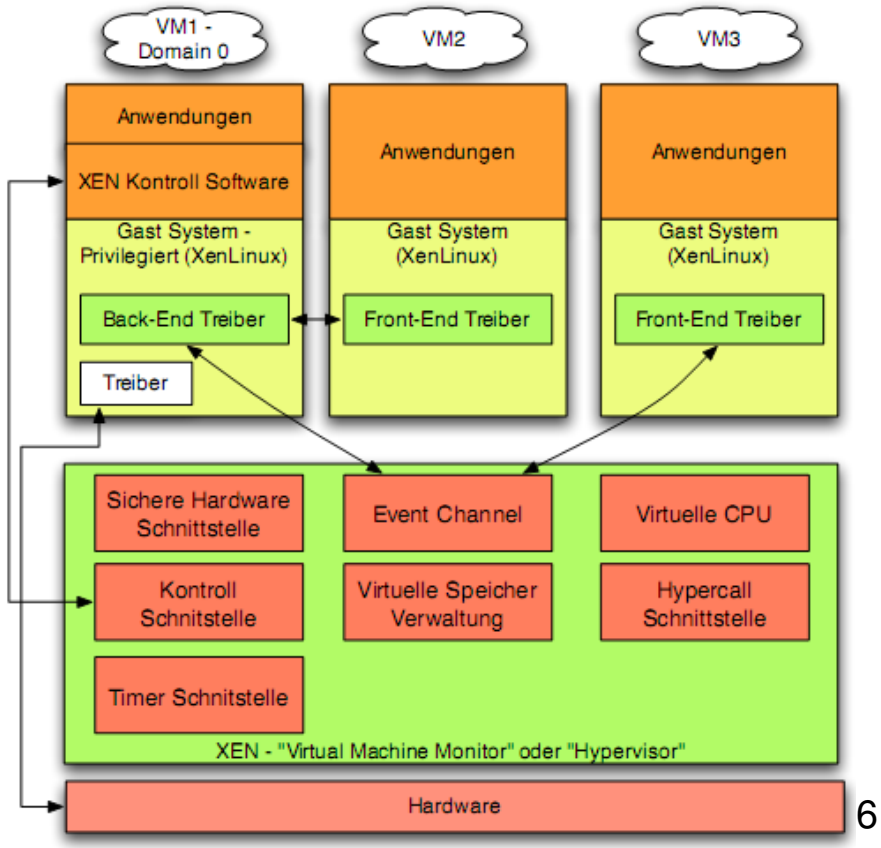
- Gastkernel weicht aus Ring 0 in Ring 1
- Problem: Gastkernel in Ring 1, will aber auf Hardware zugreifen!
- Lösung: Hypercalls



a

[a] Quelle:Informatik Spektrum 32 3/2009 S.205

XEN Architektur



[6]

---

[6] Quelle:
http://www.lrr.in.tum.de/ stodden/teaching/sem/virt/ss06/doc/virt06 – 07 – 20060531 – kern – doc%20 – %20Paravirtualisierung.pdf

Agenda

Unmodifizierte Gäste auf einem Hypervisor??

- Unterstützt ein Prozessor entweder die Intel **V**irtual **M**achine E**x**tenstion (VMX) oder den AMD **S**ecure **V**irtual **M**achine (SVM) Befehlssatz
  - 2 Modi des Prozessors möglich
  - VMX-Root-Modus (Hypervisor, Ring -1 am rechnen)
  - VMX-Non-Root-Modus (Gast-OS, Ring 0-3 am rechnen)

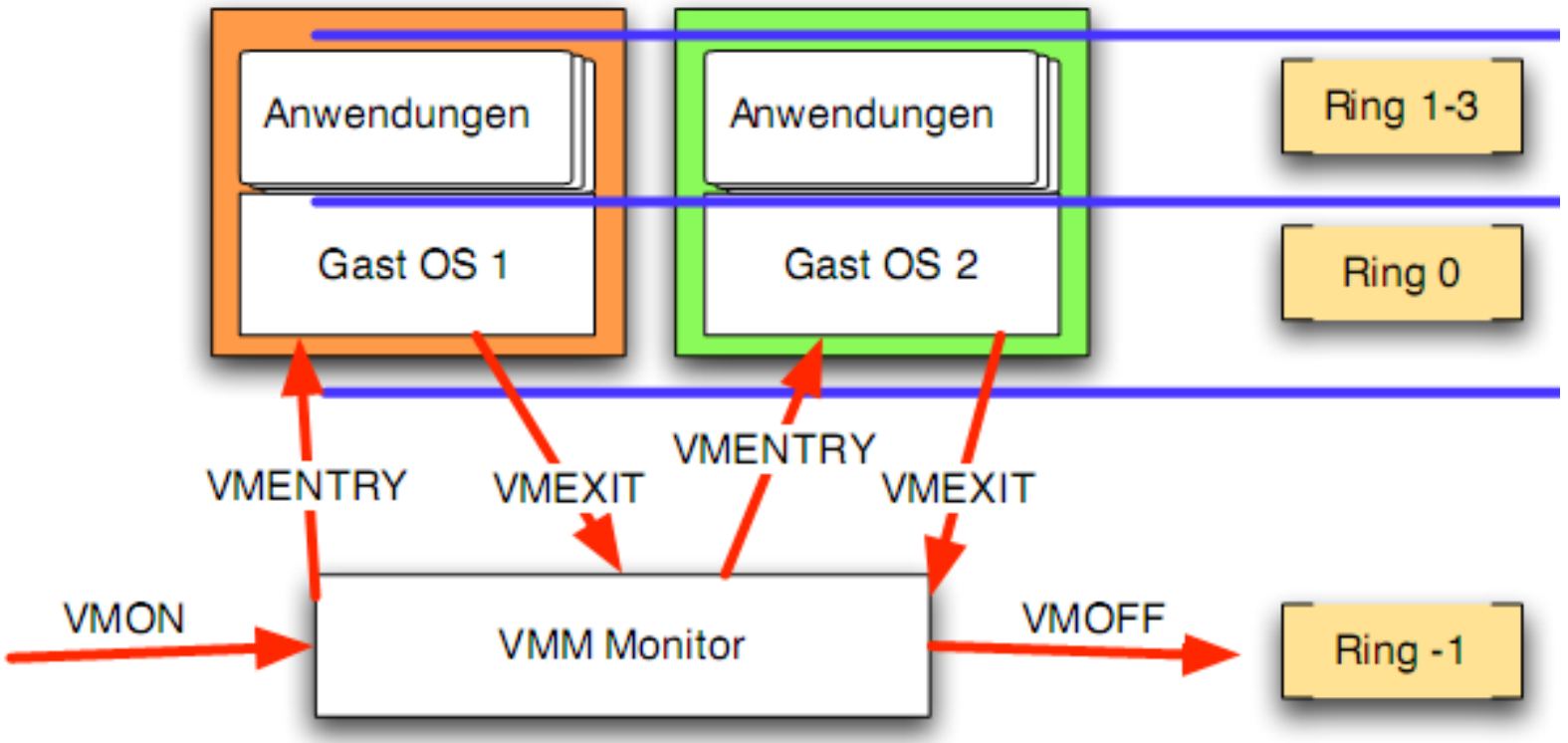
7

CPU Befehle bei VT bzw. SVM

- *VMON*: CPU in Virtualisierungsmodus schicken
- *VMENTRY* (von VMM aus): Übergabe der Ringe(0-3) an Gast
- *VMEXIT* (von VMM aus): Abgabe der Ringe an VMM
- *VMOFF*: CPU aus Virtualisierungsmodus holen

[7] Quelle
http : *//www.lrr.in.tum.de/ stodden/teaching/sem/virt/ss*06*/doc/virt*06 − 07 − 20060531 − *kern* − *doc*%20 − %20*Paravirtualisierung.pdf*

Ringaufbau und Zeitliches Beispiel



[8] Quelle:
http://www.lrr.in.tum.de/stodden/teaching/sem/virt/ss06/doc/virt06-07-20060531-kern-doc%20-%20Paravirtualisierung.pdf

Folgen von VMX/SVM

- Eine VM kann nicht mehr unterscheiden, ob sie nativ oder virtualisiert betrieben wird (Ring 0 steht immer zur Verfügung)
- Kein Performanceverlust, da keine Emulation von Hardware nötig ist

Unterschiede Vanderpool / Pacifica

- Technologien nicht kompatibel
- AMD's Pacifica virtualisiert ebenfalls den bei Intel hardwaretechnisch gelösten Speichercontroller
- Pacifica hat ebenfalls einen **D**evice **E**xclusion **V**ector (DEV) integriert, der es VMs ermöglicht auch Geräte zu benutzen, die ohne CPU auf Speicher zugreifen können.

Agenda

Relativer Performancevergleich aller Systeme zu Linux

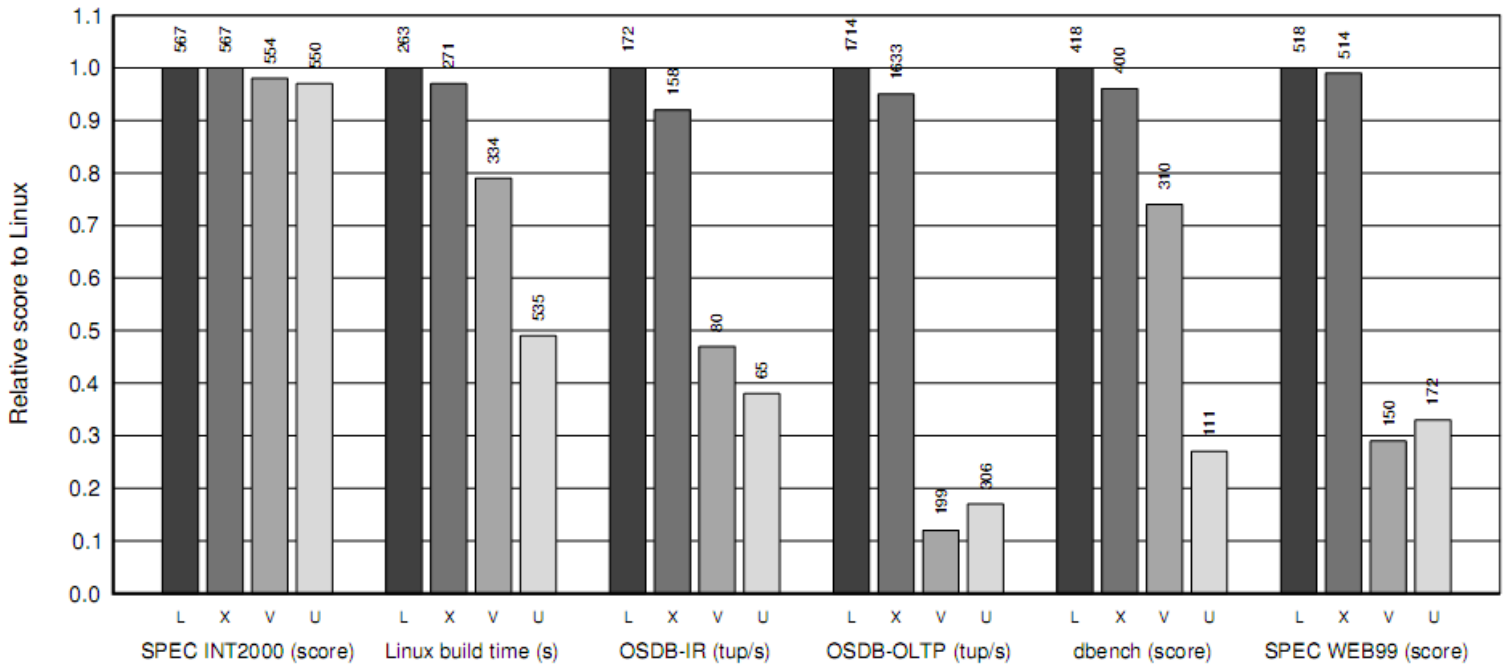


Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

[9] Quelle:
http://www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf

Fazit

- Virtualisierung eignet sich für viele Bereiche, bei der keine spezielle Hardware benötigt wird
- Virtualisierung birgt kaum Risiken oder Nachteile, wenn man sie richtig Einsetzt
- Administratoren brauchen fachübergreifendes Wissen z.B.:Netzwerk-Utils in Linux
- Große Probleme bekommt man bei allen Geräten, die nicht blockorientiert Arbeiten: ISDN Karten, USB Dongles für Lizenzen, alle A/V-USB-Geräte
- XEN ist inzwischen altbewährt, aber auf dem Rückmarsch. Neues Produkt ähnliche Technologie: Kernel Based Virtual Machine (KVM)

Gibt es Fragen