Prof. Dr. Udo Hönig, Prof. Dr. Wolfram Schiffmann

Modul 63713

Virtuelle Maschinen

LESEPROBE

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Leseprobe zum Kurs 1728

Virtuelle Maschinen

Dieser Kurs basiert auf einem Basistext, zu dem Sie unter folgender URL

http://store.elsevier.com/Virtual-Machines/Jim-Smith/isbn-9781558609105/

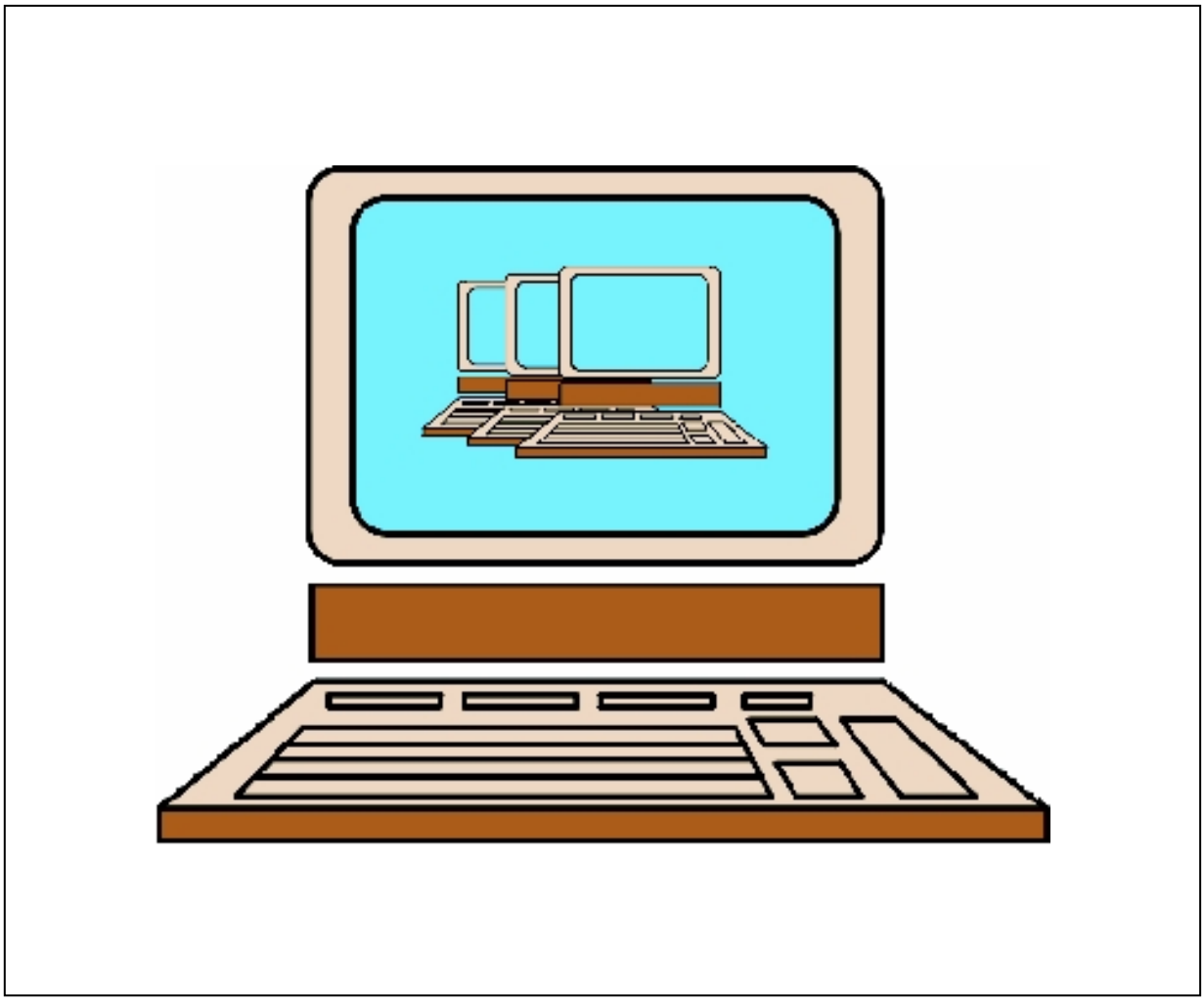
eine Leseprobe finden. Hier werden die ersten Seiten des dazugehörigen Leittextes bereitgestellt.

FernUniversität in Hagen

Kurs 1728: Virtuelle Maschinen

Leittext zu den Kurseinheiten 1 - 7

Autoren: U. Hönig und W. Schiffmann



Fakultät für Mathematik und Informatik

Inhaltsverzeichnis

Vorwort

Wir begrüßen Sie herzlich zum Kurs 1728 *Virtuelle Maschinen.*

Eine virtuelle Maschine emuliert ein komplettes Computersystem durch Softwarekomponenten, die einen einheitlichen Zugang zur Hardware eines realen Computersystems bereitstellen. Durch die Virtualisierung wird es möglich, auf ein und demselben Computersystem nacheinander oder auch gleichzeitig mehrere verschiedene Betriebssysteme laufen zu lassen. Die Virtualisierung von Computersystemen bietet eine Fülle von Vorteilen, wie z.B. die gleichzeitige Nutzung mehrerer Betriebssysteme, den einfachen und kostengünstigen Aufbau von Testumgebungen und die verbesserte Auslastung von Mehrkern-Prozessoren. Im Rahmen dieses Kurses werden die Grundlagen heutiger Virtualisierungstechniken herausgearbeitet.

Der Kurs basiert auf dem Buch "Virtual Machines"von Smith und Nair, erschienen bei Elsevier 2005 (ISBN 1558609105). Zu diesem englischsprachigen Basistext erhalten Sie anbei einen deutschsprachigen Leittext.

Kurseinheit 1

Grundlagen zu Virtualisierung und Emulation

Moderne Rechnersysteme beziehen ihre Leistungsfähigkeit aus dem Zusammenwirken der zugrunde liegenden Hardware mit der darauf betriebenen Software, die sich im allgemeinen aus einem Betriebssystem, diversen Bibliotheken und der eigentlichen Anwendungssoftware zusammensetzt. Zwischen diesen physischen und logischen Bestandteilen eines Gesamtsystems existieren wohldefinierte Schnittstellen, die von Details der gegebenen Implementierung abstrahieren und einen gemeinsamen Betrieb überhaupt erst ermöglichen. Gleichzeitig wird die Flexibilität bei der Zusammenstellung des Gesamtsystems erhöht. Sollte es dennoch zu Inkompatiblitäten zwischen einzelnen Systembestandteilen kommen, kann durch den Einsatz von Virtualisierungstechnologien eine reibungslose Interaktion erreicht werden. In dieser Kurseinheit werden zunächst die wesentlichen Bestandteile und Schnittstellen heutiger Systemarchitekturen und die daraus resultierenden Ansatzpunkte für den Einsatz von Virtualisierungstechnologien erläutert. Anschließend werden verschiedene Emulationsverfahren vorgestellt und kritisch hinsichtlich ihrer Verarbeitungsgeschwindigkeit und ihres Speicherbedarfs hinterfragt.

Lernziele

In dieser Kurseinheit lernen Sie

- die Architektur eines modernen Computersystems und die darin vorhandenen Schnittstellen,
- die im Zusammenhang mit Virtuellen Maschinen wesentlichen Grundlagen und Begriffe,
- eine Taxonomie der unterschiedlichen Typen von Virtuellen Maschinen,
- verschiedene Interpreter und Emulationsverfahren sowie ihre Vor- und Nachteile hinsichtlich Verarbeitungsgeschwindigkeit und Speicherbedarf
- sowie diverse Ansätze zur Optimierung dieser Verfahren

kennen.

1.1 Einführung in die Virtuellen Maschinen

Bitte lesen Sie Kapitel 1 des Basistextes und beachten Sie dabei die nachfolgenden Hinweise.

Obwohl Virtuelle Maschinen (VM) bereits seit mehreren Jahrzehnten in den unterschiedlichsten Aufgabengebieten eingesetzt werden, erlangen die ihnen zugrunde liegenden Konzepte und Technologien erst seit wenigen Jahren durch die zunehmende Verbreitung der JAVA-VM und die Einführung von MultiCore-Prozessoren einen gewissen Bekanntheitsgrad. Das Basiskonzept, nach dem Virtualisierungstechnologien funktionieren, ist dabei ungeachtet des jeweils vorgesehenen Einsatzbereichs oder der zugrundeliegenden System-Hardware nahezu unverändert geblieben: Systemarchitekturen setzen sich aus einer ganzen Reihe voneinander abhängiger Hardware- und Software-Komponenten zusammen, die über *wohldefinierte Schnittstellen* miteinander interagieren und durch ihr Zusammenwirken eine Verarbeitung des auszuführenden Programms ermöglichen. Diese Schnittstellen *abstrahieren* von Realisierungsdetails der einzelnen Komponenten und bewirken somit eine erhöhte Flexibilität hinsichtlich des Austauschs von Bestandteilen der Systemarchitektur. Beachten Sie hierzu bitte insbesondere auch Abbildung 1.4 des Basistextes. Während die neu hinzu kommenden bzw. modifizierten Komponenten in die hardwareseitige Richtung eine zur vorhandenen Systemarchitektur kompatible Schnittstelle aufweisen müssen, können die in Richtung des auszuführenden Anwendungsprogramms weisenden Schnittstellen durchaus von den bisherigen abweichen. In diesem Fall wird eine Systemarchitektur erreicht, die sich unter Umständen sogar deutlich von der tatsächlich vorhandenen unterscheidet. Virtualisierungstechnologien streben durch den Austausch bzw. die Modifikation einzelner oder mehrerer Architekturkomponenten eine Weiterentwicklung des Systems in Richtung eines avisierten Ziels an, beispielsweise einer beschleunigten Instruction Set Architecture (ISA) oder einer plattformunabhängigen Ausführungsumgebung. In Abhängigkeit von den betroffenen Systemkomponenten und der Art der vorgenommenen Änderungen lassen sich eine ganze Reihe unterschiedlicher Typen von Virtuellen Maschinen unterscheiden, über die Abbildung 1.13 des Basistextes einen Überblick bietet.

1.2 Emulationsverfahren

Bitte lesen Sie Kapitel 2 des Basistextes und beachten Sie dabei die nachfolgenden Hinweise. Achten Sie dabei insbesondere auf ein solides Verständnis der dargestellten Verfahren sowie ihrer Stärken und Schwächen. Nach dem Lesen sollten Sie die behandelten Optimierungsmaßnahmen kennen und ihre Einsatzmöglichkeiten erläutern können.

Eine im Kontext der Virtualisierung bedeutsame Technologie ist die Emulation, da sie einerseits eine wesentliche Grundlage für die Realisierung Virtueller Maschinen darstellt und andererseits wiederum auf Konzepten der Virtualisierung basiert. Unter einer *Emulation* versteht man ganz allgemein einen Vorgang, bei dem sowohl die Schnittstellen als auch die dahinter verborgene

Implementierung einer Systemarchitektur dahingehend verändert wird, dass sie den Schnittstellen und Funktionalitäten einer anderen Systemarchitektur entsprechen. Im Rahmen der vorliegenden Kurseinheit werden jedoch nur solche Emulatoren betrachtet, die auf der Befehlssatzebene arbeiten. Dabei werden die Befehle eines emulierten Gastsystems dahingehend ausgewertet und weiterverarbeitet, dass sie auf dem physisch vorhandenen Host-System ausgeführt werden können.

Zwei grundlegende Methoden, mit deren Hilfe eine solche Befehlssatzemulation zu verwirklichen ist, sind die *Interpretation* und die in diesem Kontext als *Binary Translation* bezeichnete *Übersetzung* der zu verarbeitenden Befehlsfolgen des Gastsystems. Zu beiden Methoden werden im Basistext verschiedene Realisierungsformen behandelt und miteinander verglichen. Ausgangspunkt ist dabei der sog. *Decode-and-dispatch-Interpreter*, der sowohl durch einen vergleichsweise einfachen Aufbau als auch durch eine relativ geringe Emulationsgeschwindigkeit gekennzeichnet ist. Weiterhin werden der *(Indirect-)Threaded-Interpreter* und der *Direct-threaded-Interpreter* besprochen, die beide über eine deutlich bessere Performance verfügen. Ein Blick auf Abbildung 2.5 sowie ein Vergleich der in den Abbildungen 2.7 und 2.9 gezeigten Programm-Fragmente erleichtert die Abgrenzung der genannten Verfahren.

Obwohl bei Binary Translation ein möglichst großer Teil des entstehenden Aufwands vor den Beginn der eigentlichen Emulation verschoben wird, gibt es dennoch zahlreiche Situationen, in denen ein solcher statischer Ansatz (*static translation*) nicht ausreichend ist. Daher müssen im allgemeinen auch während der Emulation Übersetzungsarbeiten durchgeführt werden, die dynamisch auf die jeweils vorhandenen Daten und Benutzereingaben reagieren können. Diese Verfahren werden im Basistext unter dem Oberbegriff *dynamic translation* (dynamische Übersetzung) zusammengefasst. In Kurseinheit 3 wird ausführlich auf Möglichkeiten zur Optimierung der dynamischen Übersetzung eingegangen. Eine Mischform zwischen statischer und dynamischer Übersetzung ist die inkrementelle Übersetzung (*incremental translation*), bei der einzelne Teile des auszuführenden Programms erst unmittelbar vor ihrer ersten Ausführung übersetzt werden. Durch dieses Vorgehen wird der zu Beginn der Emulation anfallende Übersetzungsaufwand reduziert und die Übersetzung von letztlich nicht benötigten Programmteilen vermieden. Die im Zusammenhang mit der inkrementellen Übersetzung eingeführten Basis-Blöcke (basic blocks) bilden die Grundlage für eine ganze Reihe von Virtualisierungstechnologien, die im Laufe der folgenden Kurseinheiten noch ausführlich behandelt werden.

Abbildung 2.38 gibt zusammen mit dem gesamten Abschnitt 2.10 eine allgemeine Übersicht über die behandelten Verfahren und deren Stärken und Schwächen. Die Kenntnis der in dieser Zusammenfassung behandelten Inhalte ist zwar eine unbedingt notwendige aber keineswegs hinreichende Voraussetzung für das Verständnis des weiteren Lehrtextes.