Betriebssysteme I

Sequentielle und eng gekoppelte parallele Systeme

Kapitel 8: Virtualisierung

Stand: WS 10/11 (25.01.11)

Prof. Dr. Wolfgang Küchlin

Dipl.-Inform., Dr. sc. techn. (ETH)

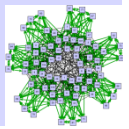
Arbeitsbereich Symbolisches Rechnen
Wilhelm-Schickard-Institut für Informatik
Fakultät für Informations- und Kognitionswissenschaften

Universität Tübingen

Steinbeis Transferzentrum
Objekt- und Internet-Technologien (OIT)

Wolfgang.Kuechlin@uni-tuebingen.de
http://www-sr.informatik.uni-tuebingen.de

SR

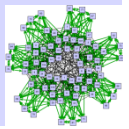


Abstraktionsschichten und Schnittstellen

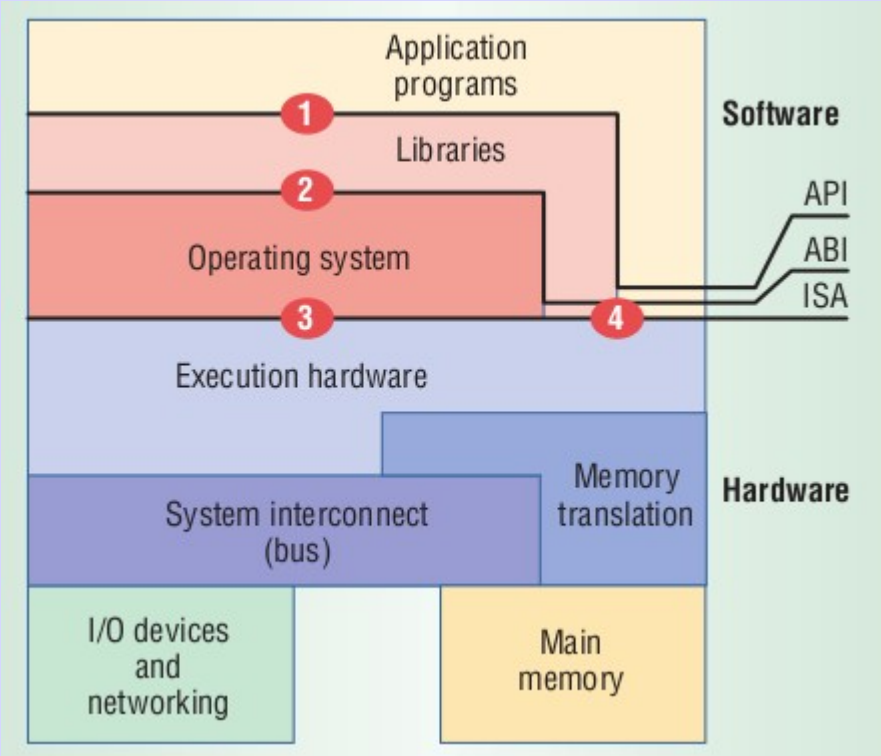
- Unterteilung von komplexen Systemen in Abstraktionsschichten
- Zwischen den Schichten gibt es eindeutig definierte Schnittstellen.

Vorteile:

- Einzelne Komponenten können räumlich und zeitlich getrennt von unterschiedlichen Teams entwickelt werden.
- Komplexität bleibt beherrschbar.



Beispiel: Architektur eines Computersystems

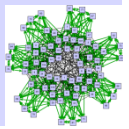


ISA – Instruction Set Architecture
(the raw machine as seen by a user or system binary)

ABI – Application Binary Interface
(the machine with OS as seen by a user process)

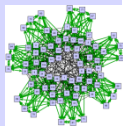
API – Application Programming Interface
(the machine with OS as seen by a high-level program)

- ISA: Maschinen-Instruktionssatz
  - (4) User-ISA
  - (3) System-ISA
    - System ist Obermenge von User
    - System-ISA enthält Instruktionen, die im Kernel-Mode verfügbar sind
- ABI: User ISA + System-Calls
  - (4) User-ISA
  - (2) System Call Interface
    - "rohe" SysCalls (trap + Parameter)
- API: User ISA + SysCall-Library
  - (4) User-ISA
  - (1) System Library Interface
    - system library (libc) wraps syscalls for portability



Problem von Schnittstellen

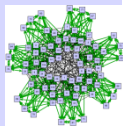
- Es gibt in der Praxis verschiedene Schnittstellen zwischen den Abstraktionsschichten.
  Beispiele:
  - verschiedene Prozessorarchitekturen –
    x86 und PowerPC haben inkompatible ISAs
  - verschiedene Betriebssysteme –
    Windows und Linux haben inkompatible ABIs/APIs
- Komponenten, die für eine Schnittstelle entwickelt wurden, arbeiten nicht mit Komponenten für eine andere Schnittstelle zusammen.
- eine Motivation für Virtualisierung



Virtualisierung (im weiteren Sinne)

Virtuelle Erzeugung eines Systems oder einer Komponente auf einem zugrundeliegenden realen System

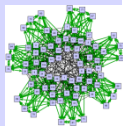
- Abbildung einer virtuellen Schnittstelle
  auf reale Schnittstelle
  durch Virtualisierungssoftware
- Die erzeugte virtuelle Schnittstelle wirkt wie eine oder mehrere Schnittstellen, die identisch oder verschieden zur realen Schnittstelle sein können.



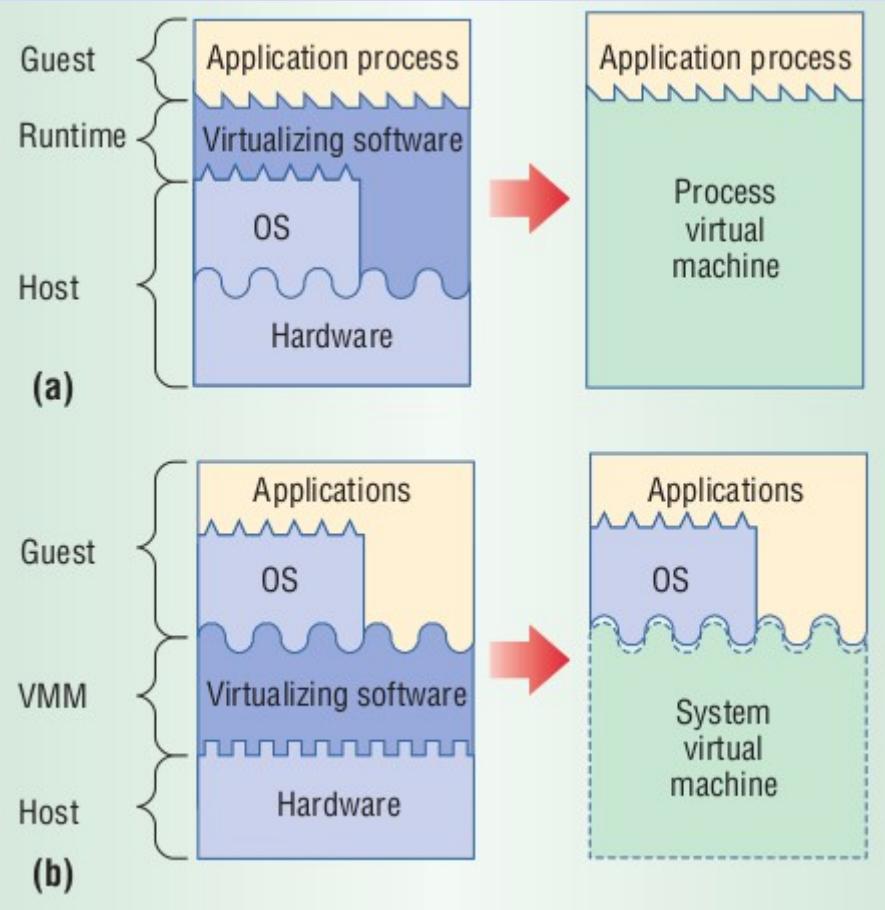
Grundidee der OS Virtualisierung (Reinform)

- Ersetze im Kernel-Mode das OS durch Virtualisierungssoftware (virtual machine monitor VMM)
- Führe das OS in User-Mode als Prozess aus („Guest OS“)
- Virtualisierungssoftware nimmt traps entgegen und ruft zugehörige Funktion des zugehörigen Guest OS

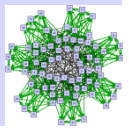
- Problem: Instruktionen können in User-Mode garnicht oder ganz anders funktionieren als in Kernel-Mode
  - Der OS code hat als Guest andere Semantik als im Kernel.



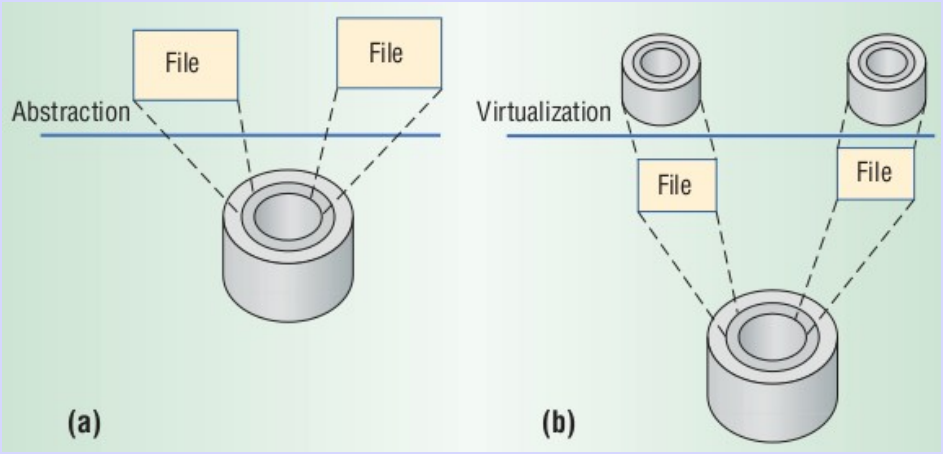
Beispiele zur Virtualisierung



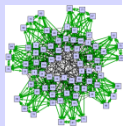
- (a) Process VM
  - virtual platform to support a single process (the *runtime* environment)
  - emulates user-ISA and/or syscalls or system library calls (ABI or API)
  - created and destroyed with process
- (b) System VM
  - complete persistent environment supporting a guest OS + processes
  - provides the guest OS with access to virtual hardware resources
  - may also emulate underlying ISA
    - *interpretation* or *dynamic binary translation*
  - the underlying hardware ist the *host*
  - the virtualization SW is the VMM



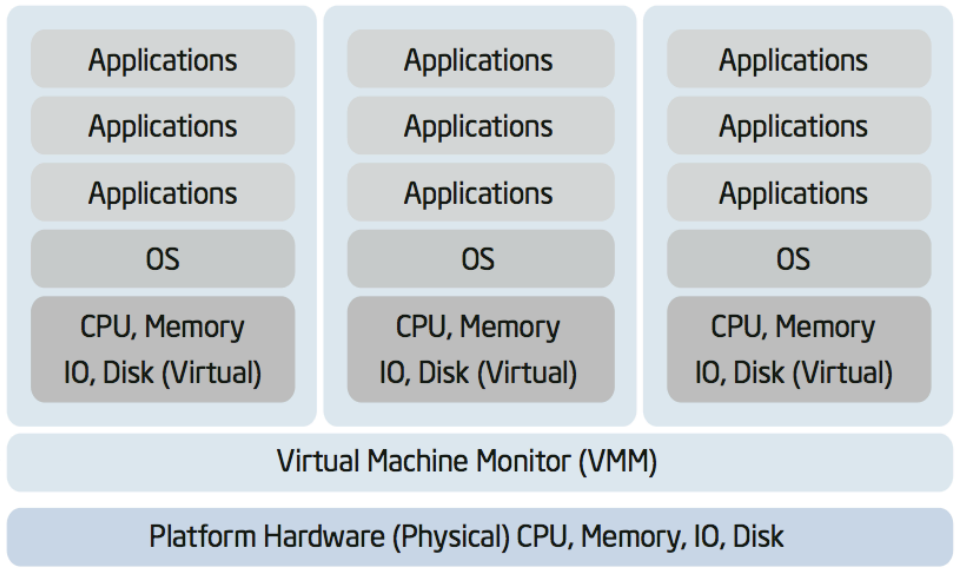
Virtualisierung vs. Abstraktion



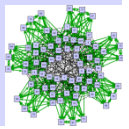
- Abstraktion erzeugt eine Schnittstelle auf einer höheren Abstraktionsebene
  - Bsp: File vs. Platte. File-Transfer abstrakter als Block-Transfer.
- Virtualisierung erzeugt eine oder mehrere Schnittstellen, die auf der selben oder tieferen Abstraktionsebene liegen.
  - Bsp: 2 virtuelle Platten auf einer physikalischen Platte realisiert
  - Block-Transfer zu 2 virt. Platten abgebildet auf 1 reale Platte (z.B. via Files)



System Virtualisierung



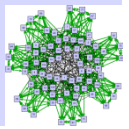
- Abstraktion der Hardware
- Entkopplung des Anwenders von der physikalisch vorhandenen Plattform
- Erreichen einer homogenen Platform mit transparenter Ressourcen-Verwaltung



Gründe für Virtualisierung

- **Support von Legacy-Programmen**
- **Erhöhung der Systemauslastung**
- Reduktion des administrativen Aufwands
  - Bündelung von Software mit OS möglich
  - keine Neu-Installation / keine Inkompatibilität bei OS upgrade
- Komplette Isolation von verschiedenen Nutzern / Diensten
- Verbesserung der Verfügbarkeit
  - OS Absturz betrifft nur Teilsystem

- auf Mainframes seit den 1970er Jahren



Gründe für Virtualisierung

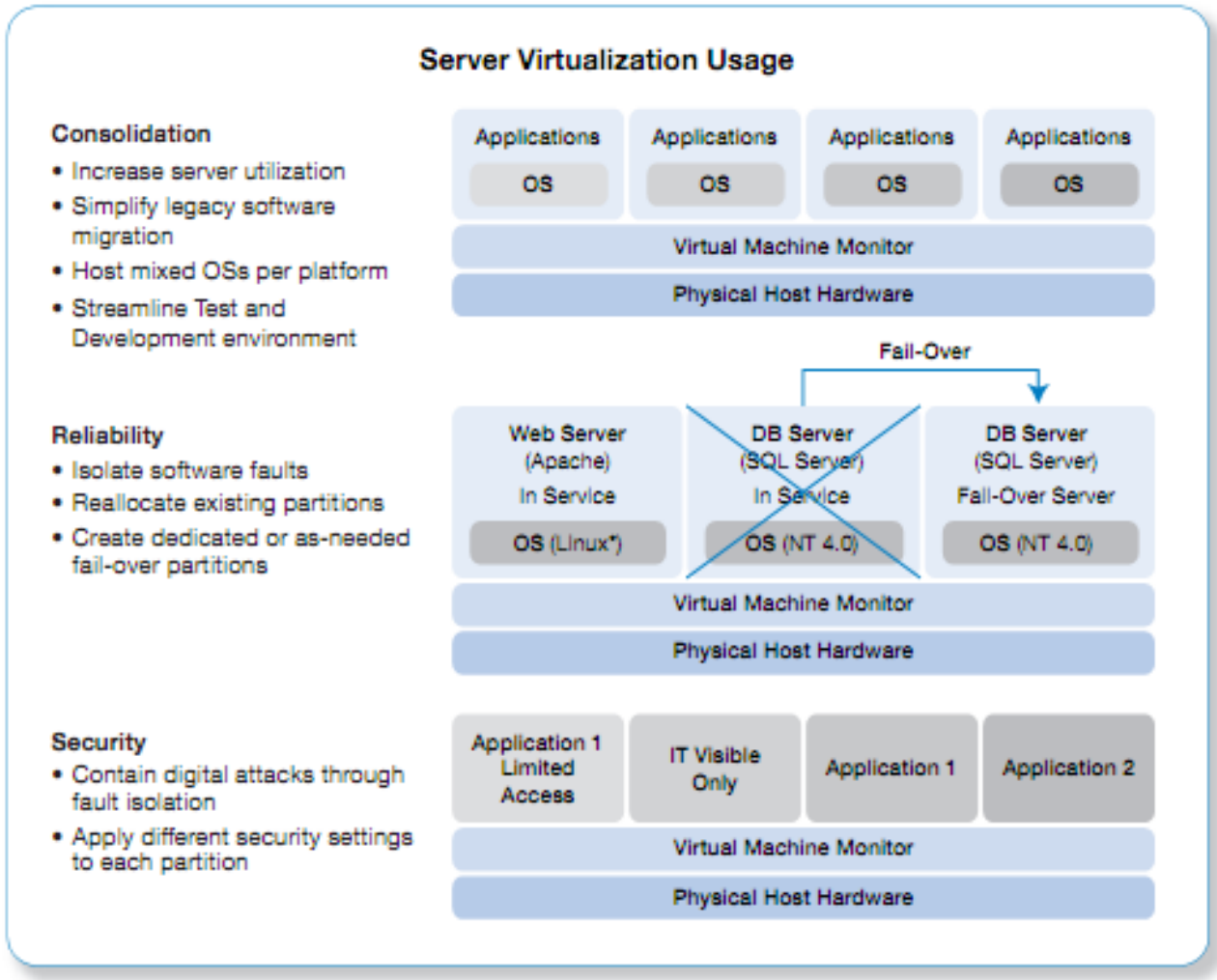
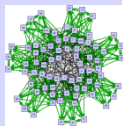


Figure 2. Server virtualization can be used to improve server utilization, reliability, and security, while simultaneously increasing business agility and reducing operational costs.

Vorteile:

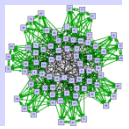
- Konsolidierung
  - eine Hardware für mehrere OS
- Zuverlässigkeit
- Sicherheit

Quelle:
Intel. Enhanced Virtualization on Intel Architecture-based Servers. White Paper, 2005



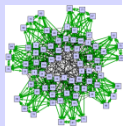
Aufbau eines modernen Rechenzentrums

- Weniger aber robuste Server
- Server bilden die Infrastruktur für eine Virtualisierung.
- Dienste werden dynamisch
  in virtuellen Maschinen (VM) betrieben.
- Rechenzentrum bietet "IT-as-a-Service“.



Forderungen nach Popek und Goldberg

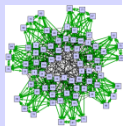
- Ein Virtual Machine Monitor (VMM) muss drei Dinge erfüllen, um Virtualisierung zu bieten:
  - Ressourcen-Kontrolle
  - Äquivalenz
  - Effizienz
- Eine CPU Architektur (ISA) muss folgende Bedingung erfüllen, um als Host für einen VMM geeignet zu sein
  - **The set of sensitive instructions must be a subset of the set of privileged instructions**
  - Consequence: all sensitive instructions of the guest OS trap to the VMM when executed in user mode
    - VMM provides kernel semantics for sensitive instructions



Ressourcen Kontrolle

Gast kann die direkte Konfiguration nicht ändern.

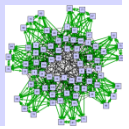
- VMM verwaltet die Systemressourcen.
- VM hat nur Zugriff auf die eigene Ressourcen.
- VMM kann Ressourcen frei zuweisen.



Äquivalenz

Jedes Programm, das auf dem Gast ausgeführt wird, muss sich verhalten wie auf der nativen Maschine.

- Programm kann nicht zwischen VM und realer Maschine unterscheiden.
- VM bietet eine identische Laufzeitumgebung.

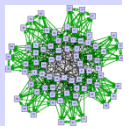


Effizienz

Instruktionen, die nativ ausgeführt werden können, sollten nativ ausgeführt werden,

ohne dass der VMM eingreifen muss.

- Überwiegende Menge der Instruktionen wird nativ auf der CPU ausgeführt.
- Nur wenige privilegierte Instruktionen werden vom VMM behandelt.
- VMM **muss** alle Instruktionen des Guest OS behandeln (können), die sich in User Mode anders verhalten als in Kernel Mode (sensitive instructions)



Emulation und Virtualisierung

Emulation

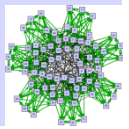
Duplizieren des Verhaltens eines Systems auf dem gleichen oder anderen System.

Beispiel: C64 Emulator

Virtualisierung

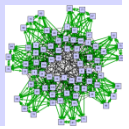
Folgt den Prinzipien von Popek und Goldberg: Äquivalenz, Effizienz und Ressourcen Kontrolle.

Beispiel: VMWare Server



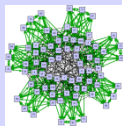
Privilegierte Instruktionen

- Modern processors have CPU modes that allow the OS and application programs to run at different privilege levels. Some processors have two levels (such as *user* and *supervisor*); i386+ CPUs have four levels (#0 =most, #3 =least privileges).
- Code is executed with a privilege level.
- Resources (segments, pages, ports, etc.) and the privileged instructions are tagged with a demanded privilege level.
- When code tries to execute a privileged instruction (or use a resource), the processor determines whether it has the permission
- If not, a "protection fault" interrupt (exception) is generated.
- **Execution of a privileged instruction in user mode traps to the OS**
- Example: LMSW Load Machine Status Word (IA32)



Sensitive Instructions

- Sensitive instructions have a semantics that is **sensitive to the execution mode**
  - verhalten sich – je nachdem in welchem Privilege Level sie ausgeführt werden – unterschiedlich.
- When executing a guest OS as an application program, its sensitive instructions change semantics
- The sensitive instructions of the guest OS must be replaced, or must be executed by the Hypervisor (VMM).
- Example (IA32):
  POPF Pop Flags off Stack
  also enable/disable IRQ in Ring 0 (IA32)
  but does nothing to IRQs in other rings (user mode).

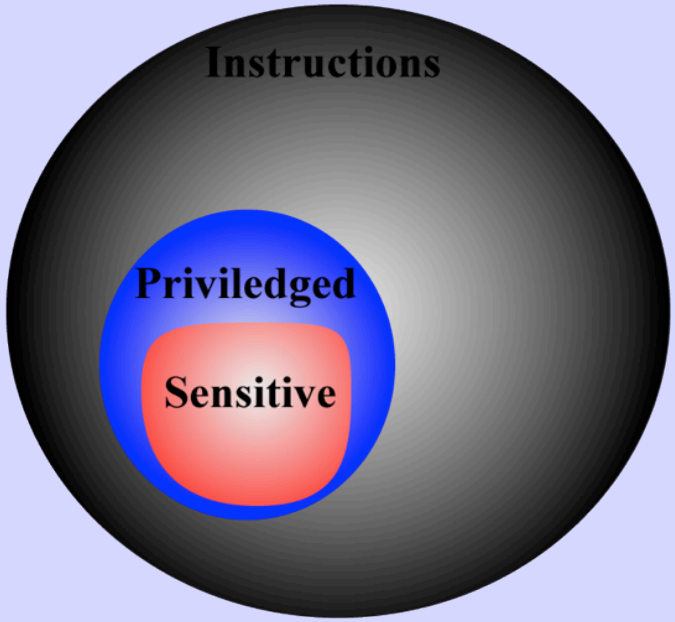


Popeg and Goldberg Theorem

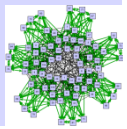
"For any third generation computer a VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged Instructions."

Anders formuliert:

"Every sensitive instruction must trap when executed in user mode"

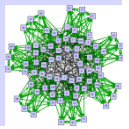


(Popek und Goldberg: Virtualization Requirements)

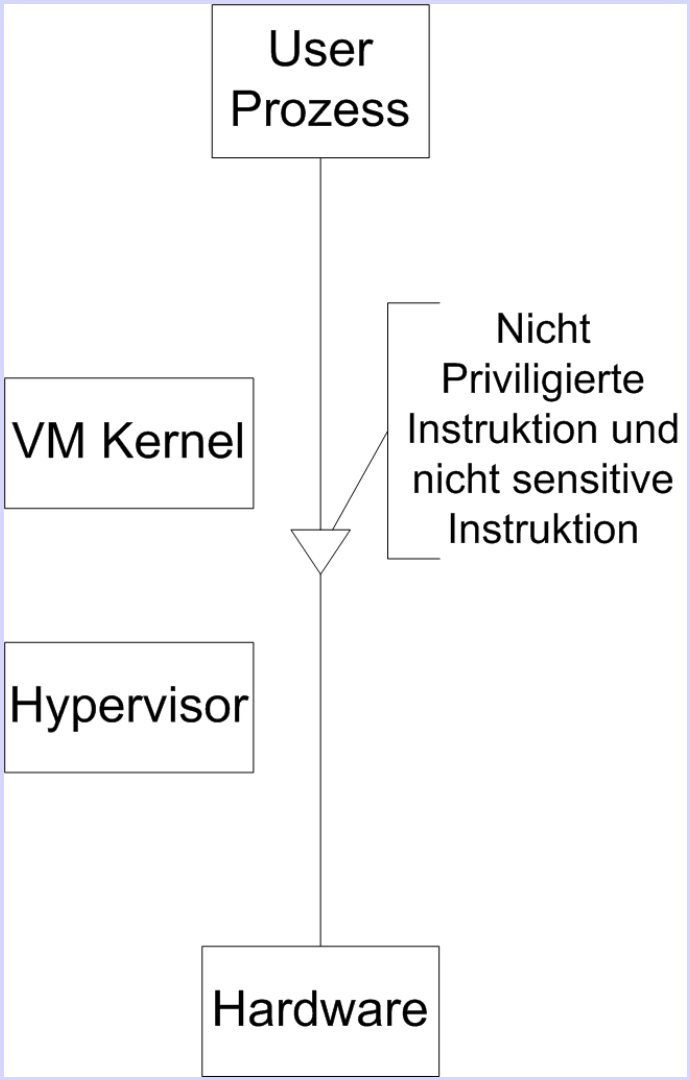


VMM einer Maschine nach Popek und Goldberg

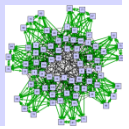
- Alle sensitiven Instruktionen sind auch privilegiert.
  - all sensitive instructions of the guest OS trap to the VMM
- When a guest OS executes a privileged instruction
  - the CPU generates a (synchronous) protection fault interrupt
  - the VMM handles the interrupt,
  - and thus intercepts the operation,
  - determines the guest OS trying to execute the instruction,
  - checks the operation for correctness,
  - and performs it on behalf of the guest.
- IBMs System/370 bietet diese Möglichkeit seit langem aufgrund eines entsprechend optimierten Befehlssatzes.



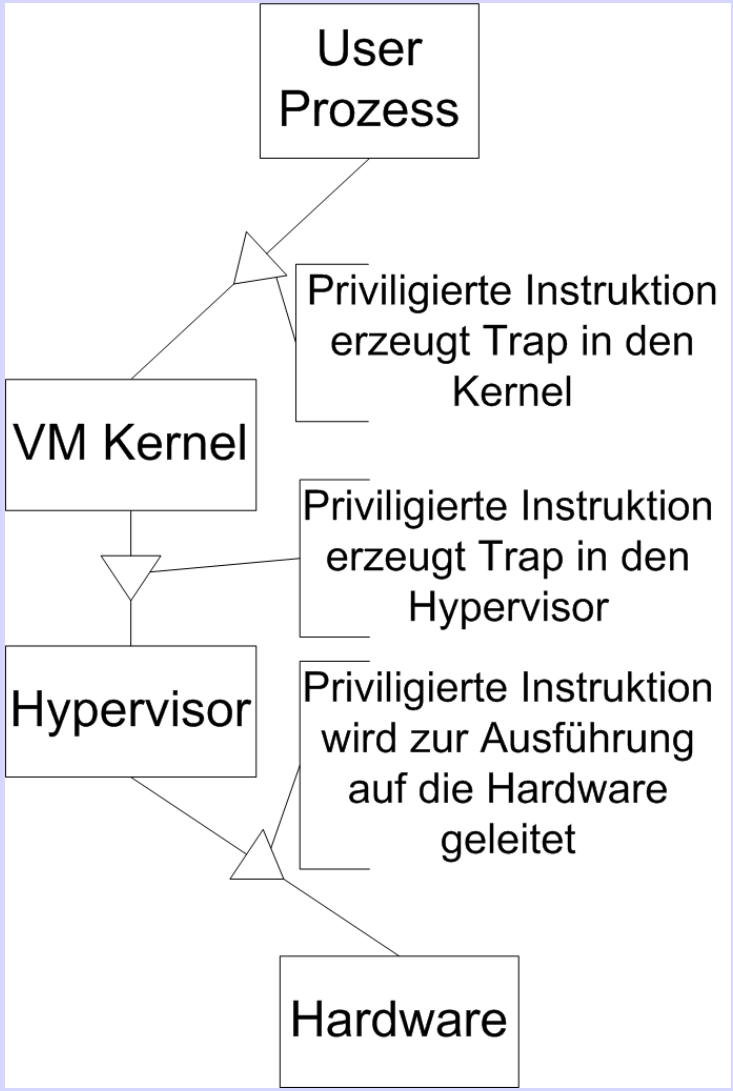
Ablauf einer nicht privilegierten Instruktion



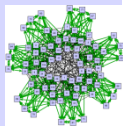
- Instruktion ist nicht privilegiert und auch nicht sensitiv.
- Instruktion kann nativ auf der Hardware ausgeführt werden.
  - falls keine andere Hardware emuliert werden muss
- Kein Eingreifen des Hypervisors ist nötig.



Ablauf eines System Calls



- Instruktionen können sensitiv sein.
- Instruktionen können Hardware Parameter ändern.
- Hypervisor behandelt den Systemcall.
- Da VM-Kernel im Userspace läuft erfolgt automatisch ein Trap in den Hypervisor.

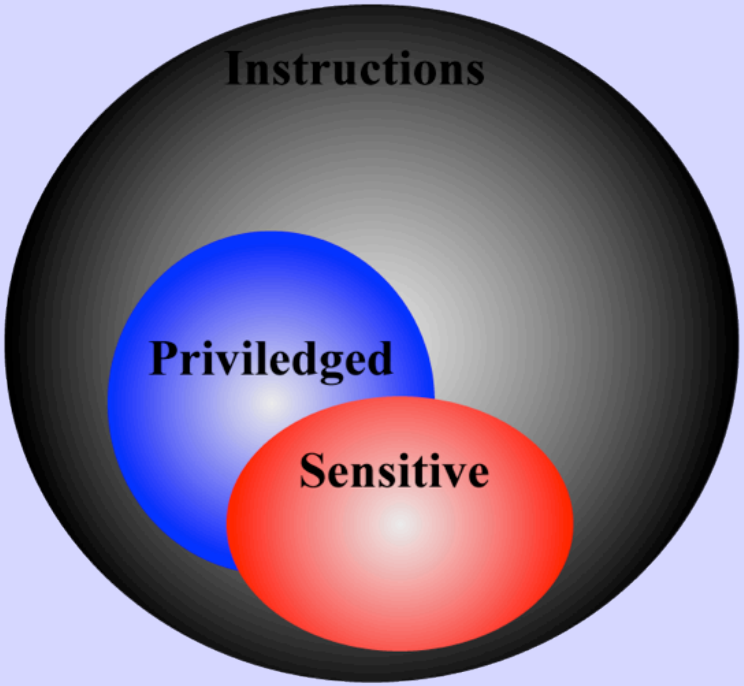


Virtualisierung auf x86 Hardware

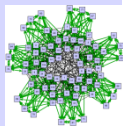
Der IA32 (x86) Befehlssatz enthält sensitive Instruktionen, die nicht privilegiert sind.

Daher erfüllt die x86-Hardware nicht die

Popek/Goldberg Bedingung für Virtualisierung.

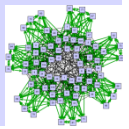


- Beispiel: POPF Instruktion
  - wird verwendet um Flags wiederherzustellen.
  - falls Ausführung in Ring 0 werden ebenfalls IRQs (re-)enabled.
  - falls Ausführung nicht in Ring 0 ist kein Zugriff auf IRQs möglich.

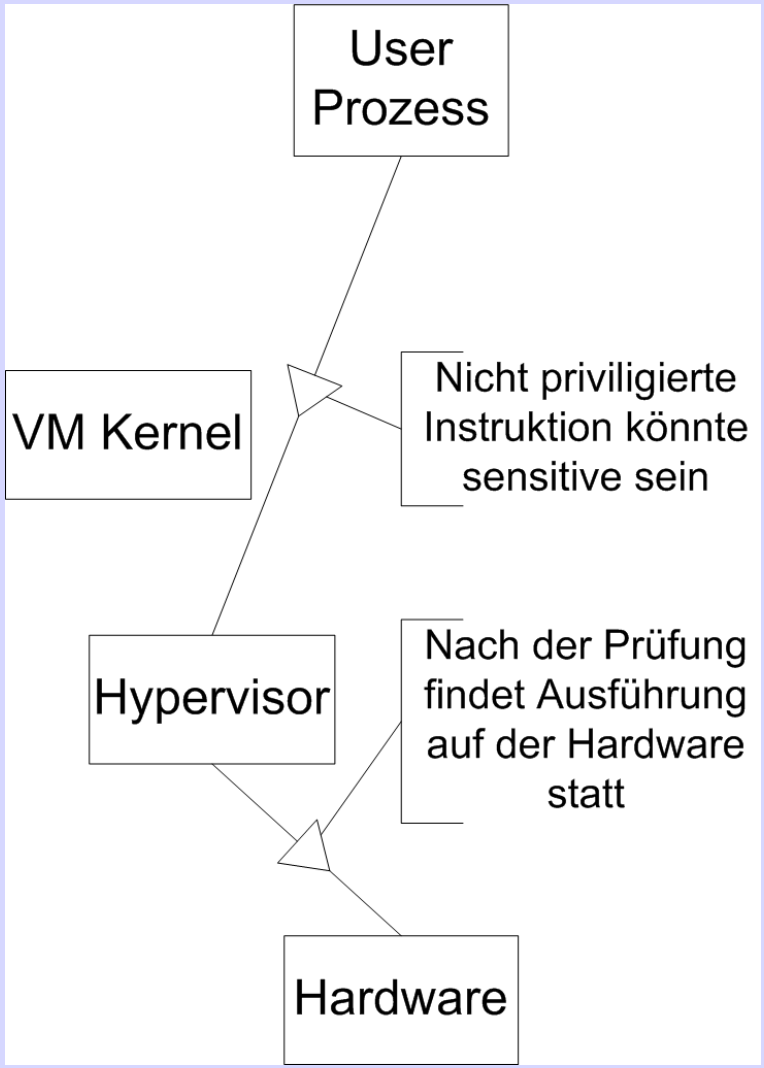


Problem bei POPF

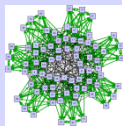
- POPF ist sowohl Ring 0 als auch Ring 1-3 Instruktion.
- Bei Virtualisierung läuft Gast-Kernel nicht in Ring 0.
- Gast-Kernel erwartet, dass bei POPF die IRQs modifiziert werden.
- POPF müsste von VMM behandelt werden, da Unterscheidung notwendig.



Ablauf eines User Prozesses ohne Popek/Goldberg



- Popek/Goldberg Bedingung ist nicht erfüllt
- Nicht alle sensitiven Instruktionen sind privilegiert.
- Nicht-privilegierte Instruktionen müssen auf Sensitivität geprüft werden.
- Hypervisor muss auch bei Userspace-Aufrufen eingreifen.

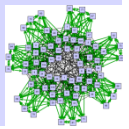


Probleme der Virtualisierung auf x86-Hardware

Sonderfälle nicht privilegierter sensitiver Instruktionen müssen behandelt werden.

Sonst kann es u.a. zu folgenden Probleme kommen:

- Ring Aliasing
- Address-space compression
- Non-faulting access to privileged state

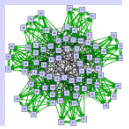


Non-faulting access to privileged state

Problem:

Es existieren Instruktionen, die im Userspace aufrufbar sind, und lesenden Zugriff auf privilegierte Teile des Systems erlauben.

(Äquivalenz)



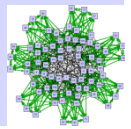
Ring aliasing

Problem:

Software wird nicht mit den Rechten ausgeführt für die sie geschrieben ist

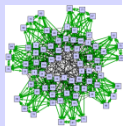
Beispiel:

OS wird in Ring 3 ausgeführt, dies kann zu ungewollten Effekten führen da das OS annimmt, die Rechte von Ring 0 zu haben. (Äquivalenz, Effizienz)



Lösungen

- Hypervisor fängt alle Instruktionen ab und übersetzt sie falls nötig (Binary translation).
- Kernel der VM wird angepasst und ist “Hypervisor aware” kann also in Ring 3 laufen (Paravirtualization).
- Hardware support für einen alternativen Ring 0 Modus (Hardware Support Intel VT).



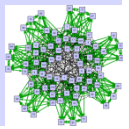
Address-space compression

Problem:

Der VMM braucht einen gewissen Teil des Speichers, um Strukturen zu halten. Betriebssysteme gehen aber davon aus, kompletten Zugriff auf den linearen Speicher zu haben.

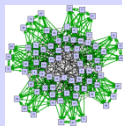
Beispiel:

Virtualisiertes OS kann nicht auf den Speicher zugreifen, der dem Hypervisor vorbehalten ist. Dadurch kann erkannt werden, dass virtualisiert wird. (Äquivalenz)



Lösungen

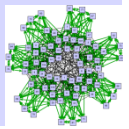
- Hypervisor fängt alle Instruktionen ab und passt sie falls nötig an (Binary translation).
- Kernel der VM wird angepasst und ist “Hypervisor aware” kann also in Ring 3 laufen (Paravirtualization).
- Hypervisor Kontrollstrukturen werden nicht in den Linearen Speicher abgelegt, sondern an eine definierte physikalische Stelle (Hardware Support Intel VT).



Lösungen

Zur Lösung der Probleme ergeben sich je nach Einsatzgebiet verschiedene Lösungswege

- Binary Translation (JIT Modell)
- Paravirtualisierung (AOT Modell)
- Hardware Support
- (OS Level Virtualisierung)



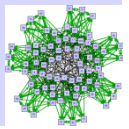
Binary Translation

Instruktionen werden zunächst
durch den VMM ausgeführt.

- ➢ Überprüfung auf sensitive Instruktionen
- ➢ Reinigung von sensitiven Instruktionen

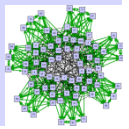
Nach initialer Ausführung durch den VMM können
Instruktionen nativ ausgeführt werden.

- Benötigt komplexe Lookup-Tabellen zum Ermitteln der sensitiven Instruktionen.
- Performance Einbußen, dafür jedoch universell einsetzbar
- VMWare ist der Pionier dieser Methode.



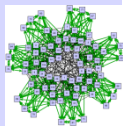
VMWare

- Pionier der x86-Virtualisierung
- basiert ursprünglich auf Binary-Translation
- grosser Provider für Rechenzentrumsinfrastruktur
- Pionier bei der Entwicklung von Features wie High Availability und Storage-Migration



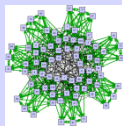
Paravirtualisierung

- Der Gast-Kernel wird modifiziert.
- Sensitive Instruktionen werden ersetzt.
- Ausführung des Codes wie nach Binary Translation
- Hohe Performance - benötigt jedoch den Quelltext des Kernels (des Guest-OS)
- Pionier ist das Xen Projekt



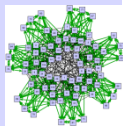
Xen

- Entwicklung der University of Cambridge
- Modifizierter Kernel erlaubt die Ausführung in Ring 3 bei x86 Prozessoren.
- Hohe Performance durch Kernel-Anpassung.
- beschränkt sich hauptsächlich auf Linux.
- Kernelversion stets einige Versionen hinter der Aktuellen zurück, da Modifizierung notwendig.



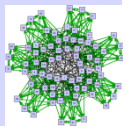
OS level Virtualisierung

- nur ein Kernel für viele Instanzen
- reine Virtualisierung des Userspace
- hohe Performance, allerdings keine unterschiedlichen Kernel möglich
- Einschränkung auf ein OS
  (Beispiel: FreeBSD bei Jails)



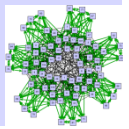
FreeBSD Jails

- bietet eine Sandbox für Nutzer und Programme.
- Zugang wird vom einzig aktiven Kernel kontrolliert.
- Sandbox ist limitiert auf FreeBSD.
- Sicherheitslücken des Kernels finden sich auch in den Jails wieder.
- fast native Performance

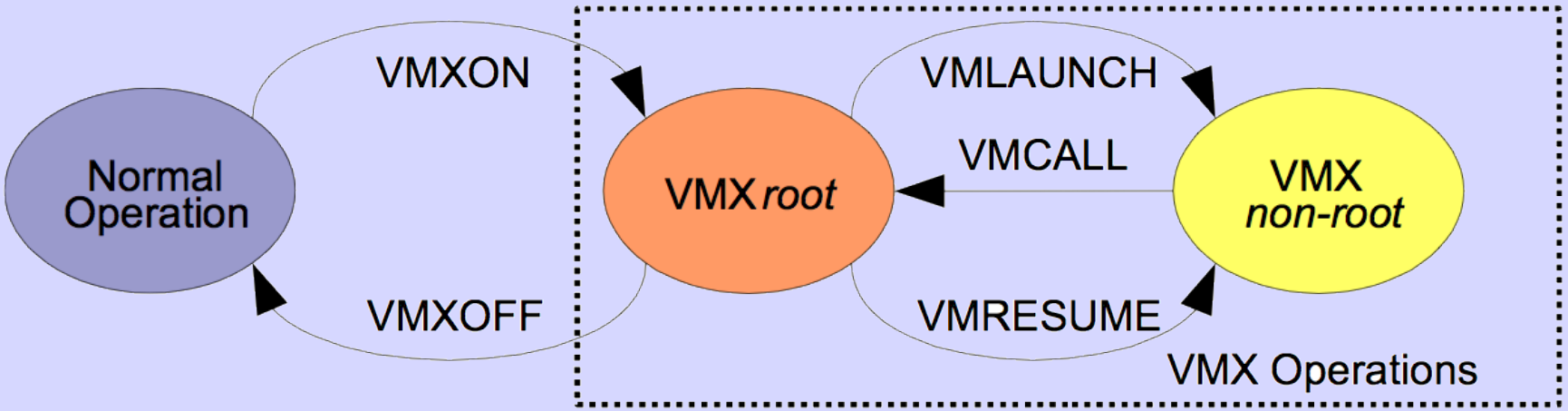


Hardware Support für x86-Virtualisierung

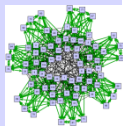
- Große Hersteller wollen x86-Virtualisierung unterstützen.
- arbeiten an Hardware Unterstützung.
- keine Änderung sondern nur Erweiterung des Befehlssatzes
- Ziel ist hohe Performance bei gleichzeitig universeller Einsetzbarkeit.



Intel Virtualisierungs-Technologien (VT-x)

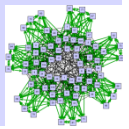


- VMM läuft im root Modus, das Guest OS als non-root
  - Eintritt in Guest heißt “VM-entry”, Austritt heißt “VM-exit”
  - Es sind jeweils alle Privileg-Ebenen 0-3 verfügbar
  - Die VMM markiert diejenigen Instruktionen, die (unabhängig vom Privileg-Level der VM) ein VM-exit bewirken sollen.



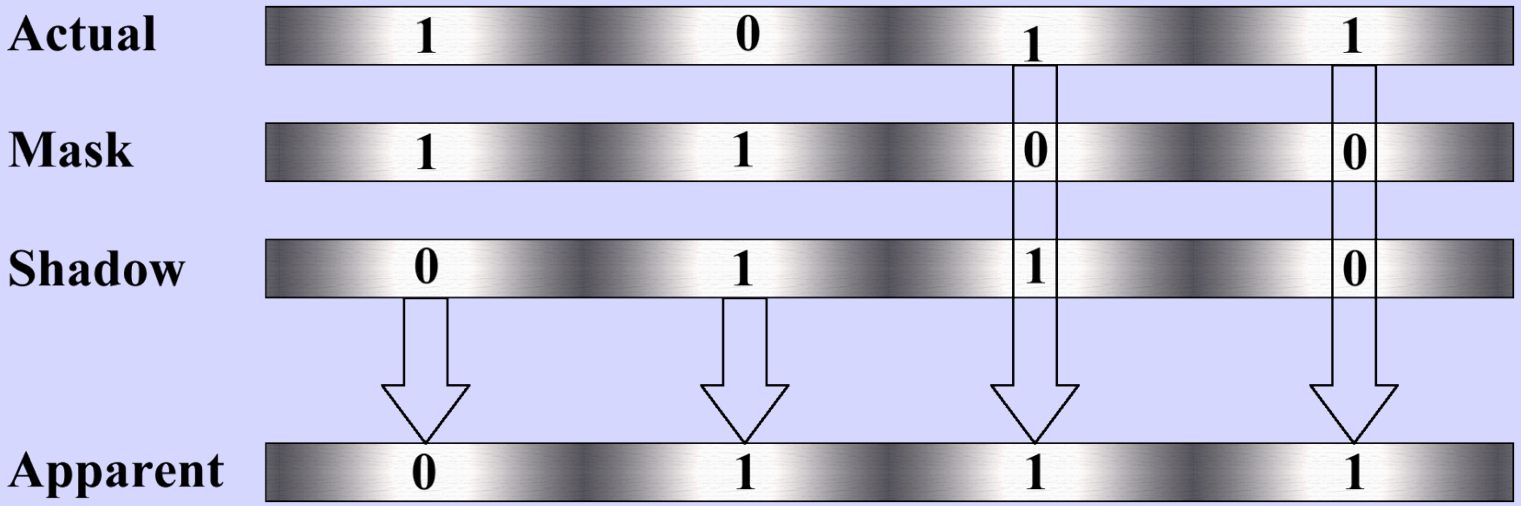
Intel Virtualisierungs-Technologien (VT-x)

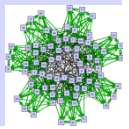
- VMX Root und Non Root Betriebs Modi für VMM und einzelne VM (Guest OS)
- VM Entry und Exit Operationen
- Virtual Machine Control Structure (VMCS)
  - sichert Prozessor-Zustände vom VMM und Guest
  - Bei VM exit wird Guest-Zustand gesichert und VMM Zustand geladen, bei VM entry wird Guest-Zustand geladen
- VM Execution Control Fields
  - erlauben es dem VMM, flexibel die Instruktionen und Ereignisse (Interrupts) zu spezifizieren, die VM-exits verursachen. Dadurch sind verschiedene Virtualisierungsstrategien möglich



VM-Execution Control Fields

- erlauben ein Maskieren von Instruktionen.
- Maskierte Instruktionen erzeugen VM-exit zum VMX Root Modus.

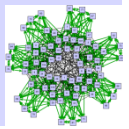




Vorteile von Intel-VT

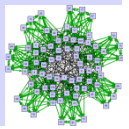
- keine Binary-Translation nötig
- höhere Kompatibilität
- bessere Isolation der VMs
  durch Hardware-Implementierung
- VMM einfacher zu implementieren
  - verschiedene Strategien möglich

- **Nachteile**
  - höhere Komplexität der Hardware
  - mögliche Optimierungen durch binary translation entfallen



Zusammenfassung

- Virtualisierung erlaubt
  die parallele Ausführung isolierter Services.
- Virtualisierung ist
  wichtiger Bestandteil moderner Architekturen.
- Virtualisierung bedeutet:
  - Äquivalenz
  - Effizienz
  - Ressourcen-Kontrolle durch VMM



Referenzen

- Enhanced Virtualization on Intel Architecture based Servers, White Paper Intel
- Jim Smith and Ravi Nair. Virtual Machines. Versatile Platforms for Systems and Processes, Chapter 8: System Virtual Machines
- VMware, Inc., www.vmware.com
- Xen.org, www.xen.org
- Intel-VT detailed overview, IEEE Computer, 2005
- Intel Technology Journal, Volume 10, Issue 03, Published August 10, 2006 www.intel.com/technology/itj/2006/v10i3/1-hardware/3-software.htm
- Workshop on Computer Architecture Education, WCAE 2008
- Popek and Goldberg. Formal Requirements for Virtualizable Third Generation Architectures
- James Smith and Ravi Nair. The Architecture of Virtual Machines. IEEE Computer Magazine May 2005.

- Philipp Fehre. Seamless Relocation of Virtual Machines in a Data Center Cloud. Diplomarbeit. Tübingen 2010.