Forschungs- und Entwicklungsprojekte

Informatik Forsch. Entw. (2004)
Digital Object Identifier (DOI) 10.1007/s00450-004-0163-7

In dieser Rubrik erscheinen in unregelmäßiger Folge Kurzdarstellungen geplanter, laufender oder abgeschlossener Projekte. Die Darstellungen werden in der Regel von den Projektbeteiligten geliefert. Die Auswahl erfolgt durch die Herausgeber. Dabei wird die Bedeutung des Projekts für die Fortentwicklung der Informatik das Hauptkriterium sein. Bei geplanten und laufenden Projekten ist ein wichtiges Kriterium der Wunsch, Kontakte zu etablieren und die Zusammenarbeit zwischen verschiedenen Gruppen zu fördern. Bei abgeschlossenen Projekten geht es primär um die Vermittlung von Erfahrungen und Ergebnissen, die sich nicht für die Veröffentlichung in redaktionellen Beiträgen eignen.

Virtuelle Maschinen: zSeries- und S/390-Partitionierung

Joachim von Buttlar[1], **Wilhelm G. Spruth**[2]

[1] IBM Deutschland Entwicklung GmbH, Schönaicher Str. 220, 71032 Böblingen (e-mail: joachim_von_buttlar@de.ibm.com)
[2] Universität Tübingen, Schickard-Institut für Informatik, Sand 13, 72076 Tübingen (e-mail: spruth@informatik.uni-tuebingen.de)

Online publiziert: ♣ 2004 –

Zusammenfassung. Der gleichzeitige Betrieb mehrerer Gast-Betriebssysteme auf einem einzigen physischen Rechner unter einem Host-Betriebssystem ist eine leistungsfähige moderne Entwicklung. Bekannte Beispiele sind VMware für die IA32-Architektur sowie das Betriebssystem z/VM und die PR/SM-LPAR Einrichtungen der zSeries-Architektur. Die Nutzung eines Betriebssystems als Gast bedingt einen Leistungsverlust. Die als Partitionierung bezeichnete Zuordnung von Systemressourcen zu den einzelnen Gast-Betriebssysteme ist schwierig, wenn eine dynamische Anpassung an sich ändernde Lastprofile erforderlich ist. Diese Probleme lassen sich mittels Erweiterungen der Hardwarearchitektur adressieren, sowie durch Softwarestrukturen, welche diese Erweiterungen nutzen. Die Erweiterungen der Hardwarearchitektur gehen über das hinaus, was auf heutigen Rechnerarchitekturen wie IA32 oder Mips verfügbar ist. Der vorliegende Beitrag erläutert den optimalen Betrieb von Gast-Betriebssystemen und die begleitenden Partitionierungsmöglichkeiten auf der zSeries-Plattform und beschreibt die zusätzlichen Hardware- und Software-Einrichtungen, welche dies ermöglichen.

Schlüsselwörter: ♣

Abstract. The capability to run multiple guest operating systems simultaneously on a single hardware platform is a powerful feature in a modern computer system. Well-known examples are VMware for the IA32-architecture and the z/VM operating system and the PR/SM-LPAR facilities of the zSeries architecture. Running an operating system as a guest results in a performance degradation. Partitioning of system resources and assigning them to individual guests may be difficult, if a dynamic adaptation to an ever changing load profile is required. Extensions of the hardware architecture and their exploitation by software permit to address these problems. Such extensions have not been available on existing architectures like IA32 or Mips. The following paper discusses the operation of guest operating systems and associated partitioning capabilities available in zSeries systems and describes supporting hardware and software facilities.

Keywords: Hypervisor, Intelligent Resource Director, IRD, Interpretive Execution Facility, LPAR, LIC, OS/390, Parallel Sysplex, Partitionierung, PR/SM, Shadow Page Table, S/390, SIE, Start Interpretive Execution, Virtual Machine, VMM, zSeries, z/OS, z/VM

1 Einführung

Viele Benutzer haben den Bedarf, mehrere Betriebssysteme auf ihrem PC laufen zu lassen. Die Gründe sind vielfältig:

- Gleichzeitiger Betrieb von Windows, Linux und evtl. Solaris, FreeBSD und OS/2.
- Gleichzeitiger Betrieb von unterschiedlichen Windows Versionen (NT, 2000, XP), weil auf Grund von Kompatibilitätsproblemen zahlreiche Anwendungen nur auf einer bestimmten Windows-Version lauffähig sind.
- Nutzung mehrerer Windows-Instanzen für Entwicklung, Test, Experiment und Produktion.

Die einfachste Lösung ist der Einsatz eines Boot-Managers. Sie hat den Nachteil, dass das Umschalten auf ein anderes Betriebssystem einen Neustart erfordert und zeitaufwändig ist. Eine als *virtuelle Maschinen* bezeichnete Lösung betreibt mehrere ***Gast***-Betriebssysteme unter einem ***Host***-Betriebssystem. Den Gast-Betriebssystemen müssen Ressourcen wie CPUs, CPU-Zeit, Hauptspeicher sowie Zugriff auf Ein-/Ausgabe-Geräte und -Anschlüsse zugeordnet werden.

Daneben existieren zwei weitere Motivationen für den Einsatz von Gast-Betriebssystemen:

Partitionierung ist eine wichtige Funktion für den Großrechnerbetrieb. Systeme wie HP 9000 Superdome, IBM z990

oder Sun Fire 15K haben bis zu 72 CPUs. Vor allem bei Transaktions- und Datenbankanwendungen unterstützen die heutigen Betriebssysteme einen symmetrischen Multiprozessorbetrieb nur bis zu maximal 16 CPUs. Es ist deshalb erforderlich, derartige Rechner zu partitionieren, wobei in jeder Partition ein eigenes Betriebssystem läuft. Die Partitionierung kann entweder statisch erfolgen, oder alternativ an sich ändernde Anforderungsprofile dynamisch angepasst werden. Für die dynamische Partitionierung ist der Betrieb von Gast-Betriebssystemen unter einem Host-Betriebssystem ein besonders leistungsfähiger Ansatz.

Die **Konsolidierung** zahlreicher Server auf einem Rechner ist ein weiterer Trend in großen Unternehmen. Dies kann ebenfalls sehr effektiv mittels des Betriebs von Gast-Betriebssystemen unter einem Host-Betriebssystem erfolgen.

Es existieren mehrere kommerzielle Lösungen. VMware ist ein häufig eingesetztes Produkt für die IA32-Architektur (Pentium, AMD, andere). Ähnliche Ansätze sind für OS/400- und AIX-Rechner vorhanden. z/VM und PR/SM sind jedoch die wichtigsten Lösungen. Ihre technologischen Eigenschaften gehen über das hinaus, was auf anderen Plattformen derzeit verfügbar ist. Experten sprechen in diesem Zusammenhang von einem zehnjährigen Entwicklungsvorsprung [11].

Der vorliegende Beitrag erläutert neuartige Einrichtungen für den Betrieb von Gast-Betriebssystemen auf der zSeries-Plattform. Hierzu gehören Erweiterungen der Hardwarearchitektur sowie Software-Komponenten, welche diese für einen optimalen Betrieb von virtuellen Maschinen nutzen. Vergleichbare Voraussetzungen fehlen bisher auf anderen Plattformen, z.B. der IA32. Abschnitt 2 erläutert die Grundlagen. Abschnitt 3 beschreibt VMware und VM/370. Eine wichtige Weiterentwicklung stellt die Interpretive Execution Facility dar, die u. a. in z/VM benutzt wird, siehe Abschnitt 4. Abschnitt 5 diskutiert die Eigenschaften von PR/SM, Speicherverwaltung, Multiprocessing und IRD. Alternative Möglichkeiten der Partitionierung werden schließlich in Abschnitt 6 vorgestellt.

2 Virtuelle Maschinen

2.1 Übersicht

Ein Datenverarbeitungssystem hat grundsätzlich die Fähigkeit, ein Datenverarbeitungssystem anderer Architektur abzubilden. Das so entstandene *virtuelle* System kann Programme ausführen, die in der anderen Architektur geschrieben sind. Dieser Sonderfall der Simulation wird als Emulation bezeichnet. Um den beträchtlichen Leistungsabfall bei der Emulation erträglich zu halten, wird das virtuelle System häufig mit zusätzlichen Hardware- und Mikroprogramm-Einrichtungen ausgestattet [7,29]. Beispielsweise existieren mehrere Emulatoren, die es ermöglichen, ein S/390-Betriebssystem auf einer Nicht-S/390-Plattform laufen zu lassen. Hierzu gehören z.B. FLEX-ES von Fundamental Software [13] oder der Public Domain Hercules Emulator [18,**?**]♣ CON03 existiert nicht in der Literaturliste♣. Im Zeitraum 1963–1972 vertrieb Honeywell einen Emulator mit dem Namen Liberator, der eine erfolgreiche Emulation von IBM-1401-Programmen auf den Honeywell-H-200-Rechnern ermöglichte [HON 01].

Es existieren zwei Emulatoren für virtuelle Systeme, welche die IA32-Architektur auf anderen Architekturen emulieren. *Bochs* ist ein in C++ geschriebener Open-Source-IA32-Emulator, der auf vielen Plattformen lauffähig ist [4]. Für den Apple MAC ist das Software-Paket *VirtualPC* von Microsoft, ehemals Connectix, verfügbar [44]. Es gestattet den Ablauf von Windows-Programmen unter dem MAC OS X-Betriebssystem. Unter dem gleichen Namen, aber mit anderen Eigenschaften, vertreibt Microsoft ein Produkt für die IA32-Plattform.

Emulatoren sind grundsätzlich langsam, weil die einzelnen Maschinenbefehle des virtuellen Systems in einer anderen Architekturumgebung interpretiert werden. Ein *Hypervisor* oder *Virtual Machine Monitor* (VMM) ist ein Emulator, welcher besonders effektiv die eigene Systemarchitektur emuliert. Die Emulation schließt die Zentraleinheit mit ihrer Betriebssystem-Konsole (Administrator Console), den Hauptspeicher sowie E/A-Einheiten ein. Dieser Sonderfall eines virtuellen Systems wird als „virtuelle Maschine“ bezeichnet. *A "virtual machine" is a fully protected and isolated copy of the underlying physical machine's hardware. Thus, each virtual machine user is given the illusion of having a dedicated physical machine* [10]. Auf virtuellen Maschinen können gleichzeitig und unabhängig mehrere Kopien verschiedener Betriebssysteme der gleichen Hardwarearchitektur ablaufen.

Wir bezeichnen im Folgenden den Hypervisor als das *Host*-Betriebssystem und das emulierte Betriebssystem als das *Gast-Betrieb*ssystem, welches auf einer *Gast-Maschine* (virtuellen Maschine) läuft. Das Host-Betriebssystem verfügt über einen *Host-Kernel (Überwacher)* und das Gast-Betriebssystem verfügt über einen *Gast-Kernel*. Wir bezeichnen als Kernel denjenigen Teil des Betriebssystems, der im Kernel-Modus ausgeführt wird. Vielfach besteht der Hypervisor nur aus einem Host-Kernel. Es gibt aber Beispiele wie VMware, wo dies nicht der Fall ist und zusätzliche Betriebssystem-Funktionen in Anspruch genommen werden.

Der Host-Kernel übernimmt die Kontrolle des Systems immer dann, wenn eine Gast-Maschine einen Maschinenbefehl auszuführen versucht, der das korrekte Verhalten des Hosts oder einer anderen Gast-Maschine beeinflussen würde. Derartige Maschinenbefehle werden als *sensitive* Befehle bezeichnet. Ihre Eigenschaften wurden erstmals in [29] definiert und sind in [31] übersichtlich dargestellt Der Host-Kernel interpretiert diese Befehle für den Gast und stellt sicher, dass die Semantik erhalten bleibt. Wenn Gast und Host die gleiche Hardware-Architektur verwenden, können im Gegensatz zur Emulation anderer Architekturen alle nicht-sensitiven Maschinenbefehle eines Prozesses direkt von der CPU ausgeführt werden. Ein Prozess führt hauptsächlich nicht-sensitive Befehle aus, daher ist der Leistungsabfall nur gering, wenn der Prozess als Gast betrieben wird. Im Gegensatz dazu ist eine vollständige Emulation sehr aufwändig.

Normalerweise arbeitet der Host-Kernel im Kernel-Modus und der Gast-Kernel im Benutzer-Modus. Bei manchen Rechnerarchitekturen sind alle sensitiven Maschinenbefehle gleichzeitig privilegierte Befehle und können nur im Kernel-Modus ausgeführt werden. Beispiele sind die S/370-, zSeries- und Alpha-Architekturen [1,32]. Die Interpretation der sensitiven Befehle ist hier besonders einfach, weil ihre Ausführung durch den Gast automatisch eine Unterbrechung auslöst und damit einen Aufruf des Host-Kernels verursacht. [31] bezeich-

net Architekturen mit diesen Eigenschaften als *virtualisierbar* (virtualizable). Bei anderen Architekturen gehören auch manche nicht-privilegierten Befehle zu den sensitiven Befehlen. Beispiele sind die IA32- und die MIPS-Architektur. Hier muss durch einen zusätzlichen Mechanismus erreicht werden, dass diese Befehle durch den Host-Kernel interpretiert werden.

Damit lassen sich die bis heute verwirklichten virtuellen Maschinen einem von vier Typen zuordnen.

Typ 1 und 2 laufen auf nicht-virtualisierbaren Rechnerarchitekturen. Typ 1 löst das Problem der nicht-privilegierten, aber sensitiven Maschinenbefehle dadurch, dass manche zusätzlichen Funktionen interpretativ abgearbeitet werden. Beispiele sind VMware und VirtualPC. Typ 2 umgeht die Schwierigkeiten durch Änderungen im Gastbetriebssystem. Beispiele sind Xen, Disco und Denali.

Typ 3 läuft auf virtualisierbaren Rechnerarchitekturen. Beispiele sind die Betriebssysteme VM/370 und ICL 2900. Typ 4 verwendet einen Hardware/Software-Co-Design-Ansatz, bei dem die Hardwarearchitektur um Funktionen erweitert ist, die einen optimalen Betrieb von virtuellen Maschinen ermöglichen. Beispiele für konkrete Implementierungen sind z/VM und PR/SM. Grundsätzliche Fragen im Zusammenhang mit dem Hardware/Software-Co-Design für virtuelle Maschinen sind u. a. von [25,33,34] untersucht worden.

2.2 Basis-Operation

Gast-Betriebssysteme werden vom Kernel des Host-Betriebssystems wie Prozesse behandelt, die in unabhängigen virtuellen Adressenräumen laufen. Die Plattenspeicher des Gast-Betriebssystems sind oft als Files des Host-Betriebssystems realisiert. Im Falle von z/VM wird ein realer Plattenspeicher in Minidisks unterteilt (Abb. 1). Eine Minidisk besteht aus einem zusammenhängenden Datenbereich auf einer Platte, dessen Anfangsadresse (Blocknummer) dem Host-Kernel bekannt ist. Das Gast-Betriebssystem adressiert die Blöcke der Minidisk beginnend bei 0. Bei allen E/A-Operationen addiert der Host die Anfangsadresse der Minidisk. Der Zugriff über ein File-Directory entfällt.

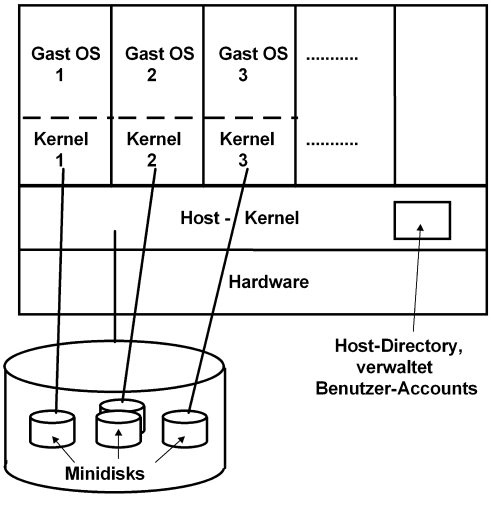

Abb. 1. Basiskonfiguration

Im folgenden Abschnitt nehmen wir an, dass die Hardware des Rechners nur über eine einzige CPU verfügt. In Wirklichkeit unterstützt z.B. der z/VM-Kernel einen symmetrischen Multiprozessor-Betrieb für bis zu 16 CPUs.

Eine virtuelle Maschine wird durch eine Datenstruktur dargestellt, welche Angaben über den zur Verfügung gestellten (virtuellen) Hauptspeicher, Plattenspeicherplatz, E/A-Konfiguration, die Zuordnung von virtuellen zu realen E/A-Einheiten sowie andere Ressourcen beschreibt. Diese Datenstruktur wird durch den Prozessleitblock des Host-Kernels sowie durch darin enthaltene Zeiger auf weitere Datenstrukturen verwirklicht.

Die Zeitscheibenverwaltung des Host-Kernels übergibt die Kontrolle der CPU einer Gast-Maschine. Der Kernel der Gast-Maschine läuft im Benutzer-Modus (User Mode). Wenn das Programm des Gast-Betriebssystems versucht, einen sensitiven Maschinenbefehl auszuführen, führt dies zu einer Programmunterbrechung. Die Programmunterbrechungsroutine des Host-Kernels interpretiert nun den sensitiven Maschinenbefehl für den Gast und gibt die Kontrolle an ihn zurück.

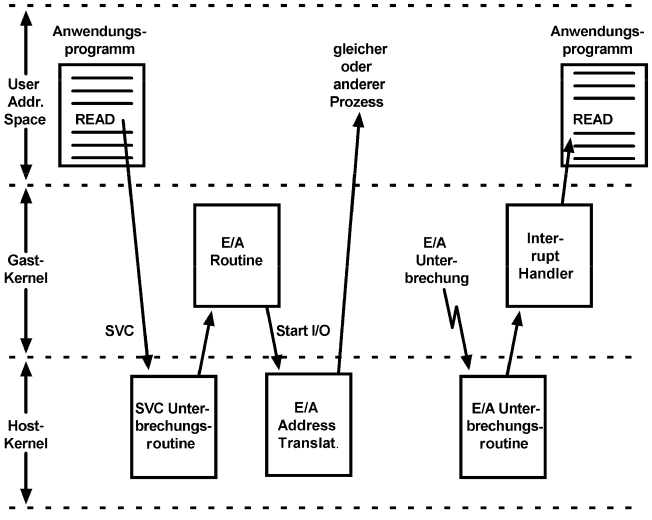

Abb. 2. Ablauf einer Ein-/Ausgabe Operation (Beispiel VM/370)

Dieser Vorgang sei am Beispiel einer VM/370-Ein-/Ausgabe-Operation erläutert, siehe Abb. 2, sowie [35, Abschnitt 8.3.2]. Ein Read-Makro im Benutzer-Adressraum eines Gast-Betriebssystems löst einen System Call (SVC-Maschinenbefehl) aus, der zu einer Unterbrechung des Host-Kernels führt. Dieser ruft den Gast-Kernel auf, der die Art des System Calls analysiert und die Ein-/Ausgabe-Routine (I/O Driver) des Gast-Kernels aufruft, welche ein S/370-Kanalprogramm erstellt (siehe [27], Abschnitt 2.3.2). Letzteres enthält normalerweise reale Hauptspeicheradressen für den Ein-/Ausgabe-Pufferbereich. Da die scheinbar realen Adressen des Gast-Betriebssystems in Wirklichkeit virtuelle Adressen des Host-Kernels sind, setzt der Host-Kernel die virtuellen Adressen anschließend in reale Adressen um, ehe die Ein-/Ausgabe-Operation mit dem SIO-Maschinenbefehl angestoßen wird.

Der Abschluss der Ein-/Ausgabe-Operation wird durch eine Ein-/Ausgabe-Unterbrechung an den Host-Kernel gemeldet. Dieser stellt nun sicher, dass die Steuerinformationen an

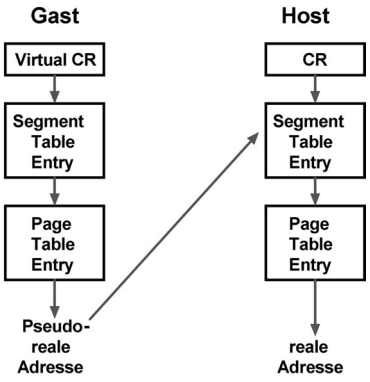

Abb. 3. Doppelte Adressübersetzung

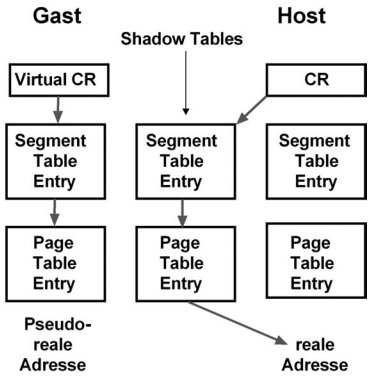

Abb. 4. Shadow Tables

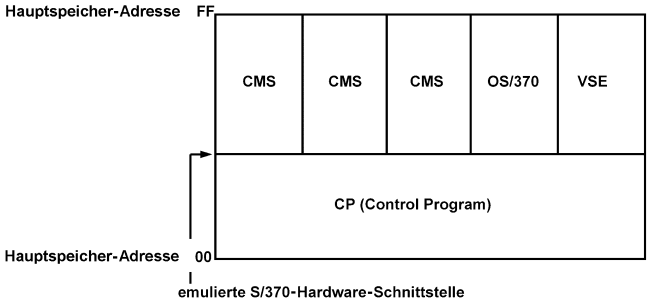

Abb. 5. VM/370

den Gast-Kernel weitergegeben werden. Damit ist das Gast-Betriebssystem sich nicht der Tatsache bewusst, dass es in Wirklichkeit unter einem Hypervisor läuft.

2.3 Verwaltung des virtuellen Speichers

Die S/370- und die IA32-Pentium-Architekturen verfügen beide über eine zweistufige Adressumsetzung mit Segment- und Seitentabellen (siehe [17, Abschnitt 6.5.2.2]). Die Anfangsadresse der derzeit gültigen Segmenttabelle steht in einem Kontroll-Register (CR), dem Segmentabellen-Basisregister.

Ein Gast-Betriebssystem speichert die Segment- und Seitentabellen in seinem Kernel-Bereich. Der Adressraum, den der Gast als real ansieht, ist jedoch in Wirklichkeit virtuell und wird vom Host-Kernel ebenfalls über Segment- und Seitentabellen beschrieben. Konzeptuell ist jetzt die in Abb. 3 dargestellte, doppelte Adressübersetzung erforderlich. Hierfür ist jedoch keine Hardware vorhanden. Daher erstellen VM/370 und VMware anhand der Gast- und ihrer eigenen Tabellen sogenannte *Shadow Tables*, die direkt die virtuellen Adressen des Gastes auf reale Adressen des Hosts abbilden. Diese Tabellen liegen im Speicherbereich des Host-Kernels (Abb. 4). Im Falle einer Fehlseitenunterbrechung wird der Gast-Kernel seine eigenen Segment- und Seitentabellen ändern. Der Host-Kernel benutzt diese Information für eine Änderung der entsprechenden Einträge in den Shadow Tables. Bevor der Host die Kontrolle an einen Gast abgibt, wird der Zeiger auf diese Shadow Tables in das Segmenttabellen-Basisregister geladen.

Zur Beschleunigung der Adressumsetzung verfügt die Hardware über einen Adressumsetzpuffer (translation lookaside buffer, TLB). Während der Ausführung des Gastes wird der TLB mit Einträgen aus den Shadow Tables gefüllt. Wenn der Gast-Kernel Einträge in seinen eigenen Tabellen ändert, muss er die entsprechenden Einträge im TLB löschen. Dazu verwendet er sensitive Maschinenbefehle, die vom Host-Kernel abgefangen werden und diesen veranlassen, die Shadow Tables abzuändern.

3 Arbeitsweise von VM/370 und VMware

3.1 VM/370

Das Betriebssystem VM/370 besteht aus einem Host-Kernel *(Control Program, CP)* und darauf aufsetzenden S/370-kompatiblen Gast-Betriebssystemen. Die Schnittstelle zwischen dem Gast-Kernel und dem Host-Kernel ist eine simulierte S/370-Hardware. Der Gast-Kernel ist sich der Existenz des Hypervisors nicht bewusst und hat den Eindruck, alleiniger Benutzer einer S/370-Maschine zu sein. Somit kann die Rolle dieses Gast-Kernels vom Kernel eines jeden unter S/370 lauffähigen Betriebssystems übernommen werden. VM/370 verfügt daneben über ein zusätzliches Gast-Betriebssystem, das besonders auf das Arbeiten in einer virtuellen Maschine zugeschnitten ist (Conversational Monitor System, CMS). Die Zusammenarbeit mehrerer Betriebssysteme unter einem VM/370-Hypervisor ist in Abb.5 wiedergegeben.

Das CP stellt jedem Gast-Betriebssystem einen eigenen virtuellen Adressraum zur Verfügung und läuft im Kernel-Modus (Kernel Mode), während die Gast-Betriebssysteme einschließlich ihrer Kernel-Funktionen nur im Benutzer-Modus (User Mode) arbeiten. Privilegierte Maschinenbefehle (z.B. Ein-/Ausgabe) werden vom CP abgefangen und interpretativ abgearbeitet. **Es existieren keine sensitiven Maschinenbefehle, die nicht gleichzeitig auch privilegiert sind.**

Im einzelnen verfügt VM/370 über die folgenden Eigenschaften:

- Der CP-Kernel bewirkt die Abbildung der virtuellen Komponenten auf reale Komponenten.
- Die Konsole des Gast-Betriebssystems wird durch die Datenstation des Benutzers oder Administrators des Gast-Betriebssystems dargestellt.
- Die virtuelle Zentraleinheit wird durch Zeitscheiben der realen Zentraleinheit abgebildet.
- Die Abbildung des virtuellen Hauptspeichers erfolgt innerhalb der Rahmen des realen Hauptspeichers.
- Die virtuellen Plattenspeicher werden durch als *Minidisks* bezeichnete Bereiche auf den realen Plattenspeichern dar-

gestellt. Es können auch ganze Plattenspeicher als Minidisks deklariert werden.
- Bei Bedarf können Kanäle (channel paths) und Plattenspeicher einem Gast-Betriebssystem fest zugeordnet werden. Dies gilt ebenso für andere Ein-/Ausgabeeinheiten (z.B. Ethernet-Anschlüsse innerhalb eines Kommunikations-Adapters).

Der CP-Kernel bildet virtuelle E/A-Einheiten auf realen E/A-Einheiten des gleichen Typs ab. Obwohl jeder Grundtyp eines E/A-Geräts der virtuellen Maschine auch im realen System vorhanden sein muss, kann z.B. im Detail ein Plattenspeichertyp des virtuellen Systems durch einen anderen Typ im realen System ersetzt werden.

CMS ist ein besonders für die Software-Entwicklung ausgelegtes Einzelplatz-Betriebssystem, ähnlich MS-DOS. Eine frühe Version von CMS war seinerzeit auf der realen S/360-Hardware lauffähig. Für jeden CMS-Benutzer wird eine eigene Instanz (Kopie) des Betriebssystems angelegt, die in einem eigenen virtuellen Adressraum läuft.

Der CMS-Kernel ist reentrant, daher wird nur eine einzige physische Kopie in gemeinsam genutzten Segmenten von beliebig vielen Benutzern gleichzeitig verwendet. Eine zusätzliche CMS-Instanz wird geschaffen, indem ein neuer virtueller Adressenraum mit Datenstrukturen sowie Zeigern auf den gemeinsam benutzten Code erstellt wird.

Unter VM/370 können beliebige S/370-Betriebssysteme laufen. Auch VM/370 kann wiederum als Gast betrieben werden, z. B. um eine neue Version zu testen. Die reale Maschine wird als Ebene 0 bezeichnet. Eine virtuelle Maschine (VM), die auf Ebene 0 läuft, wird als Ebene 1 bezeichnet. Eine VM, die auf Ebene 1 läuft, wird als Ebene 2 bezeichnet, u.s.w. [16].

Ähnliche Funktionen wie VM/370 hatten die ICL-Virtual-Machine-Environment-Betriebssysteme VME/K und VME/B, die seinerzeit auf Rechnern der ICL-2900-Architektur verfügbar waren. Neben der Emulation der eigenen Maschinenarchitektur war es auch möglich, virtuelle Maschinen aufzusetzen, welche die älteren ICL-1900- oder ICL-System-4-Architektur emulierten [24].

3.2 VMware

VMware ist ein Produkt mit Virtual-Machine-Hypervisor-Funktionalität für die IA32-Architektur und ist in zwei Versionen verfügbar: *VMware Workstation* und *ESX Server*.

VMware Workstation ist die ursprüngliche Version. Es handelt sich um ein als Host-OS bezeichnetes Anwendungsprogramm, das entweder unter Windows oder unter Linux auf einer IA32-Plattform im Benutzer-Modus läuft und als Kombination die Funktionalität eines Host-Kernels bereitstellt. Eine zusätzliche, als *VMDriver* bezeichnete Komponente wird in Windows oder Linux integriert. Es bestehen Ähnlichkeiten mit der Art, wie eine virtuelle Maschine in einem DOS-Fenster unter Windows abläuft. VMware Workstation nutzt die existierenden Ein-/Ausgabe-Einrichtungen des Host-Betriebssystems. Wenn ein Gast-Betriebssystem eine Ein-/Ausgabe-Operation ausführt, wird diese von VMDriver abgefangen und unter Benutzung geeigneter Systemaufrufe interpretiert. Ebenso werden Ein-/Ausgabe-Unterbrechungen unter Beteiligung des VMDrivers bearbeitet.

Das Host-OS kann Seiten auslagern, die einer spezifischen virtuellen Maschine zugeordnet sind. Lediglich ein kleiner Satz an Seiten wird ständig im Hauptspeicher gehalten. Dies bedeutet, dass VMware Workstation vom Host-Betriebssystem wie ein normaler Benutzer-Prozess behandelt wird. Falls der Algorithmus des Hosts zur Seitenersetzung nicht optimal arbeitet, kann dies zu einer VMware-Leistungsverschlechterung führen.

Die Nutzung von Host-Betriebssystemkomponenten verursacht einen Leistungsverlust, insbesondere bei Ein-/Ausgabe-Operationen (siehe hierzu [40]). VMware stellt deshalb mit seinem ESX Server einen eigenen Host-Kernel zur Verfügung. Dieser benötigt keine Unterstützung durch das Host-Betriebssystem, läuft auf der realen Hardware und hat vergleichbare Funktionen wie der VM/370-Host-Kernel. Es werden bis zu 64 virtuelle Maschinen unterstützt. Gast-Betriebssysteme, die unter dem ESX-Host–Kernel laufen, verfügen über einen mit der Adresse 0 beginnenden linearen virtuellen Adressenraum. *Die* Adressumsetzung erfolgt ähnlich wie bei VM/370 mit Hilfe von *shadow page table*s und einer als *pmap* bezeichneten Datenstruktur. Gast-Maschinenbefehle, die Gast-Seitentabellen oder den TLB abändern, werden vom ESX-Host–Kernel abgefangen und interpretiert. Der TLB enthält hierzu immer die in den shadow page tables enthaltenen Adressumsetzungen. Die wichtigsten Ein-/Ausgabe-Treiber sind in Bezug auf maximale Leistung im Gastbetrieb-Modus optimiert. Die derzeitige Version ist in der Lage, einen symmetrischen Multiprozessor (SMP) mit bis zu 2 CPUs zu unterstützen. Eine Übersicht ist in [9] enthalten; Eigenschaften von ESX Servern sind in [45] beschrieben.

Im Vergleich zu VM/370 sind ESX Server und VMware benachteiligt, weil einige kritische Eigenschaften in der IA32-Architektur fehlen. Für den Betrieb von Gast-Maschinen ist es erforderlich, dass alle Maschinenbefehle, welche den privilegierten Maschinenstatus abändern oder auch nur lesen, nur im Kernel-Modus ausgeführt werden können [29].

Dies sei an einem Beispiel erläutert. Wenn ein Gast ein Kontroll-Register schreibt, muss der Host-Kernel diesen Maschinenbefehl abfangen, damit nicht das reale Kontroll-Register des Hosts verändert wird. Der Host-Kernel wird jetzt nur die Effekte der Instruktion für diesen Gast simulieren. Liest der Gast anschließend diese Kontroll-Register wieder aus, so muss diese Instruktion ebenfalls abgefangen werden, damit der Gast wieder den Wert sieht, den er vorher in das Register geschrieben hat, und nicht etwa den realen Wert des Kontroll-Registers, der nur für den Host sichtbar ist.

Da die IA32-Architektur diese Bedingung nicht erfüllt, ist es nicht möglich, wie unter VM/370 alle Maschinenbefehle einfach im Benutzer-Modus auszuführen und auf Programmunterbrechungen zu vertrauen, wenn auf privilegierte Maschinenstatusinformation zugegriffen wird. Beispielsweise:

> *Many models of Intel's machines allow user code to read registers and get the value that the privileged code put there instead of the value that the privileged code wishes the user code to see* [8].

Eine detaillierte Diskussion dieses Problems ist in [26] und [31] zu finden

VMwares ESX Server überschreibt hierzu dynamisch Teile des Gast-Kernels und schiebt Unterbrechungsbedingungen dort ein, wo eine Intervention des Host-Kernels erforderlich

ist. Als Folge hiervon tritt ein deutlicher Leistungsverlust auf, besonders bei Ein-/Ausgabe-Operationen. Manche Funktionen sind nicht vorhanden oder können nicht genutzt werden. Kompatibilitätsprobleme treten auf, Anwendungen sind möglicherweise nicht lauffähig.

3.3 Weitere Implementierungen

Ein anderes Produkt für die IA32-Plattform ist VirtualPC von Microsoft [21]. VirtualPC wurde ursprünglich von Connectix entwickelt. SW-Soft setzt ihre Virtuozzo Virtual-Private-Servers (VPS) Software vor allem für Hosting-Services bei Hosting-Providern ein [43]. Als Konkurrent tritt Ensim mit der konzeptuell ähnlichen Extend-Software auf [12]. SW-Soft bezeichnet seine Produkte als *virtuelle Server*. Der als Virtual Environment (VE) bezeichnete Gast ist vom gleichen Typ wie das Host-Betriebssystem, normalerweise Linux. Er erscheint nach außen hin wie ein kompletter Server, verfügt aber nicht über einen eigenen Kernel, sondern benutzt stattdessen die Kernel-Funktionen des Hosts. Zwischen dem Host-Betriebssystem und den virtuellen Environments befindet sich eine Zwischenschicht, die die Steuerung und Verwaltung der VEs übernimmt. Sie erweckt den Anschein, als ob eine Gruppe von Anwendungsprozessen auf getrennten Instanzen des Kernels liefe. Die VEs sind gegeneinander abgeschottet, können unabhängig voneinander gestartet und beendet werden und besitzen je ein eigenes Dateisystem.

Mehrere Projekte adressieren das Problem der fehlenden Virtualisierungseinrichtungen der Host-Architektur, indem sie das Gast-Betriebssystem abändern.

Xen ist ein GNU Open Source Software Virtual Machine Monitor, der derzeitig von der Systems Research Group des University of Cambridge Computer Laboratory entwickelt wird [2] und ebenfalls auf der IA32-Plattform läuft. Um die Probleme mit der IA32-Architektur zu umgehen, wird, im Gegensatz zu VMware, ein als *Paravirtualization* bezeichneter Ansatz verwendet. Hierbei ist die Architektur der virtuellen Maschine nicht vollständig mit der Host-Architektur identisch. Die Benutzerschnittstelle (Application Binary Interface, ABI) ist die gleiche. Der Gast-Kernel unterstellt jedoch Abweichungen zur IA32-Architektur. Dies verbessert das Leistungsverhalten, erfordert aber Änderungen des Gast-Kernels. Hiervon ist nur ein sehr kleiner Teil des Kernel-Codes betroffen. Derzeit existiert ein funktionsfähiger Linux-Port (XenoLinux), dessen ABI mit dem eines nicht-virtualisierten Linux 2.4 identisch ist. An Portierungen für Windows XP und BSD wird gearbeitet.

Ein ähnlicher Ansatz wird von *Denali verfolgt*. Als Gast-Betriebssystem dient *Ilwaco*, eine speziell an den Denali-Hypervisor angepasste BSD-Version [46,47]. Denali unterstützt nicht das vollständige IA32 ABI. Es wurde für Netzwerk-Anwendungen entwickelt und unterstellt Einzelbenutzer-Anwendungen. Mehrfache virtuelle Adressräume sind unter Ilwaco nicht möglich.

Disco ist ein Hypervisor für einen ccNUMA-Cluster mit MIPS-R10000-Prozessoren und IRIX 5.3 als Gast-Betriebssystem. Auch die MIPS-Architektur gestattet keine vollständige Virtualisierung des virtuellen Adressenraums des Kernels. Ähnlich Xen wurde der IRIX-5.3-Gast-Kernel für einen Betrieb unter Disco leicht abgeändert [6].

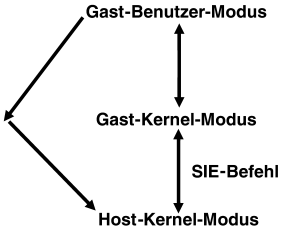

Abb. 6. Zustands-Übergänge

Plex86 ist ebenfalls ein Open-Source-Projekt. Als Gast-Betriebssystem wird ein abgeändertes Linux verwendet, welches z.B. keine Ein-/Ausgabe-Unterstützung enthält. Ein-/Ausgabe-Operationen werden von einem Hardware Abstraction Layer direkt an den Plex86-Hypervisor weitergereicht [28].

4 Interpretive Execution Facility

4.1 Architekturerweiterung

Die unterschiedlichen Entwicklungsstufen der S/360-, S/370-, S/390- und zSeries-Architektur und der entsprechenden Betriebssysteme sind in [22], Kapitel 2 (Hardware) und Kapitel 4 (Betriebssysteme) beschrieben.

Die Entwicklung der VM-Betriebssysteme begann 1967 mit der Ankündigung von CP/67 und führte geradlinig zu den heutigen Produkten z/VM und PR/SM. Alle Modelle sind virtualisierbar, die sensitiven Maschinenbefehle sind gleichzeitig privilegiert. Im Vergleich zu VM/370 ermöglichen zusätzliche Einrichtungen der Hardware-Architektur eine wesentliche Verbesserung des Leistungsverhaltens. In der zSeries- und S/390-Architektur sind diese Einrichtungen in der *Interpretive Execution Facility* zusammengefasst.

Normalerweise unterscheidet eine Architektur zwischen Kernel- und User-Modus. Zusätzlich führt die Interpretive Execution Facility eine weitere Trennung ein, die Einteilung in Host- und Gast-Modus. Der Host-Kernel startet den Gast unter Verwendung des Maschinenbefehls SIE (Start Interpretive Execution). Dabei wird die CPU vom Host-Modus in den Gast-Modus umgeschaltet (Abb. 6), der aber seinerseits Benutzer- und Kernel-Modus kennt.

Mit der Interpretive Execution Facility sind zusätzliche Hardware-Einrichtungen verfügbar. Hierzu gehören ein eigenes Gast-Program-Status-Wort, Zeitgeber, sowie ein zweiter Satz der Kontroll-Register. Diese Register werden von der Gast-Maschine benutzt. (In Wirklichkeit existiert noch ein dritter Satz, der es einer Gast-Maschine ermöglicht, ihrerseits als Host für eine Gruppe von Gast-Maschinen zu dienen. Hierauf wird an dieser Stelle nicht näher eingegangen.) Damit können die meisten privilegierten Maschinenbefehle des Gast-Kernels innerhalb des Gast-Modus ausgeführt werden. Maschinenbefehle oder Zustände, die weiterhin die Unterstützung durch den Host-Kernel erfordern, führen zu einer *Interception*. Diese beendet den SIE-Maschinenbefehl. Die CPU befindet sich wieder im Host-Modus und der Host-Kernel führt den auf SIE folgenden nächsten sequenziellen Maschinenbefehl aus.

Der Operand des SIE-Maschinenbefehls (Basis Register Inhalt plus Displacement) zeigt auf einen 512 Byte großen

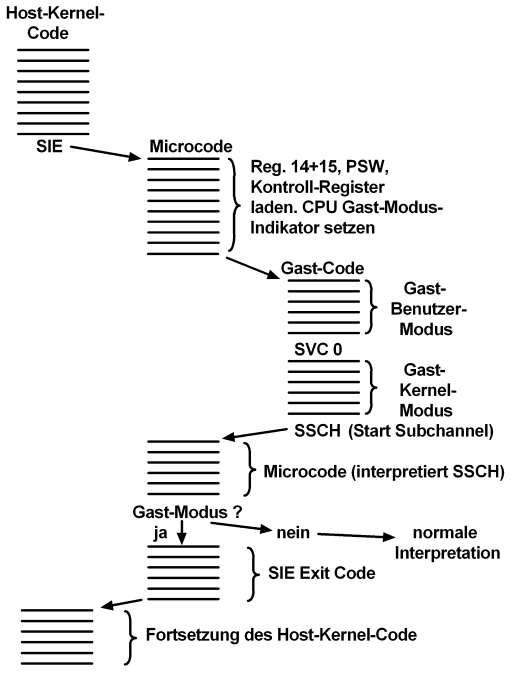

Abb. 7. Ablauf einer SIE Instruktion

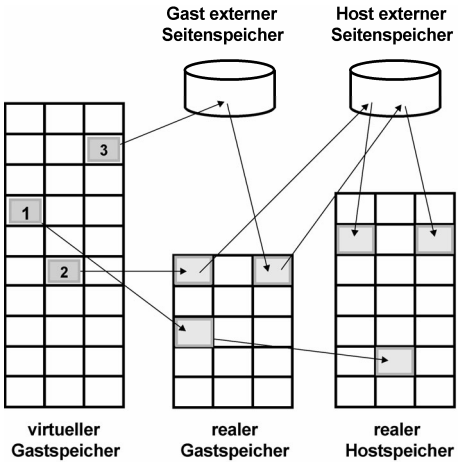

Abb. 8. Adressumsetzung der Gast-Maschine

State Descriptor, den der Host-Kernel für jede Gast-Maschine innerhalb des Hauptspeichers unterhält. Dieser State Descriptor speichert eine Beschreibung des Zustandes einer Gast-Maschine, wie Hauptspeicher, Zeitgeber u.a. Ein vollständiger Satz Kontroll-Register sowie ein Teil der Mehrzweck-Register (General Purpose Register, GPR, [19, Abb. 2.2.1]) des Gastes sind ebenfalls im State Descriptor abgebildet. Diese Register des Gastes werden beim Aufruf von SIE automatisch geladen und bei Beendigung von SIE in den State Descriptor zurückgeschrieben. Alle anderen Register des Gastes muss der Host-Kernel explizit laden (und ggf. vorher seine eigenen Register speichern). Nach Beendigung von SIE muss er diese geänderten Register wieder abspeichern (und ggf. seine eigenen Register wieder laden).

SIE ist ein komplexer Maschinenbefehl, der mittels Firmware implementiert wird. Seine Ausführung kann ähnlich wie bei anderen lang laufenden Maschinenbefehlen (Beispiel Move Character Long, MVCL) unter bestimmten Bedingungen unterbrochen werden. Abb. 7 zeigt, wie die CPU nacheinander die verschiedenen Code-Ebenen und Modi durchläuft. Nach Abschluss des SIE-Befehls wird der im Host-Kernel unmittelbar folgende Befehl ausgeführt.

Das Umschalten vom Host-Modus in den Gast-Modus durch den SIE-Befehl ermöglicht es, die Ausführung einer Untermenge der privilegierten Maschinenbefehle direkt an den Gast-Kernel ohne Inanspruchnahme des Host-Kernels weiterzuleiten. Für welche der privilegierten Befehle dies gilt wird durch Einträge im State Descriptor spezifiziert. Ohne die Interpretive Execution Facility müssten alle privilegierten Maschinenbefehle durch den Host-Kernel interpretiert werden.

4.2 Seitenersetzung (Paging)

Unter z/VM unterliegt der Gast-Speicher der Seitenersetzung durch den Host-Kernel. Dadurch kann der Host-Kernel im Prinzip beliebig viele Gast-Systeme parallel betreiben. Handelt es sich bei dem Gast um ein Betriebssystem, das seinerseits virtuelle Speicher und Seitenersetzung unterstützt, so ist beim Speicherzugriff eine zwei- oder mehrstufige Adressumsetzung erforderlich: Zunächst wird unter Verwendung der Gast-Tabellen die virtuelle Adresse des Gastes in eine vom Gast als real angesehene Adresse umgesetzt. Diese ist vom Blickwinkel des Hosts wieder eine virtuelle Adresse, die anhand der Host-Tabellen auf eine reale Adresse des Host umgesetzt wird. (Die Unterscheidung zwischen realen und absoluten Adressen der zSeries-Architektur wird hier der Einfachheit halber vernachlässigt, siehe [30, Fig. 3-5]).

Die Adressumsetzung der virtuellen in reale Adressen erfolgt durch eine mikroprogrammierte Address Translation Engine als Teil der Hardware der CPU. Im Host-Modus setzt diese anhand von Segment- und Seitentabellen virtuelle in reale Adressen um. Ein Gast kann, wie schon erwähnt, seinerseits virtuellen Speicher verwalten, wodurch eine zusätzliche Adressübersetzung erforderlich wird. Im Gegensatz zu IA32 und den früheren S/370-Systemen verfügen S/390- und zSeries-Systeme über Hardware, um diese Adressumsetzung vollständig durchzuführen. Die Adressumsetzung kann damit im Gast-Modus anders als im Host-Modus erfolgen. Sie benutzt das in Abb. 3 dargestellte Verfahren, ohne Verwendung von Shadow Tables und damit ohne Eingriff des Host-Kernels.

Wir bezeichnen als virtuellen Gastspeicher den Adressenraum, den der Gast als virtuell betrachtet. Der reale Gastspeicher wird vom Gast als real angesehen, ist aber in Wirklichkeit ebenfalls virtuell. Der reale Hostspeicher ist der tatsächliche reale Speicher, wie ihn der Host sieht. In Abb. 8 sind die unterschiedlichen Adressräume dargestellt. Die Adressen im virtuellen Gastspeicher werden durch die Address Translation Engine mittels der Segment- und Seitentabellen des Gastes in reale Gastspeicheradressen umgewandelt. Unter Verwendung der Segment- und Seitentabellen des Hosts werden diese wiederum auf reale Hostadressen umgesetzt, mit denen der eigentliche Speicherzugriff erfolgt.

Seite 1 befindet sich im realen Gastspeicher und auch im realen Hostspeicher. Es erfolgt keine Seitenersetzung (paging). Seite 2 befindet sich im realen Gastspeicher, aber nicht im realen Hostspeicher. Dieser Fall kann z.B. auftreten, wenn nach Zeitscheibenende des Gastes diese Seite vom Host auf

den externen Seitenspeicher ausgelagert wird. Eine Fehlseitenunterbrechung tritt auf, die vom Host-Kernel abgearbeitet werden muss. Seite 3 befindet sich nicht im realen Gastspeicher. Die Fehlseitenunterbrechung muss vom Gast-Kernel abgearbeitet werden. Wenn der Gast diese Seite von einer Platte liest, wird sie in den realen Hostspeicher geholt.

E/A-Geräte arbeiten nur mit realen Adressen, daher müssen alle E/A-Puffer zumindest für die Dauer der E/A-Operation im realen Hostspeicher vorhanden sein.

All dies bedeutet zusätzlich Hauptspeicherzugriffe. Dieser potentielle Leistungsabfall wird effektiv durch besonders große Adressumsetzpuffer aufgefangen. Die zSeries-Rechner haben hierfür einen 2-stufigen TLB, wobei beide Stufen 4-Wege-assoziativ sind. Die erste Stufe hat 4 x 128 Einträge, die zweite Stufe 4×1024 Einträge. Diese Einträge bleiben möglicherweise in dem Zeitraum zwischen Zeitscheibenende einer virtuellen Maschine und dem nächsten Zeitscheibenanfang erhalten. Hierzu wird im TLB zusätzlich die Adresse des SIE State Descriptors festgehalten.

5 PR/SM und LPARs

5.1 Licensed Internal Code

Ein PC verfügt über ein BIOS. Letzteres implementiert Maschinencode, der in einem Teil des Hauptspeichers liegt, auf den ein Benutzerprozess nicht unmittelbar zugreifen kann. Eine äquivalente zSeries- oder S/390-Funktion heißt *LIC (Licensed Internal Code*). Auch hierauf kann ein *B*enutzerprozess nicht direkt zugreifen. LIC wird beim Hochfahren eines Rechners durch einen als *IML (Initial Microcode Load)* bezeichneten Vorgang vom Plattenspeicher in den Hauptspeicher geladen [36, Abschnitt 7.2]. Der PALcode (Privileged Architecture Library code) der Alpha-Architektur hat eine ähnliche Funktion (siehe [32], Kapitel 6).

Die in Abschnitt 5.2 beschriebene PR/SM-Einrichtung ist ebenfalls ein Teil von LIC.

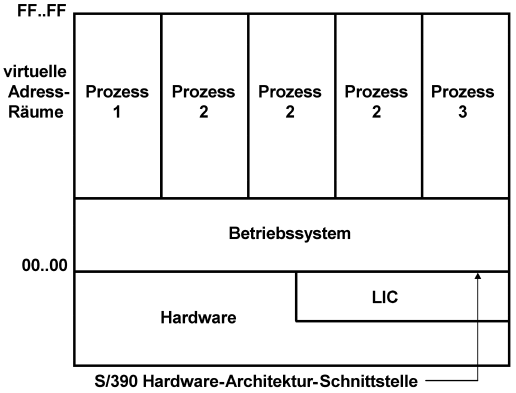

Abb. 9. zSeries- oder S/390-Struktur

LIC-Funktionen werden im Wesentlichen mit Hilfe regulärer Maschinenbefehle verwirklicht. Im Rahmen der zSeries- und S/390-Architektur gelten sie als Teil der Hardware. Sie sind, wie in Abb. 9 dargestellt, in einem Speicherbereich untergebracht, den die Software (Betriebssystem und Benutzerprozesse) nicht adressieren kann.

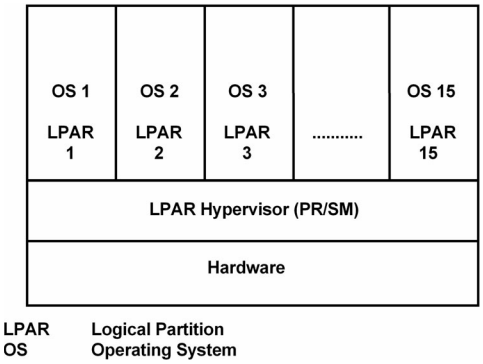

Abb. 10. IBM PR/SM und LPAR

In der zSeries-Architektur werden Ein-/Ausgabe-Einheiten, besonders Plattenspeicher, über Kanäle (channel paths) angeschlossen. Ein Kanal ist grob mit einem SCSI-Bus vergleichbar. Während es möglich ist, ein Ein-/Ausgabegerät direkt an einen Kanal anzuschließen, wird in der Regel an den Kanal eine Steuereinheit (control unit) angeschlossen, mit der wiederum zahlreiche Ein-/Ausgabe-Geräte, z.B. Plattenspeicher, verbunden sind. Normalerweise ist eine Steuereinheit über mehr als einen Kanal mit der Zentraleinheit verbunden und ein bestimmter Plattenspeicher kann in der Regel über mehr als einen Kanal angesprochen werden.

Die Verbindung der Kanäle mit einem zSeries-Rechner erfolgt über ein *Kanal-Subsystem* (channel subsystem). Das Kanal-Subsystem kann bis zu 256 Kanäle betreiben; ein z990-Rechner verfügt über 1-4 Kanal-Subsysteme. Die Steuerung eines Kanal-Subsystems erfolgt durch einen oder mehrere System Assist Prozessoren (SAP). Ein SAP ist ein zSeries- oder S/390-Prozessor-Chip, auf dem ausschließlich Licensed Internal Code (LIC) abläuft. SAPs entlasten die CPUs während der Ausführung einer Ein-/Ausgabe-Operation. Einzelheiten sind in [19, Abschnitt 3.2 und Abb. 3.2.7], in [27, Abschnitt 4.1], sowie in [41] beschrieben.

5.2 Basis-Konfiguration

PR/SM (Processor Resource/System Manager) ist ein mit Hilfe von Licensed Internal Code (LIC) implementierter Hypervisor. Er ist ein Bestandteil der zSeries- und S/390-Architektur und erlaubt die Partitionierung eines physischen Rechners in mehrere virtuelle Maschinen, die jeweils in einer *logischen Partition* (logical partition, LPAR) laufen. Jede virtuelle Maschine hat ein eigenes Betriebssystem, einen eigenen, unabhängigen realen Hauptspeicherbereich und eigene Kanäle sowie Ein-/Ausgabe-Geräte. PR/SM steuert die einzelnen Betriebssysteme auf den verschiedenen CPUs eines symmetrischen Multiprozessors (SMP). Jede Partition kann nur die Ressourcen benutzen, die ihr zugeordnet sind. (Zusätzliche Einrichtungen ermöglichen eine gemeinsame Nutzung von Kanälen oder Ein-/Ausgabe-Geräten durch mehrere LPARs.)

Historisch ist PR/SM aus VM hervorgegangen. S/390-Rechner anderer Hersteller verfügen über mit PR/SM vergleichbare Einrichtungen; Beispiele sind die Hitachi MLPF oder die Amdahl Multiple Domain Facility. Abb. 10 zeigt, wie von PR/SM 15 logische Partitionen verwaltet werden, in denen 15 unterschiedliche Betriebssystem-Instanzen laufen.

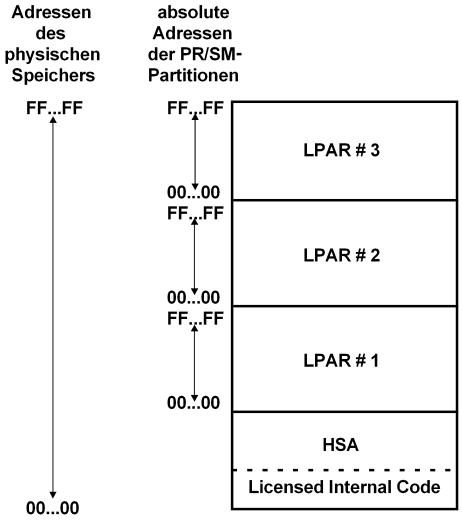

Abb. 11. PR/SM- und LPAR-Speicheraufteilung des physischen Speichers

Eine logische Partition (LPAR) enthält folgende Einheiten:

- eine oder mehrere CPUs (oder aber auch nur Bruchteile einer im *Time-Sharing*-Verfahren genutzten CPU),
- einen als *Zone* bezeichneten Teil des physischen Hauptspeichers,
- Kanäle (channel paths).

Es können maximal 30 LPARs definiert werden. Ressourcen wie Kanäle und Hauptspeicher können im laufenden Betrieb den LPARs dynamisch zugeordnet werden; eine Abgabe durch das Gast-Betriebssystem ist ebenfalls möglich. Außerdem kann eine LPAR als *Coupling Facility* definiert werden. Die Coupling Facility ermöglicht einen als *Sysplex* bezeichneten Cluster mit mehreren hundert CPUs und sehr guten Skalierungseigenschaften. Einzelheiten sind in [37] beschrieben.

5.3 Speicherverwaltung

Normalerweise laufen virtuelle Maschinen in einem virtuellen Adressenraum, der von einem Hypervisor verwaltet wird. Im Gegensatz dazu laufen PR/SM-Gast-Maschinen in einem realen Adressenraum. Damit entfällt die in Abb. 3 dargestellte doppelte Adressübersetzung.

Wir bezeichnen den mit Hilfe von Siliziumchips implementierten Hauptspeicher als den physischen Speicher. Eine Aufteilung des Adressenraums des physischen Speichers in mehrere reale Adressenräume für die einzelnen LPARs ist in Abb. 11 wiedergegeben. Ein weiterer Teil (Hardware System Area, HSA) speichert Konfigurationsdateien sowie den Licensed Internal Code (LIC).

Der einer spezifischen LPAR zugewiesene Teil des physischen Speichers wird als *Zone* bezeichnet und mit den realen Adressen einer Gast-Maschine direkt angesprochen. Der reale Adressenraum beginnt mit der Adresse Hex 00...00 und hat einen zusammenhängenden (contiguous) Namensraum. Die realen Adressen einer jeden LPAR werden wie in Abb. 12

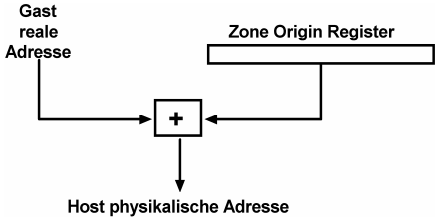

Abb. 12. LPAR Adressumsetzung

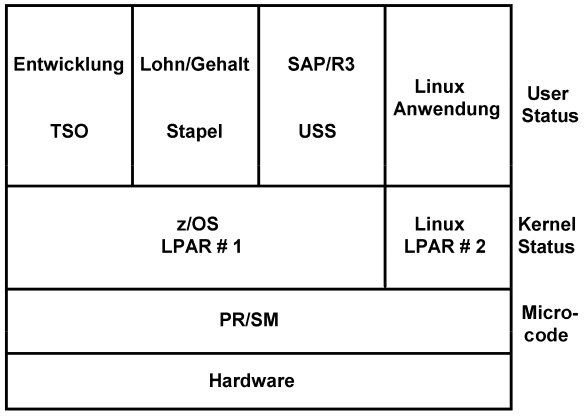

Abb. 13. PR/SM und LPAR

gezeigt in die Adressen des physischen Hauptspeichers umgesetzt (linear verschoben). Die Umsetzung erfolgt mit Hilfe eines Registers, hier als *Zone Origin Register* bezeichnet, sowie eines *Zone Limit Registers*, welches die Größe des zugeordneten physischen Adressenbereiches überprüft. Für eine Beschreibung dieses als *Relocation* bezeichneten Verfahrens siehe [17, Abschnitt 5.4.3]. Die Zone Origin und Zone Limit Register sind für jede LPAR einmal vorhanden.

Es ist möglich, die Größe der Zone dynamisch zu ändern. Hierfür ist eine zusätzliche einstufige Adressumsetzung vorhanden, die den realen Speicher der LPAR in Blöcken von 64 MByte auf einen nicht zusammenhängenden Teil des physischen Speichers abbildet.

Kanäle sind in der Regel den LPARs fest zugeordnet. zSeries-Kanäle, wie auch die SCSI-Anschüsse in anderen Architekturen, arbeiten immer mit realen Hauptspeicheradressen. Deshalb verwenden alle zSeries-Kanäle ebenfalls Zone Origin und Zone Limit Register.

Abbildung 13 zeigt, wie auf einem System mit zwei LPARs sowohl z/OS als auch z/Linux als Gast-Betriebssysteme laufen. z/Linux, offiziell als *Linux on zSeries* bezeichnet, ist ein regulärer Port von Linux auf die zSeries- bzw. S/390-Plattform. Dieser Code läuft sowohl unmittelbar auf der Hardware als auch in einer LPAR.

Weiterhin ist es möglich, z/VM in einer LPAR als Host-Betriebssystem einzusetzen und darin wiederum Gast-Betriebssysteme zu betreiben („second level guest“). Für den Betrieb von Dutzenden oder Hunderten von Linux-Instanzen ist dies eine populäre Konfiguration. Dieser Ansatz wird in jüngster Zeit häufig benutzt, um eine größere Anzahl getrennter physischer Server auf einer einzigen zSeries-Plattform zu konsolidieren.

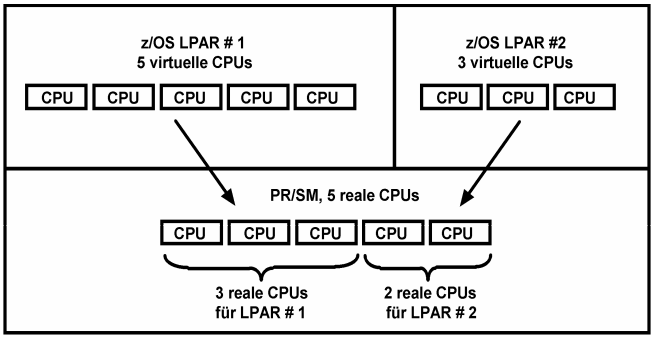

Abb. 14. Zuordnung von CPUs

5.4 Multiprocessing

Großrechner werden heute in der Regel als symmetrische Multiprozessoren ausgebildet. Ein Gast-Betriebssystem kann reale Multiprozessor-Hardware ausnutzen, indem es selbst im Multiprozessing-Status mit mehreren virtuellen CPUs arbeitet.

PR/SM verfügt wie ein normaler Betriebssystem-Kernel über einen Scheduler/Dispatcher, der den virtuellen CPUs Zeitscheiben zuordnet. Die Länge der Zeitscheiben ist variabel und wird dynamisch an die sich ändernden Bedingungen angepasst. Der Gast läuft bis zum Zeitscheibenende (im Millisekunden-Bereich) auf einer realen CPU. Unabhängig davon verwenden die Gast-Kernel ihre eigene Zeitscheibensteuerung, um die Prozesse des Gast-Betriebssystems zu steuern.

Im einfachsten Fall (*dedicated mode*) werden die realen CPUs den einzelnen LPARs fest zugeordnet. Als Beispiel sei ein Rechner mit fünf realen CPUs und zwei LPARs angenommen. Hier ist es z.B. möglich, von den fünf existierenden realen CPUs drei der LPAR 1 und zwei der LPAR 2 fest zuzuordnen.

Ein verbessertes Leistungsverhalten kann erreicht werden, wenn auf eine feste Zuordnung der realen CPUs verzichtet wird (*shared mode*). Hierzu ordnet die PR/SM Prioritätssteuerung den einzelnen LPARs *Gewichte* (weights) zu. Je nach Gewicht werden der einzelnen LPAR mehr oder weniger reale CPUs zugeordnet, wobei auch Bruchteile von CPUs möglich sind.

Beispielsweise erhält LPAR 1 ein Gewicht von 60% und LPAR 2 ein Gewicht von 40%, siehe Abb. 14. Damit erhält LPAR 1 eine Garantie, dass ihr mindestens 3 der 5 realen CPUs zur Verfügung stehen, während LPAR 2 eine Garantie für 2 reale CPUs erhält. Wenn eine der LPARs ihre CPUs nicht zu 100% auslastet, steht die restliche CPU-Kapazität der (den) anderen LPAR zur Verfügung. Um dies auszunutzen ist es möglich – und üblich –, dass die Anzahl der virtuellen CPUs des Gast-Betriebssystems größer ist als die Anzahl der realen CPUs, die der LPAR garantiert werden.

CPUs für eine LPAR sind entweder alle zugeordnet (dedicated mode) oder alle gemeinsam genutzt (shared mode). Bei mehreren LPARs können einige als dedicated und der Rest als shared deklariert werden. CPUs, die man als einer LPAR zugeordnet definiert, sind nicht verfügbar, um für andere aktive LPARs zu arbeiten. Die Ressourcen von gemeinsam benutzten CPUs bilden einen Pool und werden aktiven LPARs zugewiesen, wie sie benötigt werden. Eine Mischung aus LPARs mit *shared* CPUs und LPARs mit zugeordneten CPUs kann definiert und parallel aktiviert werden. Dies, zusammen mit der Zuordnung von Gewichten, geschieht durch den System-Administrator unter Benutzung der Hardware-Konsole. Änderungen können vom System-Administrator dynamisch durchgeführt werden, ohne den laufenden Betrieb zu stören.

5.5 Intelligent Resource Director (IRD)

Der in der z/Architektur implementierte *Intelligent Resource Director (IRD)* ist eine Erweiterung des PR/SM- und LPAR-Konzeptes. IRD übernimmt die Optimierung der Prozessor- und Kanal-Ressourcen über die verschiedenen logischen Partitionen (LPARs). Die drei wichtigsten Funktionen des IRD sind:

- *LPAR CPU-Management,*
- *Dynamic Channel Path Management* und
- *Channel Subsystem Priority Queuing*.

Die IRD-Funktionen werden vor allem von der *Work Load Manager*-Komponente (WLM) des z/OS- Betriebssystems genutzt (siehe [19, Abschnitt 5.7]). Sie greifen besonders in der als *Sysplex* bezeichneten zSeries-Cluster-Konfiguration [37].

Die IRD- und WLM-Funktionen optimieren die Nutzung von Ressourcen mit dem Ziel, den gesamten System-Durchsatz zu steigern. LPAR-CPU-Management wird von WLM benutzt, um die Gewichte und die Anzahl der zugeordneten CPUs dynamisch für alle z/OS-shared-mode-LPARs den sich ändernden Anforderungen anzupassen. Im Gegensatz zu der Darstellung in Abschnitt 5.4 geschieht dies automatisch, ohne Eingriffe des System-Administrators.

Kanäle sind ähnlich wie CPU-Zeit und Hauptspeicherplatz kritische Systemressourcen und werden im einfachsten Fall einzelnen LPARs fest zugeordnet. In vielen Fällen ist die Optimierung des Ein-/Ausgabe-Verkehrs ähnlich wichtig wie die optimale Nutzung der CPUs oder des Hauptspeichers. Die *Dynamic-Channel-Path-Management*-Komponente des IRD optimiert die Zuordnung der Kanäle zu den LPARs. Hierzu wird nur ein (kleinerer) Teil der Kanäle den LPARs fest zugeordnet (z.B. 2 pro LPAR); der (größere) Rest wird von WLM und IRD je nach Bedarf dynamisch zugeteilt.

Ein-/Ausgabe-Anforderungen werden in eine Warteschlange gestellt, ehe sie vom zSeries-Kanal-Subsystem abgearbeitet werden können. Dies geschieht normalerweise nach dem *first-in – first-out* Verfahren. Die *Channel-Subsystem-Priority-Queuing*-Komponente des IRD ermöglicht es dem WLM, die Reihenfolge der Abfertigungen zu optimieren.

5.6 Sicherheit

Virtuelle Maschinen eignen sich grundsätzlich, um die Sicherheit von Anwendungen zu verbessern. Beispielsweise existiert für die IA32-Architektur eine T*rusted Virtual Machine* mit dem Namen *Terra*. Einzelheiten sind in [14] enthalten.

Die PR/SM-Technologie hat darüber hinausgehende Sicherheitseigenschaften. Logische Partitionen werden durch den Hypervisor voneinander abgeschottet und können daher nicht wie die Prozessoren innerhalb eines Multiprozessorsystems miteinander kommunizieren. Insbesondere sind

die Hauptspeicherbereiche der Partitionen strikt getrennt und können sich nicht überlappen. Der Geltungsbereich für Unterbrechungen (z. B. E/A-Unterbrechungen) liegt ebenfalls nur innerhalb einer logischen Partition. Der Administrator des Gesamtsystems kann E/A-Verbindungen zwischen Partitionen konfigurieren, z. B. mit Hilfe von *Hipersockets*. Die E/A-Puffer werden jedoch durch die einzelnen Partitionen selbst festgelegt; sie behalten damit die volle Kontrolle über ihren eigenen Hauptspeicherbereich.

Das Konzept der Partitionierung gilt nicht nur für CPUs und Hauptspeicher, es setzt sich im Kanalsubsystem fort. Ist ein Kanal einer Partition fest zugeordnet, kann er von einer anderen Partition aus nicht angesprochen werden. Ist der Kanal so konfiguriert, dass er von mehreren Partitionen aus gleichzeitig benutzt werden kann, so sorgt er selbst für die Trennung der Aufträge aus den verschiedenen Partitionen.

PR/SM-LPARs haben entsprechend einer Zertifizierung der Regierung der Vereinigten Staaten die gleichen Sicherheitseigenschaften wie räumlich getrennte Rechner. In Deutschland wurde der zSeries-900 vom Bundesamt für Sicherheit der *EAL5 Certification Level* zuerkannt. Die zSeries ist derzeitig der einzige Server, der bisher auf dieser Ebene zertifiziert wurde [23].

5.7 Leistungsverhalten

Wie groß ist der Leistungsabfall, wenn ein Betriebssystem als virtuelle Maschine z.B. unter PR/SM läuft? Es ist sehr schwierig, diese Frage zu beantworten, weil die Art der Anwendung eine entscheidende Rolle spielt. Für rechenintensive Anwendungen, z. B. aus dem wissenschaftlichen Bereich, wird der Leistungsabfall geringer sein als bei Transaktions- und Datenbankanwendungen mit einem großen Aufkommen an Ein-/Ausgabe-Operationen. Leistungsmessungen ohne eine Zuordnung zum erwarteten Einsatzgebiet sind deshalb wenig aussagekräftig.

Die z/VM- und PR/SM-Entwicklungsteams verfügt über umfangreiche Untersuchungen des Leistungsverhaltens, welche die tatsächliche Nutzungen möglichst naturgetreu widerspiegeln. Die Ergebnisse werden für die Weiterentwicklung von z/VM und PR/SM genutzt, sind aber nicht veröffentlicht. Unter PR/SM hat eine virtuelle Maschine mit fest zugeordneten Ein-/Ausgabe-Einheiten eine Leistungseinbuße von 1–2%, verglichen mit einer realen Maschine. Da dieser Leistungsabfall vernachlässigbar ist, werden heute nahezu alle größeren OS/390-und z/OS-Rechner mit PR/SM betrieben. Beim neuesten Rechnermodell z990 ist ein Betrieb ohne PR/SM nicht mehr vorgesehen. Auch wenn nur ein einziges Betriebssystem installiert ist, läuft dies im Fall der z990 unter PR/SM in einer virtuellen Maschine.

6 Partitionierung

Moderne Großrechner werden heute als Cluster von symmetrischen Multiprozessoren (SMP) ausgebildet. Partitionierungseinrichtungen gehören hierbei zu den wichtigsten Eigenschaften. Unter Partitionierung verstehen wir die Eigenschaft, einen Server in kleinere Segmente (*Partitionen*) aufzuteilen, wobei

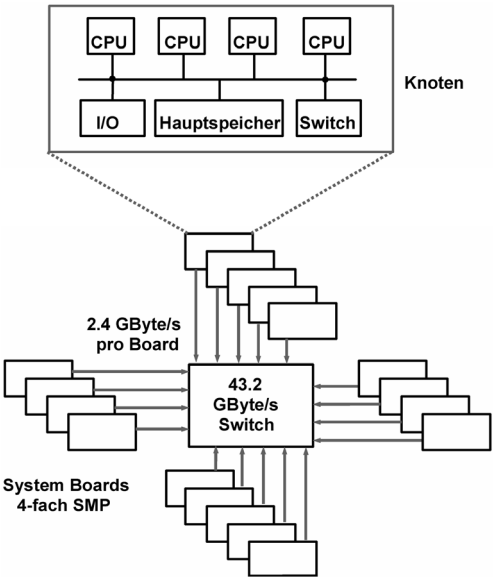

Abb. 15. Sun Fire 15K-Cluster

jede Partition ein eigenes Betriebssystem enthält. Ein moderner Trend besteht darin, mehrere physische Unix-Server zu einem einzigen Unix-Cluster mit getrennten Partitionen zusammenzufassen (Server-Konsolidierung). Damit ist es möglich,

- eine bessere Auslastung der Ressourcen mit Hilfe eines Workload Managers zu erreichen,
- das Systems Management zu vereinfachen,
- die Total Cost of Ownership (TCO) zu verringern und
- den Flächenbedarf und den Energieverbrauch zu reduzieren.

In den Partitionen können unterschiedliche Betriebssysteme, Anwendungen und Arbeitslasten ablaufen. Leichtgewichtige Internet-Anwendungen wie WWW-Dienste können mit schwerergewichtigen Anwendungen, z.B. datenbankorientierten Transaktionen, auf dem gleichen Server parallel laufen. Die durchschnittliche CPU-Auslastung von Unix-Servern beträgt heute in vielen Fällen 15-20% [**?**].

Partitionierungseinrichtungen lassen sich in zwei Klassen gliedern. Bei der physischen Partitionierung – der am häufigsten anzutreffenden Art – sind die Partitionierungsgrenzen mit den Baugruppen (z.B. Prozessor-Boards) identisch, aus denen der Server besteht. Ein typisches Beispiel ist der Sun Fire 15K-Cluster [38]. Er besteht aus 72 CPUs auf 18 als *System Boards* bezeichneten Knoten mit je vier UltraSPARC-III-1,2 GHz-Prozessoren pro Knoten. Abb. 15 zeigt die Konfiguration. Der Hewlett-Packard Superdome-Rechner hat eine sehr ähnliche Struktur.

Auf jedem System Board befinden sich Anschlüsse für den zentralen Switch. Weiterhin enthält das System Board E/A-Controller, die eine Verbindung zu getrennten E/A-Boards herstellen. Auf den E/A-Boards ist ein PCI-Bus implementiert, der SCSI-Adapterkarten für den Anschluss von Plattenspeichern aufnehmen kann. Dies kann z.B. der serielle Fibre Channel SCSI-Anschluss sein.

Jedes System Board stellt einen symmetrischen Multiprozessor (SMP) dar. Die CPUs eines System Boards können auf den Hauptspeicher eines anderen System Boards zugreifen. Mehrere System Boards können zu einem einzigen SMP

zusammengeschlossen werden. Bei Transaktions- und Datenbankanwendungen wird ein SMP jedoch selten aus mehr als 8-12 CPUs bestehen. Die 18 System-Boards eines Sun 15K-Rechners werden deshalb in Gruppen aufgeteilt. Jede Gruppe stellt eine Partition dar, wobei in jeder Partition ein SMP mit einer einzigen Betriebssysteminstanz läuft. Einer Partition sind die Ein-/Ausgabe-Anschlüsse ihrer System Boards fest zugeordnet.

Die Arbeitslast für den Rechner besteht aus unterschiedlichen Klassen von Anwendungen, z.B. Stapelverarbeitung, interaktive Anwendungen und Web-Anwendungen. Eine Lastverteilung (Workload Manager) verteilt diese Anwendungen auf die einzelnen Partitionen. Diese Zuordnung kann statisch erfolgen.

Mit Hilfe eines als *dynamische Rekonfiguration* bezeichneten Verfahrens ist es möglich, eine kontinuierliche Anpassung an die sich ständig ändernden Belastung und vor allem auch an kurzfristig auftretende Leistungsspitzen vorzunehmen [39]. Dies geschieht dadurch, dass ein vollständiges System Board einschliesslich seines Hauptspeichers und seiner Ein-/Ausgabe-Anschlüsse aus einer Partition entfernt und einer anderen Partition zugeordnet wird. Dabei sind diejenigen System Boards, in deren Hauptspeicher ein Teil der Betriebssysteminstanz untergebracht ist, von der dynamischen Rekonfiguration ausgeschlossen.

Diese Art der Rekonfiguration erfolgt mit einer Abstufung (Granularität) von System Boards. Eine sehr viel feinere Abstufung ist mit Hilfe der logischen Partitionierung (in der Literatur häufig als virtuelle Partitionierung bezeichnet) möglich, wie sie von z/VM und PR/SM verfügbar gemacht wird. Spezifisch können CPU-Ressourcen, Hauptspeicherplatz und Ein-/Ausgabe-Kanäle mit beliebig feiner Granularität den einzelnen Partitionen dynamisch zugeordnet werden. Ähnlich wird VMware GSX auf den Unisys ES7000-Enterprise-Servern für diesen Zweck eingesetzt. Der Unisys ES7000-Server besteht aus bis zu 32 IA32 CPUs. Auch HP plant für die Partitionierung des Superdome den zusätzlichen Einsatz von VMware GSX.

7 Ausblick

Heutige Rechnerarchitekturen werden für den Betrieb von virtuellen Maschinen als virtualisierbar oder als nicht-virtualisierbar klassifiziert. Bei den nicht-virtualisierbaren Architekturen verursacht die Behandlung von sensitiven Maschinenbefehlen erhebliche Probleme. Die Diskussion dieser Schwierigkeiten im Zusammenhang mit der IA32-Architektur ist ein dominierendes Thema in den wissenschaftlichen Veröffentlichungen aus jüngster Zeit.

Wir vermuten, dass dieses Problem an der Wurzel angepackt werden muss und dass die Zukunft deshalb den virtualisierbaren Rechnerarchitekturen gehört. Dabei halten wir Erweiterungen für denkbar, welche existierende Architekturen, wie z.B. IA32, virtualisierbar machen. Die Erweiterung um einen Host/Gast-Modus, um zusätzliche Adressübersetzungsmechanismen sowie um eine HAL-(Alpha) bzw. LIC-(zSeries) ähnliche Funktionalität, erscheint prinzipiell möglich. Hierfür werden Ansätze des Hardware/Software-Co-Designs zweckmäßig sein, wie sie z.B. in [25] diskutiert werden.

Virtuelle Maschinen und virtuelle Partitionierung stellen eine wichtige zukunftsorientierte Technologie dar. Wir glauben, dass die in z/VM und PR/SM implementierten Hardware- und Softwareeigenschaften einen Weg aufzeigen, wie die zukünftige Entwicklung auf dem Gebiet der virtuellen Maschinen aussehen könnte.

Acronyms

ABI	Application Binary Interface
CMS	Conversational Monitor System
COTS	Commercial Off-The-Shelf
CP	Control Program
CR	Kontroll-Register (Control Register)
GPR	Mehrzweck-Register (General Purpose Register)
HCD	Hardware Configuration Definition
HSA	Hardware System Area
IA32	32 Bit Intel Architektur (Pentium, 32 Bit AMD)
IML	Initial Microcode Load
IRD	Intelligent Resource Director
IA32	Intel Architecture, 32 Bit
IPL	Initial Program Load
LIC	Licensed Internal Code
LPAR	Logische Partition
PR/SM	Processor Resource/System Manager
SAP	System Assist Prozessor
SIE	Start Interpretive Execution
SMP	Symmetric Multiprocessor
SVC	Supervisor Call (System Call)
TLB	Adressumsetzpuffer (Translation Lookaside Buffer)
VM	Virtual Machine
VMM	Virtual Machine Monitor
WLM	Work Load Manager

Literatur

1. Amdahl G, Blaauw G, Brooks F (1964) Architecture of the IBM System/360. IBM J. Res. Devel. 8 (2), 87–101
2. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A (2003) Xen and the Art of Virtualization. Proceedings of the nineteenth ACM symposium on Operating systems principles, Bolton Landing, NY, (SOSP 2003), October 2003, pp 164–177, ISBN:1-58113-757-5
3. Bitner B (2002) z/VM Guest Performance. WAVV 2002, Fort Mitchell, Kentucky. ♣ Diese Literaturstelle wird im Text nicht erwähnt♣ http://www-1.ibm.com/servers/eserver/zseries/os/vse/pdf2/wavv02/wavvvmguest.pdf
4. Bochs User Manual http://bochs.sourceforge.net/cgi-bin/topper.pl?name=New+Bochs+Documentation&url=http://bochs.sourceforge.net/doc/docbook/user/book1.html
5. Borden TL, Hennessy JP, Rymarczyk JW (1989) Multiple operating systems on one processor complex. IBM Systems Journal 28(1):104–123 ♣ Diese Literaturstelle wird im Text nicht erwähnt♣
6. Bugnion E, Devine S, Govil K, Rosenblum M (1997) Disco: Running commodity operating systems on scalable multiprocessors. In: Proceedings of the 16th ACM SIGOPS Symposium on Operating Systems Principles, vol 31(5) of ACM Operating Systems Review, pp 143–156

7. Buzen JP, Gagliardi VO (1973) The Evolution of Virtual Machine Architecture. Proc National Computer Conference, S. 291–299
8. Short History of IBM's Virtual Machines. http://www.caplore.com/CP.html
9. Chapman M (2001) System Partitioning on IBM xSeries Servers. ftp://ftp.pc.ibm.com/pub/pccbbs/pc_servers_pdf/system_ partitioning.pdf
10. Creasy RJ (1981) The Origin of the VM/370 Time-Sharing System. IBM Journal of Research and Development 25(5), September
11. Jurassic Park mit Zukunft. Computer Zeitung, 19.5.2003, S. 1 http://www.computer-zeitung.de/O/50/Y/82807/VI/10043254/VS/Jurassic/default.aspx?O=50&Y=82807&VI=10043254&VS=JURASSIC
12. Ensim Extend. http://www.ensim.com/products/materials/ensim_virtual_private_server_data_sheet.pdf
13. System/390 on Intel-Based Servers, Flex-ES Technical Overview. http://www.funsoft.com/index-technical.html
14. Garfinkel T, Pfaff B, Chow J, Rosenblum M, Boneh D (2003) Terra: A Virtual Machine-Based Platform for Trusted Computing . Proceedings of the nineteenth ACM symposium on Operating systems principles, Bolton Landing, NY, USA, pp 193–206, ISBN:1-58113-757-5
15. Goldberg RP (1972) Architectural Principles for Virtual Computer Systems. PhDThesis, Harvard University, Cambridge. ♣ Diese Literaturstelle wird im Text nicht erwähnt♣ Also: Survey of Virtual Machine Research IEEE Computer 7(6), June 1974
16. Gum PH (1983) System/370 Extended Architecture: Facilities for Virtual Machines. IBM Journal of Research and Development 27(6), November
17. Herrmann P (1998) Rechnerarchitektur. Vieweg 1998, ISBN 3-528-05598-7
18. The Hercules System/370, ESA/390, and z/Architecture Emulator. http://www.conmicro.cx/hercules/
19. Herrmann P, Kebschull U, Spruth WG (2004) Einführung in z/OS und OS/390. Oldenbourg-Verlag, 2. Auflage, ISBN 3-486-27393-0
20. Honeywell Series 200. http://perso.club-internet.fr/febcm/english/honeywell_series_200.htm
21. Honeycutt J (2003) Microsoft Virtual PC Technical Overview, November 2003. http://www.microsoft.com/windowsxp/virtualpc/evaluation/techoverview.asp
22. Hoskins J, Frank B (2002) Exploring IBM e-server zSeries and S/390 Servers. Maximum Press, 7. Auflage, ISBN 1-885068-70-0
23. IBM eServer zSeries partitioning achieves highest certification. http://www-1.ibm.com/servers/eserver/zseries/security/certification.html
24. Croudy J (♣ Jahr?) The ICL 2900 Series. http://pink-mouse-productions.com/icl/2900.htm
25. Kent KB (2003) The Co-Design of Virtual Machines Using Reconfigurable Hardware. PH.D. Thesis, Dept. of Computer Science, University of Victoria. http://www.cs.unb.ca/∼ken/papers/dissertation.pdf
26. Lawton K (2000) Running multiple operating systems concurrently on an IA32 PC using virtualization techniques. http://denali.cs.washington.edu/relwork/papers/plex86.txt
27. Lehmann H, Spruth WG (2003) Eigenschaften einer modernen Ein-/Ausgabe Architektur. it – Information Technology, Volume 45, Issue 01, S. 20
28. The new Plex86 x86 Virtual Machine Project. http://plex86.sourceforge.net/
29. Popek GJ, Goldberg RP (1974) Formal Requirements for Virtualizable Third Generation Architectures. Comm ACM 17(7): 412–421
30. z/Architecture Principles of Operation. IBM Form No SA22-7832
31. Robin JS, Irvine CE (2000) Analysis of the Intel Pentium's ability to support a secure virtual machine monitor. In: Proceedings of the 9th USENIX Security Symposium, Denver, CO
32. Sites R (1992) Alpha Architecture Reference Manual. Digital press, ISBN 1-55558-098-x
33. Smith JE, Sastry S, Heil T, Bezenek T (1998) Achieving High Performance via Co-Designed Virtual Machines. International Workshop on Innovative Architecture, Maui High Performance Computer Center, Oct 26–28. http://www.engr.wisc.edu/ece/faculty/smith_james.html
34. Dhodapkar A, Smith JE (2001) Saving and Restoring Implementation Contexts with co-Designed Virtual Machines. Workshop on Complexity Effective Design, Gothenburg, Sweden, June 30. http://www.ece.wisc.edu/∼jes/papers/wced01.ashutosh.pdf
35. Spruth WG (1977) Interaktive Systeme. Science Research Associates (SRA, heute Oldenbourg Verlag), Stuttgart/München, ISBN 3-921439-15-9
36. Spruth WG (1989) The Design of a Microprocessor. Springer, Berlin, ISBN 3-540-51395-7
37. Spruth WG, Rahm E (2002) Sysplex-Cluster Technologien für Hochleistungs-Datenbanken. Datenbank-Spektrum, Heft 3, S. 16–26
38. Storer L (2001) New Sun Fire 15K server takes product line from $1,000 to $10 million. Serverworld. http://www.serverworldmagazine.com/monthly/2001/11/sunfire15k.shtml
39. Sun Fire 15K Dynamic Reconfiguration User Guide. Sun Product Documentation http://docs.sun.com/db/coll/dynrecon
40. Sugerman J et al (2001) Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor. Proceedings of the 2001 USENIX Annual Technical Conference Boston, June 25–30. www.usenix.org/events/usenix01/sugerman/sugerman.pdf
41. ABCs of OS/390 System Programming, Volume 1. IBM Form No SG24-5597-00, April 2000
42. Valluri A (♣ Jahr?) Sun builds on N1 vision. http://www.expresscomputeronline.com/20030512/newsan1.shtml
43. Virtuozzo White Paper. http://www.erexi.com.tw/whitepapers/virtuozzo_overview.pdf
44. Virtual PC for Mac. http://www.microsoft.com/mac/products/virtualpc/virtualpc.aspx
45. Waldspurger CA (2002) Memory Resource Management in VMware ESX Server. Proc Fifth Symposium on Operating Systems Design and Implementation (OSDI '02)
46. Whitaker A, Shaw M, Gribble SD (2002) Scale and performance in the Denali isolation kernel. In: Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), ACM Operating Systems Review, Winter 2002 Special Issue, pages 195–210, Boston, MA, USA
47. Whitaker A, Shaw M, Gribble SD (♣ Jahr?) Denali:Lightweight Virtual Machines for Distributed and Networked Applications. www.cs.ucsb.edu/∼rich/class/cs595-os/ denali_usenix2002.pdf

Ein Mirror für die in digitaler Form vorliegenden Literatur-Referenzen ist unter www-ti.informatik.uni-tuebingen.de/∼spruth/Mirror/partit/index.html eingerichtet

Joachim von Buttlar ist Chefberater in der Hardware-Entwicklung der IBM Deutschland Entwicklung GmbH. Nach Abschluss seines Studiums an der Technischen Universität Berlin begann er 1984 seine Tätigkeit bei IBM, wo er Firmware für das Channel Subsystem von S/390- und zSeries-Systemen entwickelt. Im Rahmen einer Auslandsabordnung arbeitete er von 1990 bis 1991 für IBM in Endicott, New York. In den letzten Jahren hat er eine hoch effiziente Simulationsumgebung für die Firmware zukünftiger zSeries-Systeme entwickelt.

Wilhelm G. Spruth ist Honorarprofessor am Schickard-Institut für Informatik der Universität Tübingen und am Institut für Informatik der Universität Leipzig. Sein Arbeitsgebiet umfasst Client/Serversysteme, Transaktionsverarbeitung und OS/390. Für die studentische Ausbildung steht ein OS/390-Rechner http://jedi.informatik.unileipzig.de zur Verfügung.