Middleware – Cloud Computing

Virtualisierung

Wintersemester 2020/21

Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)

Lehrstuhl für Verteilte Systeme und Betriebssysteme

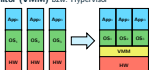
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Überblick

Virtualisierung
- Motivation
- Grundlagen
- Paravirtualisierung mit Xen
- Betriebssystemvirtualisierung mit Linux-VServer

Virtualisierung (im Kontext der Vorlesung)

- Einführung eines **Virtual Machine Monitor (VMM)** bzw. Hypervisor
  - Hinzufügen einer zusätzlichen Schicht bzw. Indirektionsstufe
  - *Virtuelle Maschine (VM)*: Vom VMM bereitgestellte Umgebung
  - Einsatzbereich (Beispiel): Server-Konsolidierung in Datenzentren



- Virtualisierungsebenen
  - **Systemvirtualisierung**
    - Virtualisierung der *Instruction Set Architecture*
    - Beispiel: Paravirtualisierung
  - **Prozessvirtualisierung**
    - Virtualisierung des *Application Binary Interface*
    - Beispiel: Betriebssystemvirtualisierung

Überblick

Virtualisierung
- Motivation
- Grundlagen
- Paravirtualisierung mit Xen
- Betriebssystemvirtualisierung mit Linux-VServer

Anforderungen an ein virtualisiertes System

- Eigenschaften nach [Popek et al.]
  - Äquivalenz
    - **Identisches Verhalten** im Vergleich zur nichtvirtualisierten Ausführung
    - Erlaubte Ausnahmen: Menge an verfügbaren Ressourcen, zeitliches Verhalten
  - Ressourcenkontrolle
    - VMM hat die **vollständige Kontrolle** über alle System-Ressourcen
    - VMM teilt VM Ressourcen zu, kann ihr diese aber auch wieder entziehen
  - Effizienz
    - Ausführung eines Großteils aller Instruktionen **direkt durch die Hardware**
    - Kein Umweg über den VMM für die meisten Instruktionen

„A **virtual machine** is taken to be an **efficient, isolated duplicate of the real machine.**"
[Popek et al.]

- Literatur
  - Gerald J. Popek and Robert P. Goldberg
    Formal requirements for virtualizable third generation architectures
    Communications of the ACM, 17(7):412–421, 1974.

Virtualisierbarkeit nach [Popek et al.]

- Existenz (mindestens) **zweier Betriebsmodi**
  - Uneingeschränkter Modus (*Supervisor Mode*)
  - Eingeschränkter bzw. Nutzer-Modus (*User Mode*)
- Kategorisierung von Instruktionen
  - **Privilegierte vs. nichtprivilegierte Instruktionen**
    - Privilegierte Instruktionen: *Trap* bei Aufruf im Nutzer-Modus
    - Nichtprivilegierte Instruktionen: Kein Trap bei Aufruf im Nutzer-Modus
  - **Sensitive vs. „harmlose" (innocuous) Instruktionen**
    - Sensitive Instruktionen können
      - Zustände außerhalb des Isolationsbereichs des Aufrufers beeinflussen
      - durch externe Zustände beeinflusst werden
    - Harmlose Instruktionen: alle nichtsensitiven Instruktionen
- Kriterium für Virtualisierbarkeit

Die Menge der sensitiven Instruktionen muss eine Teilmenge der Menge der privilegierten Instruktionen sein

Virtualisierung mittels Trap-and-Emulate

- **Reduzierung der Privilegien** des in der VM laufenden Betriebssystems

Beispiel: Privilegienzugewiesen der x86-Architektur, Ring 0: Supervisor Mode

- Aufgaben des Virtual Machine Monitor
  - Verwaltung von **Schattendatenstrukturen** (z. B. Register) für VM
  - Abfangen (→ Trap) der von der VM initiierten privilegierten Instruktion
  - **Emulation des von der VM erwarteten Verhaltens** einer Instruktion

Fallbeispiel: x86-Architektur

- Ergebnis einer Studie von [Robin et al.]
  - Insgesamt 17 von ~250 Instruktionen bei Pentium-CPUs problematisch
  - **Prozessoren im Sinne der Definition von [Popek et al.] nicht virtualisierbar**
- Beispiel: Zugriffe auf das Code-Segment-Register (CS)
  - In Teilen des Registers ist der aktuelle Betriebsmodus codiert
  - `Push`-Instruktion
    - Kopieren von Registerinhalten auf den Stack
    - **Nichtprivilegierte Instruktion**
  - Problematischer Zugriff in virtualisierter Umgebung
    - Ausführung eines Prozesses in einer VM im vermeintlichen Ring 0
    - Prozess liest per `Push` den Inhalt des CS-Registers aus
    - CS-Registerinhalt offenbart **Betriebsmodus mit geringeren Privilegien**
- Literatur
  - John Scott Robin and Cynthia E. Irvine
    Analysis of the Intel Pentium's ability to support a secure virtual machine monitor
    Proceedings of the 9th USENIX Security Symposium, S. 129–144, 2000.

Virtualisierung mittels Binary Translation

- Einsatz eines Interpreters
  - Virtuelle Maschine hat keinen direkten Zugriff auf CPU
  - **Übersetzung von VM-Instruktionen** auf Instruktionen der Zielplattform
  - Üblicherweise Basis-Blöcke als Übersetzungseinheit
- Vorgehen bei **identischen Instruktionssätzen** von VM und Hardware
  - **1:1-Abbildung aller nichtsensitiven Instruktionen**
  - Anpassung der sensitiven Instruktionen durch den VMM (Beispiele)
    - Illusion eines Betriebsmodus mit höheren Privilegien
    - Übersetzung von Speicheradressen
- Anwendungsbeispiel für x86-Architektur: VMware Workstation
- Literatur

Keith Adams and Ole Agesen
A comparison of software and hardware techniques for x86 virtualization
Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '06), S. 2–13, 2006.

Überblick

Virtualisierung
- Motivation
- Grundlagen

Paravirtualisierung mit Xen

Betriebssystemvirtualisierung mit Linux-VServer

Paravirtualisierung

- Ansatz
  - **Verzicht auf Einhaltung der Äquivalenz-Bedingung** von [Popek et al.]
  - Bereitstellung einer der Instruction Set Architecture „ähnlichen" Schnittstelle
  - Erweiterung des VMM um zusätzliche Methoden, die vom Betriebssystem einer virtuellen Maschine direkt aufgerufen werden können → **Hypercalls**
  - Sicherstellung der Isolation durch den VMM
- Konsequenzen
  - Hohere Effizienz durch Kooperation zwischen VM und VMM
  - Vereinfachte Implementierung des VMM
  - Um in einer paravirtualisierten Umgebung laufen zu können, ist eine **Portierung des (Gast-)Betriebssystems** erforderlich
- Beispiele
  - Xen
  - VMware ESX Server

Xen

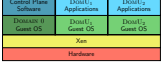
- Zielsetzungen
  - Gleichzeitiger Betrieb von bis zu 100 VMs auf demselben Rechner
  - **Einsatz heterogener Betriebssysteme** in virtuellen Maschinen
- **Xen-Hypervisor**
  - Virtual Machine Monitor für die x86-Architektur
  - **Keine Modifikation der Anwendungen erforderlich**
  - Hier betrachtet: Auf Paravirtualisierung basierende Xen-Variante
- Literatur

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho et al.
Xen and the art of virtualization
Proc. of the 19th Symp. on Operating Systems Principles (SOSP '03), S. 164–177, 2003.

David Chisnall
The definitive guide to the Xen hypervisor
Prentice Hall, 2007.

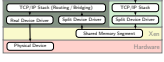
Architektur

- **Privilegierte Domäne** (*Domain 0, Dom0*)
  - Beim Systemstart von Xen automatisch erzeugt
  - Zugriff auf die Kontrollschnittstelle zur Verwaltung von Gastdomänen
    - Starten und Stoppen
    - VM-Scheduling: Abbildung von virtuellen Prozessoren auf reale Prozessoren
    - Konfigurierung von Speicherzuteilung, Netzwerkzugriff,...
- **Gastdomänen** ($DomU_x$)
  - Nichtprivilegierte virtuelle Maschinen mit eigenem Betriebssystem
  - Ausführung von Nutzeranwendungen



Verwaltung von Geräten

- Ansatz
  - Einsatz der Treiber des Betriebssystems der Domain 0
  - **Indirekter Hardware-Zugriff** über privilegierte Domäne
- Beispiel: Senden eines Netzwerkpakets aus einer Gastapplikation
  - Datenaustausch zwischen Domänen erfolgt per Shared Memory
  - Aufgaben der Domain 0
    - **Multiplexen der Hardware** für Zugriff mehrerer Gastdomänen
    - Anwendung von Firewall-Regeln



Überblick

Virtualisierung
Motivation
Grundlagen
Paravirtualisierung mit Xen
Betriebssystemvirtualisierung mit Linux-VServer

Betriebssystemvirtualisierung

- Ausgangspunkt
  - Nicht immer ist es erforderlich, virtuelle Maschinen mit heterogenen Betriebssystemen auf demselben physischen Rechner auszuführen
  - Optimierungsmöglichkeiten durch **Festlegung auf ein Betriebssystem**
- Ansatz
  - Verlagerung der Virtualisierung auf Ebene des *Application Binary Interface*
  - **Identischer Betriebssystemkern** für alle virtuellen Maschinen
  - Instanziierung des Betriebssystems
  - Ausnutzung von existierenden Mechanismen zur **Isolation von Prozessen**
- Beispiele
  - Linux-VServer
  - Docker [Gerber Übung]
  - FreeBSD Jail
  - Solaris Containers

- Container-basiertes Betriebssystem
  - **Standard-Linux mit Erweiterungen**
  - Nutzung bereits in Linux bestehender Mechanismen
    - Prozessspezifische, feingranulare Rechteverwaltung mittels *Linux Capabilities*
    - **Festlegung von Ressourcenlimits** pro Prozess (z. B. CPU)
    - Erweiterte Dateiattribute (z. B. `IMMUTABLE`: Schutz vor Modifikationen)
    - chroot: Ändern des Wurzelverzeichnis eines Dateisystems
- Einsatz (Beispiele)
  - **PlanetLab**
  - High-Performance-Cluster
- Literatur

Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson **Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors** *Proc. of the 2nd European Conf. on Computer Systems (EuroSys '07)*, S. 275–287, 2007.

Herbert Potzl, **Linux-VServer Paper**, `http://linux-vserver.org/Paper`

- **Virtuelle Maschinen**
  - VM für Administrations- und Verwaltungsaufgaben (*Host VM*)
  - VMs für Ausführung von Nutzerapplikationen (*Guest VMs*)
- Partitionierung von Ressourcen
  - Jede Partition stellt einen eigenen **Sicherheitskontext** dar
  - VM: **Gruppe von Prozessen**, die demselben Kontext zugeordnet sind
  - Isolation verschiedener Kontexte voneinander

Virtualisierung von Subsystemen

- Prozesse
  - **Filter zur Trennung von Prozessen** verschiedener VMs
  - Linux-Scheduler kombiniert mit *Token Bucket Filter*
    - Jeder virtuellen Maschine wird ein Token-Bucket zugeordnet
    - Jeder Token-Bucket wird mit einer individuellen Rate befüllt
    - Das Token-Kontingent einer aktiven VM wird schrittweise reduziert
    - Solange Tokens verfügbar sind, ist die korrespondierende VM lauffähig
- Netzwerk
  - **Anhängen der VM-Kontext-ID** an die Netzwerkpakete einer VM
  - Zuweisung von Netzwerkadressen zu VMs
  - Spezielle Behandlung der Adresse `localhost` erforderlich
- Dateisystem
  - Geteiltes Dateisystem für sich selten ändernde Dateien (z. B. Bibliotheken)
  - **Copy-on-Write-Ansatz** bei Modifikation: Erzeugung einer privaten Kopie