UNIVERSITÄT LEIPZIG

Mainframe Internet Integration

Prof. Dr. Martin Bogdan
Prof. Dr.-Ing. Wilhelm G. Spruth

SS2013

Virtualisierung Teil 1

Partitionierung

Skalierung eines Symmetrischen Multiprozessors

Ein Symmetrischer Multiprozessor (SMP) mit zwei statt einer CPU sollte idealer weise die zweifache Leistung haben. Dies ist aus mehreren Gründen nicht der Fall.

Die Gründe für den Leistungsabfall sind Zugriffskonflikte bei der Hardware und Zugriffskonflikte auf Komponenten des Überwachers. Zugriffskonflikte bei der Hardware sind weniger kritisch. Durch leistungsfähige interne Verbindungen wird eine hohe Bandbreite zwischen den einzelnen Prozessoren, dem Hauptspeicher und den I/O-Kanälen erreicht. Anders dagegen bei Zugriffskonflikte auf Komponenten des Überwachers. Besonders kritisch ist es, wenn mehrere CPUs eines SMP gleichzeitig einen Dienst des Betriebssystem-Kernels in Anspruch nehmen wollen, der nur seriell durchgeführt werden kann. Ein typisches Beispiel ist der Scheduler.

Zur Lösung wird der Kernel in Bereiche aufgeteilt, die mit Locks (Sperren) versehen werden, und die von mehreren CPUs unabhängig voneinander benutzt werden können. So kann z.B. der Kernel für eine CPU eine Fehlseitenunterbrechung bearbeiten, und parallel dazu für eine andere CPU eine I/O Operation durchführen. Damit ein gleichzeitiger Zugriff durch zwei CPUs möglich ist, werden die beiden Teile des Kernels durch Locks geschützt.

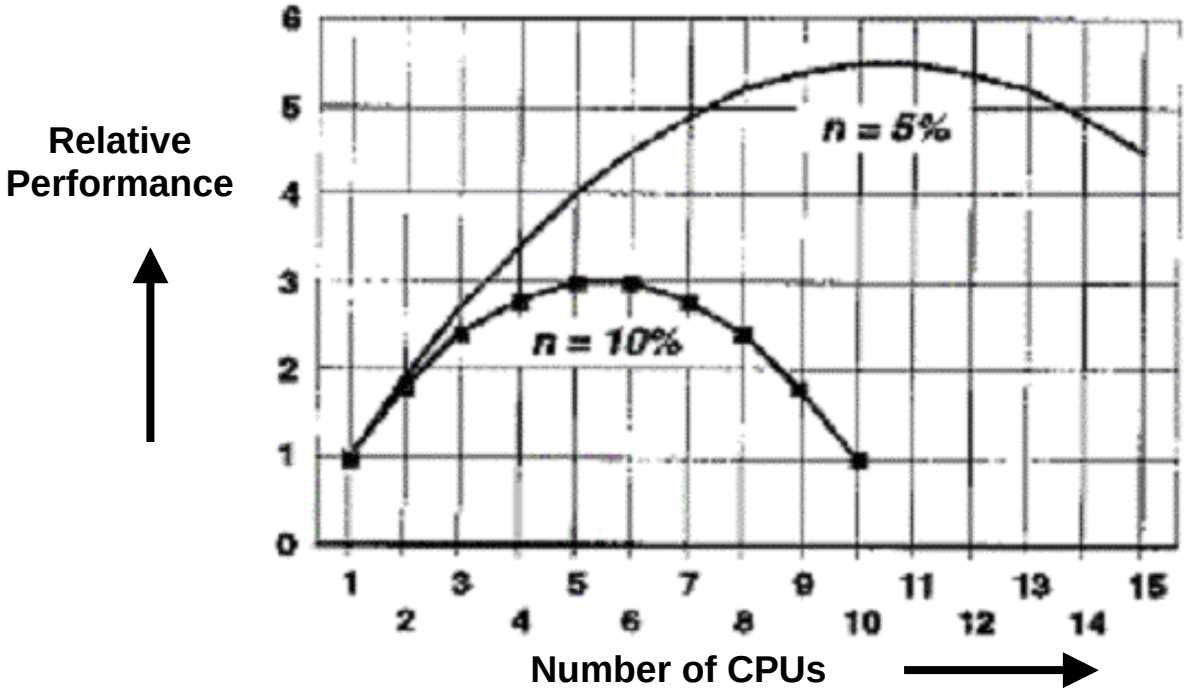
Gelegentlich muss eine CPU darauf warten. dass eine andere CPU eine Resource des Kernels freigibt. Um diese Wahrscheinlichkeit zu verringern, kann man den Kernel in eine möglichst große Anzahl von Teilbereichen gliedern, die durch Locks vor dem gleichzeitigen Zugriff durch mehrere CPUs geschützt werden. Teilt man jedoch den Kernel in zu viele Bereiche auf, verbringt er einen Großteil seiner Verarbeitungszeit mit der Lockverwaltung. Durch Feintuning und Abstimmung der einzelnen Kernel-Komponenten kann die Skalierung von SMP Rechnern verbessert werden. Mit wachsender Anzahl der CPUs verstärkt sich das Problem.

Wir haben in den vergangenen Jahrzehnten Fortschritte für SMP Rechner mit zunächst 2, dann 4, später 8, dann 12, 16 usw. CPUs gemacht, indem Wege gefunden wurden, den Kernel möglichst optimal in mehr und mehr Teile zu gliedern, die durch Locks von der gleichzeitigen Nutzung durch zwei oder mehr CPUs geschützt werden können, ohne dass der Aufwand für die Lock Verwaltung zu groß wird..

Verfahren für die Aufteilung des Kernels wurden mühsam während der letzten Jahrzehnte entwickelt, und werden immer noch weiter entwickelt. Eine stetige, aber kleine Performance Verbesserung wird so jedes Jahr erreicht.

Wie groß die Beeinträchtigung ist, hängt sehr stark von der Art der laufenden Anwendungen ab. Anwendungen, die Dienste des Kernels nur selten in Anspruch nehmen verursachen nur wenig Beeinträchtigungen. Ein Beispiel ist die Berechnung der Mandelbrot Menge (Apfelmännchen).

Bei den kritischen Transaktions- und Datenbankanwendungen skalieren die meisten SMP Betriebssysteme heute (2013) bis zu 12 – 16 CPUs. Als einzige Ausnahme skaliert z/OS bis zu 24 – 32 CPUs. Der Vorsprung rührt daher, dass die z/OS Entwickler 15 – 20 Jahre vor dem Rest der IT Industrie begonnen haben, eine möglichst optimale Aufteilung des Kernels in durch Locks geschützte Bereiche zu erforschen.



Die beiden dargestellten Kurven repräsentieren das typische Leistungsverhalten eines SMP als Funktion der Anzahl der eingesetztwen CPUs. Die untere Kurve nimmt an, dass bei 2 CPUs jede CPU 10 % ihrer Leistung verliert; bei der oberen Kurve sind es 5 %. Mit wachender Anzahl von CPUs wird der Leistungsgewinn immer kleiner; ab einer Grenze wird der Leistungsgewinn negativ.

Dis bedeutet, dass SMPs nur mit einer begrenzen maximalen CPUs sinnvoll betrieben werden können. Diese Grenze ist sehr stark von der Art der Anwendungen abhängig.

Angenommen, ein Zweifach Prozessor leistet das Zweifache minus n % eines Einfach Prozessors. Für n = 10 % ist es kaum sinnvoll, mehr als 4 Prozessoren einzusetzen. Für n = 5 % sind es 8 Prozessoren.

Bei einem EC12 Rechner mit z/OS ist n << 2 %. Es kann sinnvoll sein, einen SMP mit bis zu 32 Prozessoren einzusetzen. Meistens sind es weniger.

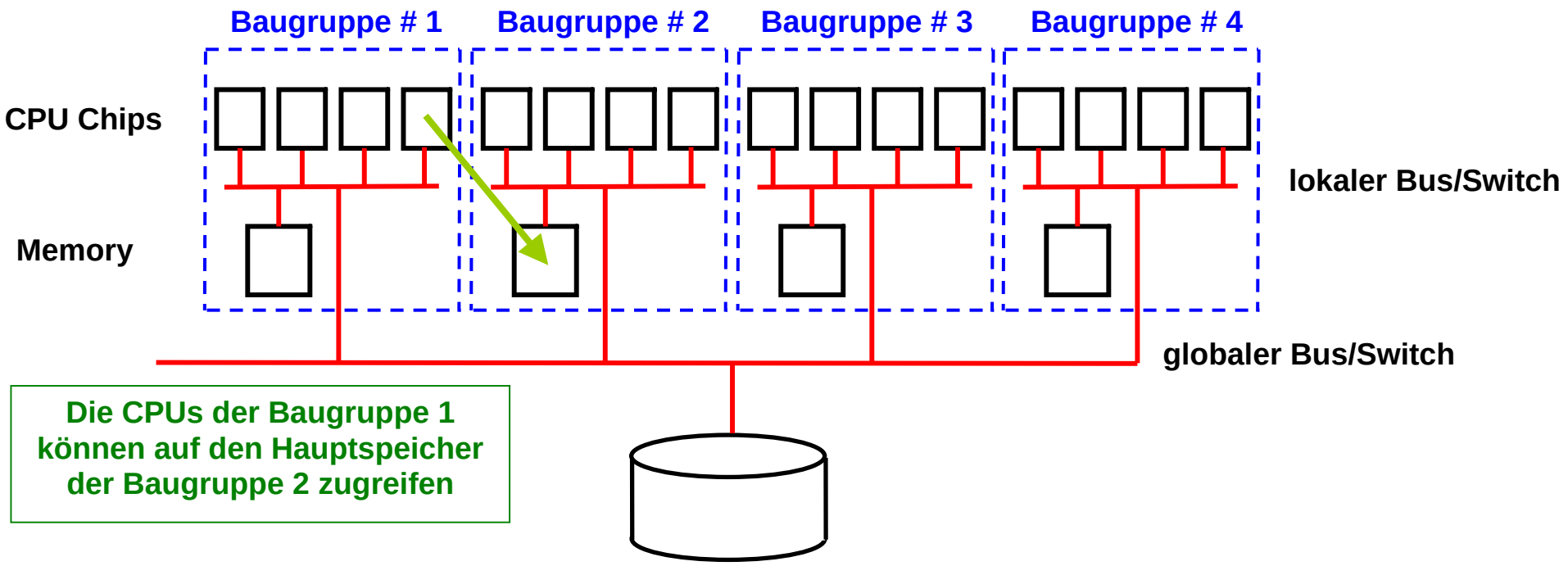
Warum Partitionierung

Wenn wir also einen Rechner mit 101 – 256 CPUs betreiben, muss dieser in mehrere unabhängige SMPs partitioniert werden, wobei jede Partition einen SMP darstellt. Die mehrfachen Partitionen können als Cluster (lose gekoppelt) betrieben werden und werden durch ein Hochgeschwindigkeitsnetzwerk (z.B. ein Crossbar Switch) miteinander verbunden.

Im einfachsten Fall geschieht die Partitionierung bei einem Sun 25K oder M9000 Rechner oder HP Superdome Server dadurch, dass mehrere Baugruppen (System Boards oder Cell Boards) einen SMP darstellen, und die zu getrennten SMPs gehörigen Baugruppen durch Schalter voneinander getrennt werden (sog. „harte“ Partitionen).

Die System Boards oder Cell Boards eines Sun 25k oder eines HP Superdome Rechners enthalten jeweils ihren eigenen Hauptspeicher. Wenn ein SMP aus mehr als einer Baugruppe besteht, muss eine SMP Betriebssystem Instanz auf den Hauptspeicher aller beteiligten Baugruppen zugreifen können.

Eine „Non-Uniform Memory Architektur“ (NUMA) macht dies möglich.

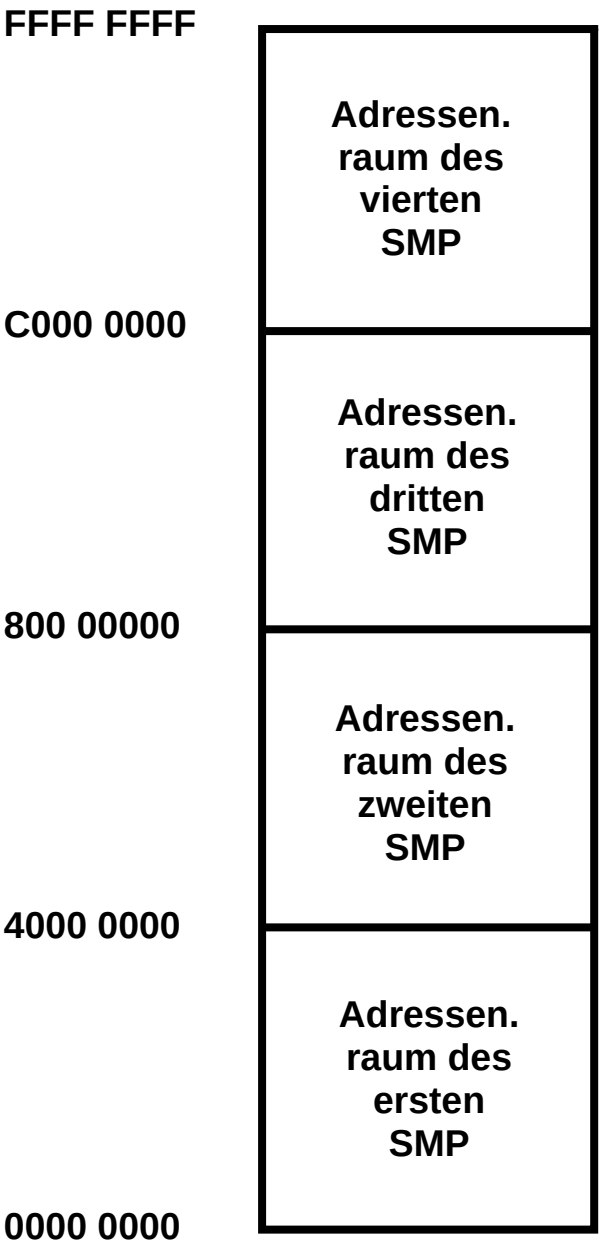


Non-Uniform Memory Architektur (NUMA)

Im einfachsten Fall partitioniert man den Rechner so, dass jede Baugruppe (System Board oder Cell Board) gleichzeitig ein SMP ist. Die CPUs einer Baugruppegreifen dann nur auf den Hauptspeicher der gleichen Baugruppe zu. Es ist aber auch möglich, dass eine CPU über den globalen Bus/Switch auf den Hauptspeicher einer fremden Baugruppe zugreift. Es ist damit möglich, dass zwei (oder mehr) Baugruppen einen einzigen SMP bilden. Ein Rechner mit dieser Einrichtung verfügt über eine „Non-uniform Memory Architektur“ (NUMA).

In dem gezeigten Beispiel besteht der Rechner (System) aus 4 Baugruppen (z.B. „System“ oder „Cell“ Boards). Jede Baugruppe enthält 4 CPU Sockel und einem von allen CPU Sockeln gemeinsam genutzten Hauptspeicher.

Frage: Mit welchen realen Hauptspeicheradressen arbeiten die einzelnen Baugruppen ?



NUMA Adressenraum

Hierzu ist es notwendig, dass die 4 Hauptspeicher der 4 Baugruppen mit unterschiedlichen realen Adressen arbeiten.

Dargestellt ist als Beispiel der Adressenraum eines NUMA Rechners mit 4 Baugruppen. Jeder SMP verfügt über einen Hauptspeicher von 4 GByte.

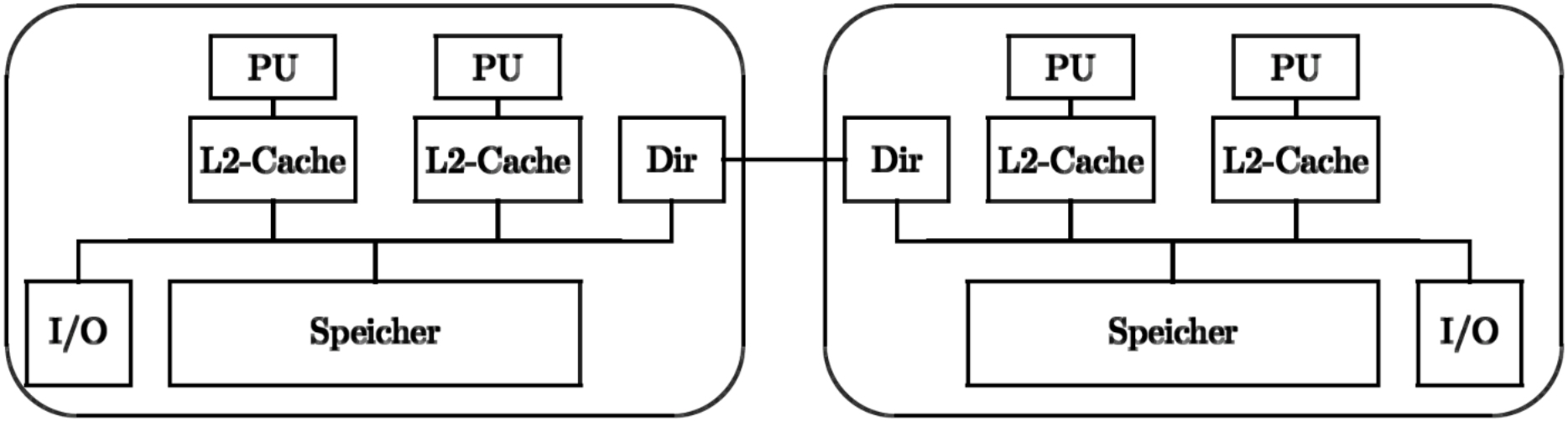
Der erste der 4 Baugruppen benutzt die Adressen Hex 0000 0000 bis 3FFF FFFF.

Der zeite der 4 Baugruppen benutzt die Adressen Hex 4000 0000 bis 7FFF FFFF.

Der dritte der 4 Baugruppen benutzt die Adressen Hex 8000 0000 bis BFFF FFFF.

Der vierte der 4 Baugruppen benutzt die Adressen Hex C000 0000 bis FFFF FFFF.

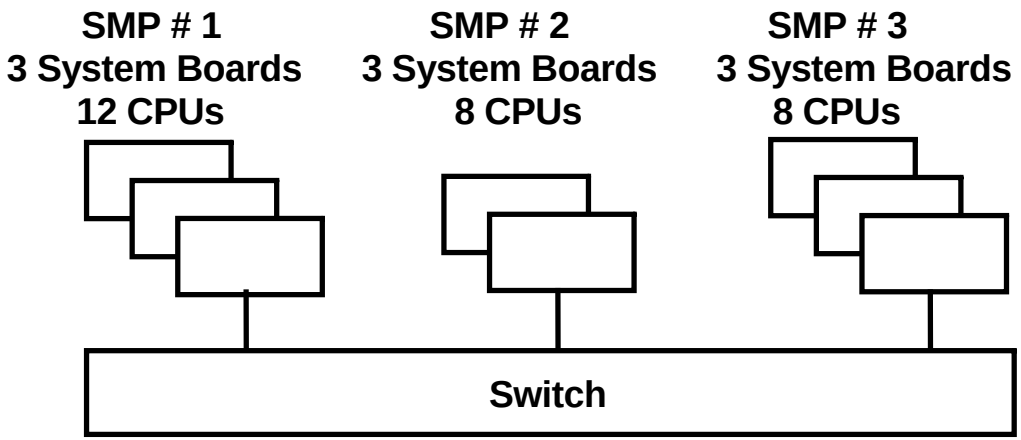
Da alle Prozesse der 4 Baugruppen mit virtuellen Adressen arbeiten, ist diese Aufteilung der realen Adressen problemlos möglich.



NUMA-Rechner

Die Hauptspeicher DIMMs aller Baugruppen bilden einen gemeinsamen physischen Speicher. Da jeder Prozessor auf die Speicher fremder SMPs zugreift, muss ein Verfahren für den Zugriff auf nicht-lokale Speicher existieren. Dies ist durch die Dir-Einheit (Directory) dargestellt. Diese enthälten Informationen über alle auf den externen SMPs befindlichen Speichersegmente und deren Inhalte.

Bei der Partitionierung eines Rechners in mehrere SMPs kann ein SMP immer nur auf einen Teil des physischen Speichers zugreifen. Zusätzliche Einrichtungen teilen den physischen Speicher in mehrere logische Speicher für die einzelnen SMPs auf.



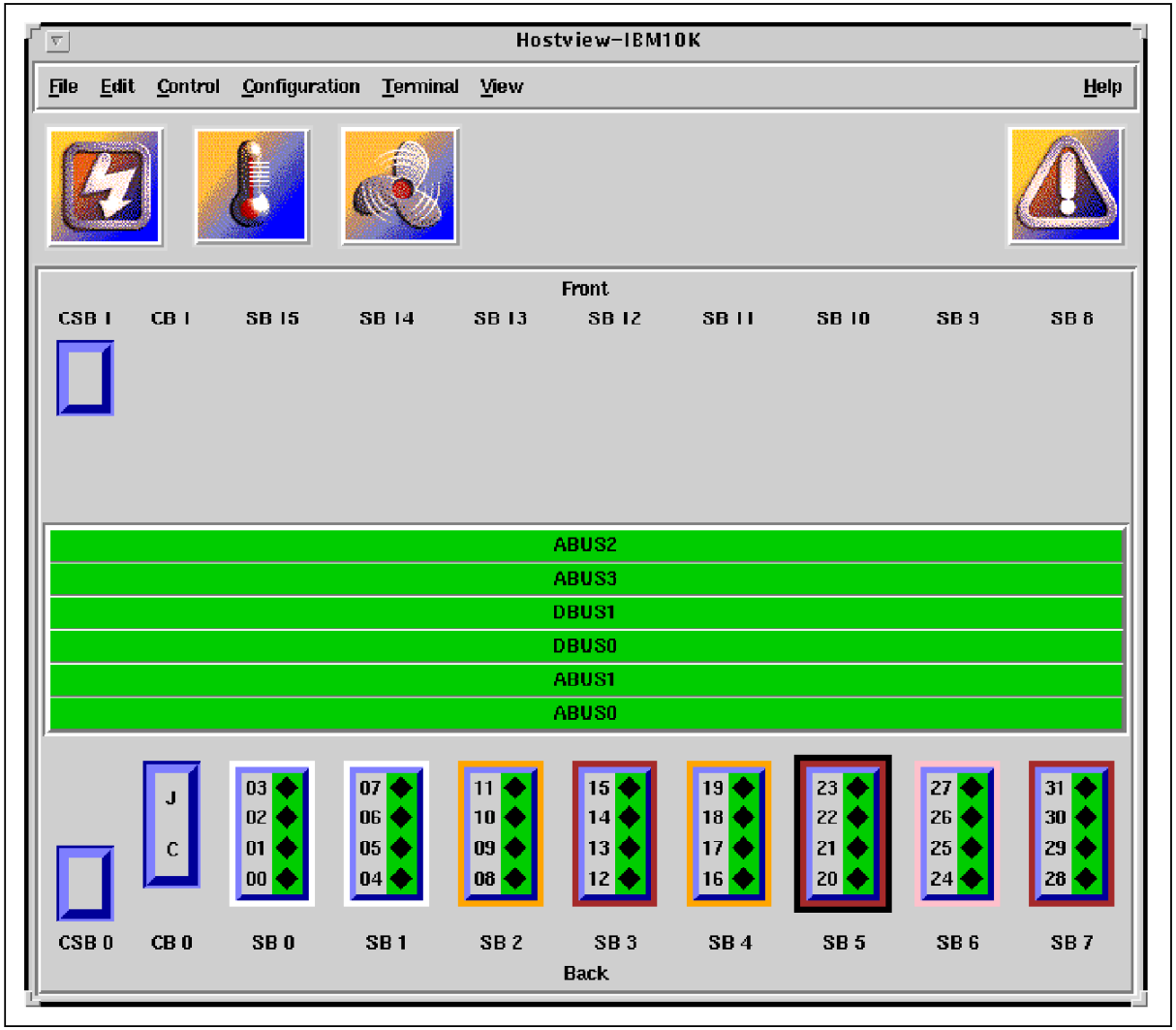
Harte Partitionierung

Als Beispiel ist die Aufteilung eines Sun 25K (oder HP Superdome) Servers mit 8 Baugruppen (System Boards) mit je 4 single Core CPU Chips in mehrere parallel laufende SMPs gezeigt.

Die 8 System Boards werden in 3 Partitionen und 3 SMPS mit 3, 2 und 3 System Boards aufgeteilt. Jeder SMP hat ein eigenes Betriebssystem. Die Aufteilung erfolgt statisch mit Hilfe von mechanischen oder elektronischen Schaltern.

Elektronische Schalter können während des laufenden Betriebes umkonfiguriert werden. Der System Administrator kann die Zuordnung der System Boards zu den einzelnen SMPs ändern

Für Transaktionsanwendungen unter Unix sind kaum mehr als 3 System Boards (12 CPU Cores) pro SMP realistisch.



Sun Fire Administrator Konsole

Dargestellt sind 8 System Boards SB0 .. SB7. Sie werden in 3 SMPs partitioniert:

SMP1 besteht aus: SB0, SB1, SB6
SMP2 besteht aus: SP2, SB4
SMP3 besteht aus: SP3, SB5, SB7

Angenommen, SMP2 ist überlastet, während SMP3 überschüssige Kapazität hat.

Der Administrator beschließt, SMP3 das System Board SB5 wegzunehmen und SMP2 zuzuordnen.

Auf der dargestellten Konsole klick der Administrator auf das SB5 Ikon und zieht es mit der Maus rüber auf den SB2 Icon.

Die Granularität der Zuordnung auf die einelnen SMPs (Partitionen) war ursprünglich in Einheiten von einem ganzen System Boards. Dies ist bei neueren Systemen verfeinert worden.

Alternativen zur Harten Partitionierung

Die Harte Partitionierung (Hardware partitioning) ist einfach und sauber zu implementieren, indem man System Boards oder Gruppen von System Boards durch elektronische Schalter elektrisch voneinander trennt. Die Harten Partitionen verhalten sich dann wie getrennte physische Rechner. Über einen Switch können die einzelnen Harten Partitionen miteinander kommunizieren.

Die Harte Partitionierung ist jedoch nicht sehr flexibel. Große Unix Rechner wie z.B. HP Superdome und Sun 25K sowie das Nachfolgersystem Sun M9000 bieten deshalb zusätzlich eine „Virtuelle Partitionierung“ an, auch als „Software Partitioning“ bezeichnet. Bei den Sun und HP Servern hatte die virtuelle Patitionierung ursprünglich erhebliche Performance Probleme. In den letzten Jahren wurden deutliche Verbesserungen erreicht.

Beide Alternativen haben Vor- und Nachteile. Der Benutzer entscheidet von Fall zu Fall, welche der beiden Alternativen er bei einer Neuinstallation einsetzten will.

Eine harte Partitionierung ist bei dem IBM Unix „System p“ und bei Mainframes nicht erforderlich und nicht verfügbar. An ihrer Stelle werden „Logische Partitionen“ (LPARs) eingesetzt, die auf anderen Pattformen bis heute nicht verfügbar sind.

Virtuelle Partitionierung

Andere Bezeichnung: Virtualisierung

Die Virtuelle Partitionierung mit Software ist in der Regel flexibler als die Hardware-Partitionierung. Allerdings ist durch den Einsatz von Software der Overhead größer, der für die Steuerung der Umgebung benötigt wird.

Da die Hardware-Umgebung virtuell abgebildet wird. kann auch mit Hardware gearbeitet werden, die physisch gar nicht vorhanden ist. Weiterhin werden die Ressourcen einer virtuellen Maschine nur dann benötig, wenn in der virtuellen Maschine Anwendungen laufen.

Die Virtuelle Partitionierung mit Hilfe von Software wird häufig auch als Virtualisierung bezeichnet.

Emulator

Ein Emulator ist ein Software Paket, welches auf einem Rechner mit der Hardware-Architektur x (Host) einen Rechner mit der Hardware-Architektur y (Gast) emuliert. Jeder einzelne Maschinenbefehl eines Programms in der Architektur y wird durch den Emulater interpretiert.

Beispiele:

Hercules **und** ***IBM zPDT*** **emulieren einen System z Rechner mit dem z/OS Betriebssystem auf einem x86 Rechner.**

Microsoft VirtualPC **Typ 1 emuliert einen Intel/AMD Windows Rechner auf einem Apple MAC PowerPC Rechner mit einem (früher verfügbaren) PowerPC Mikroprozesssor.**

Bochs **ist ein in C++ geschriebener Open Source Emulator, der einen Rechner mit der die Intel/AMD Architektur auf vielen anderen Plattformen emuliert, z.B. PowerPC.**

Die Emulation ist sehr Performance-aufwändig. Ein Performance Verlust um einen Faktor 10 oder noch schlechter ist häufig anzutreffen. Mehrere emulierte Gast Rechner auf einem Host Rechner sind zwar möglich, aber wegen des Performance Verlustes nicht üblich.

Eine Java Virtuelle Maschine emuliert eine (heute nicht mehr verfügbare) Java Hardware Architektur auf einer x86, PowerPC oder System z Hardware. Die Java Hardware Architektur wurde seinerzeit mit dem Ziel entwickelt, den Performance Verlist möglichst klein zu halten (Faust Formel Faktor 3).

Hercules und IBM zPDT

Hercules ist ein Public Domain S/390 und System z Emulator, der für Linux, Windows (98, NT, 2000, and XP), Solaris, FreeBSD und Mac OS X Plattformen verfügbar ist. Moderne Mikroprozessoren sind schnell genug, so dass der Betrieb von z/OS unter Hercules auf einem PC für Experimentierzwecke oder für die Software Entwicklung für einen einzelnen Programmierer eine durchaus brauchbaren Performance ergibt. Siehe http://www.hercules-390.org/.

Hercules ist eine einwandfrei funktionierende S/390 und System z Implementiereng. Nicht verfügbar ist z/OS. Es würde zwar laufen, aber IBM vergibt keine Lizenzen für die Hercules Plattform. Verfügbar ist die letzte lizenzfreie Version MVS 3.8 aus dem Jahre 1981, die zwar kompatibel, aber eben doch sehr veraltet ist.

Allerdings verhält sich die Firma IBM zivilisierter als manche Unternehmen der Film- oder Musikindustrie bezüglich der Verfolgung von Hercules Lizenzverletzungen.

z Personal Development Tool (zPDT) ist ein offzielles IBM Produkt, vergleichbar zu Hercules. Es wird mit z/OS ausgeliefert, benötigt einen Dongle und läuft auf einer x86 Plattform. Der System z Emulator läuft unter Linux. Die Lizenzgebühren für zPDT sind jedoch recht hoch.

Virtuelle Maschine

Eine virtuelle Maschine ist etwas Ähnliches wie eine emulierte Maschine. Der Unterschied ist: Auf einem Host Rechner mit der Hardware-Architektur x wird ein (oder mehrere) Gast Rechner der gleichen Architektur abgebildet. Die meisten Maschinenbefehle der virtuellen Maschine brauchen nicht interpretiert zu werden. Der Leistungsverlust einer virtuellen Maschine hält sich deshalb in erträglichen Grenzen (bis zu 50 % unter VMWare, wenig mehr als 1 % unter z/OS PR/SM).

Beispiele für Hardware oder Betriebssysteme, die Virtuelle Maschinen unterstützen, sind:

VM/370, z/VM **und der** ***PR/SM*** **Hypervisor für die System z und S/390 Hardware-Architektur.**

Für den PowerPC Microprozessor existiert der LPAR Hypervisor.

XEN, Virtual Box, KVM**, (alle Open Source),** ***VMWare*** **und** ***Microsoft VirtualPC*** **Typ 2 bilden mehrere Windows oder Linux Gast Maschinen auf einem Windows oder Linux Host ab.**

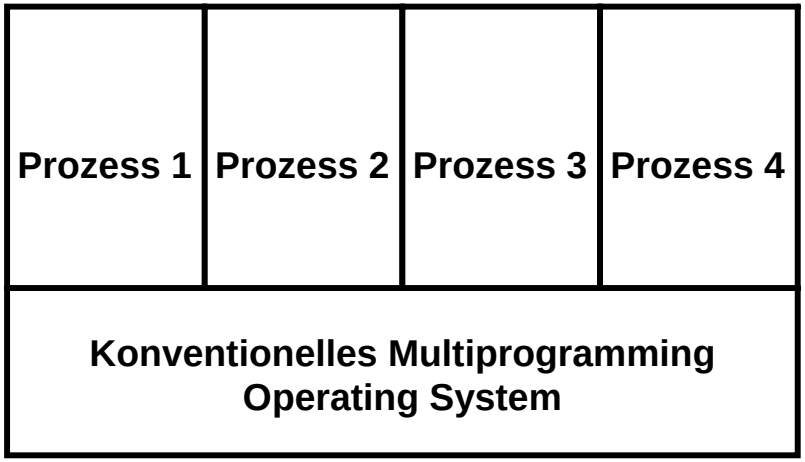
Oracle VM Server for Sparc ist eine Weiterentwicklung von XEN.

Intel Virtualization Technology für die Itanium Architecture (VT-i) sowie die x86 (Pentium) Architecture (VT-x) ist eine Hardware Erweiterung, welche die Virtualisierung verbessert.

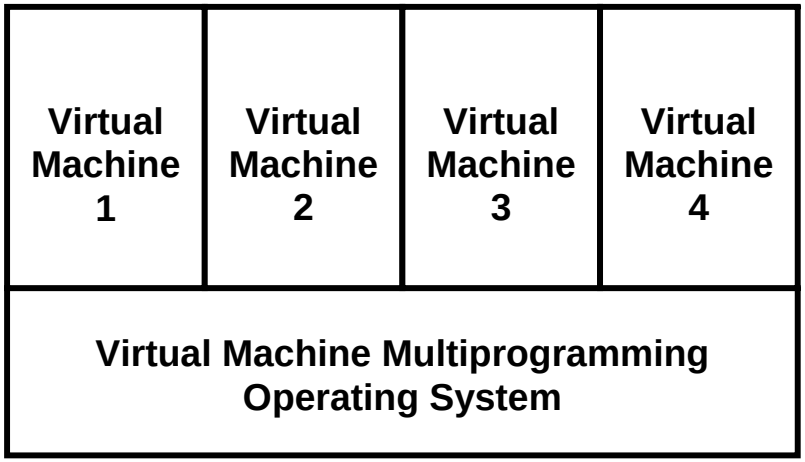
Pacifica für AMD Prozessoren ist eine zu VT-x nicht kompatible (und nach Ansicht mancher Experten überlegene) Alternative .

Paravirtualization wird von Xen und Denali implementiert.

**Weiterführende Literatur: Joachim von Buttlar, Wilhelm G. Spruth: Virtuelle Maschinen. zSeries und S/390 Partitionierung.
IFE - Informatik Forschung und Entwicklung, Heft 1/2004, Juli 2004. http://www-ti.informatik.uni-tuebingen.de/~spruth/publish.html**



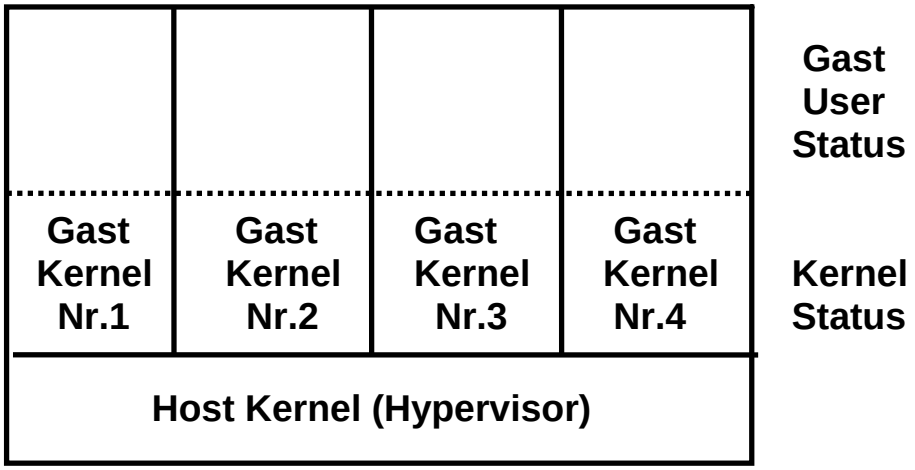
Unter einem normale multiprogrammierten Betriebssystem laufen zahlreiche Prozesse gleichzeitig ab. Jeder Prozess läuft in einem eigenen virtuellen Adressenraum.



Unter einem Virtual Machine Operating System laufen spezielle Prozesse, von denen jeder eine virtuelle Maschine implementiert. Jede virtuelle Maschine hat ihr eigenes Betriebssystem, unter dem Anwendungen der virtuellen Maschine laufen. Auf zwei virtuellen Maschinen können unterschiedliche Betriebssysteme laufen.

Das Virtual Machine Operating System wird häufig als Hypervisor oder als Host Betriebssystem bezeichnet. Analog bezeichnet man das emulierte Betriebssystem als das *Gast-Betrieb*ssystem, welches auf einer *Gast-Maschine* (virtuelle 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.

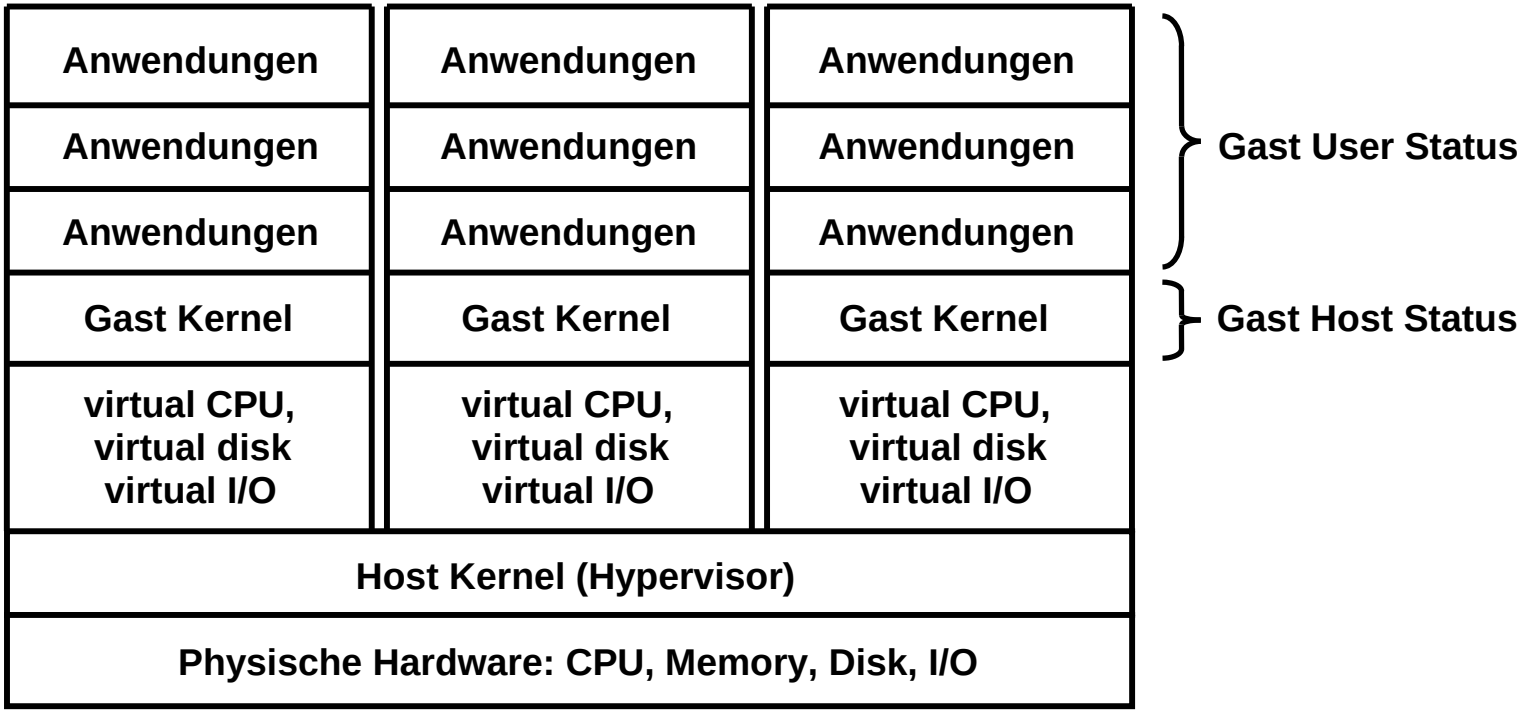
Die Begriffe virtuelle Maschine und Gast Maschine sowie die Begriffe Hypervisor, Host Kernel und Virtual Machine Monitor (VMM)bedeuten jeweils das Gleiche und werden austauschbar verwendet.



Gleichzeitiger Betrieb mehrerer Betriebssysteme auf einem Rechner

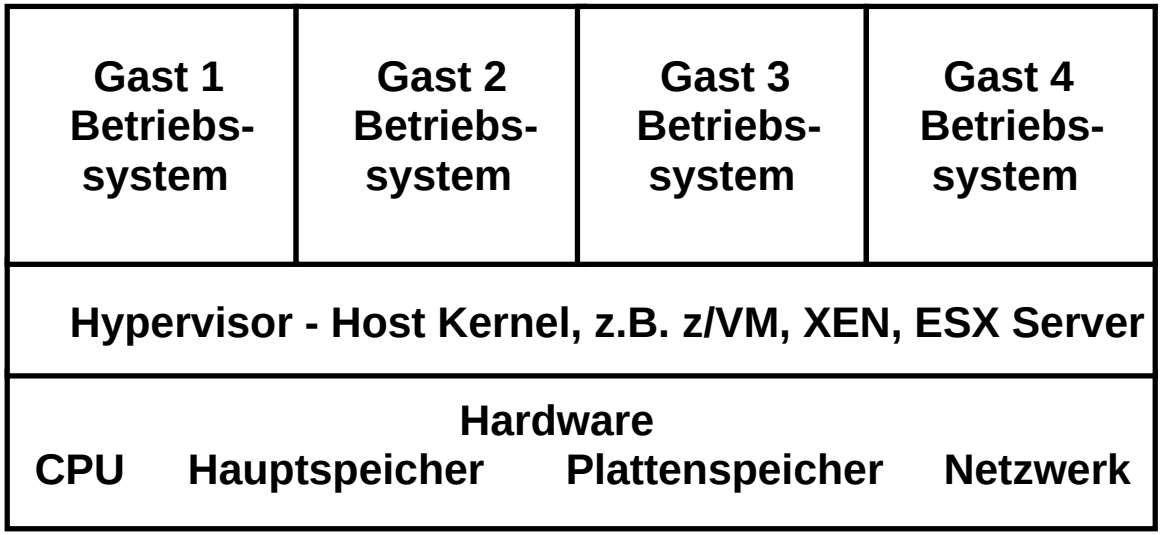
Der Ansatz, mehrere *Gast*-Betriebssysteme unter einem *Host*-Betriebssystem zu betreiben, wird als virtuelle Partitionierung bezeichnet. Beispielsweise kann ein Rechner mit 80 CPUs als 5 SMPs mit je 16 CPUs mittels der virtuellen Partitionierung betrieben werden.

Der Host-Kernel übernimmt die Kontrolle über den Rechner 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. 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 fremder 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 er als Gast betrieben wird. Im Gegensatz dazu ist eine vollständige Emulation sehr aufwendig.

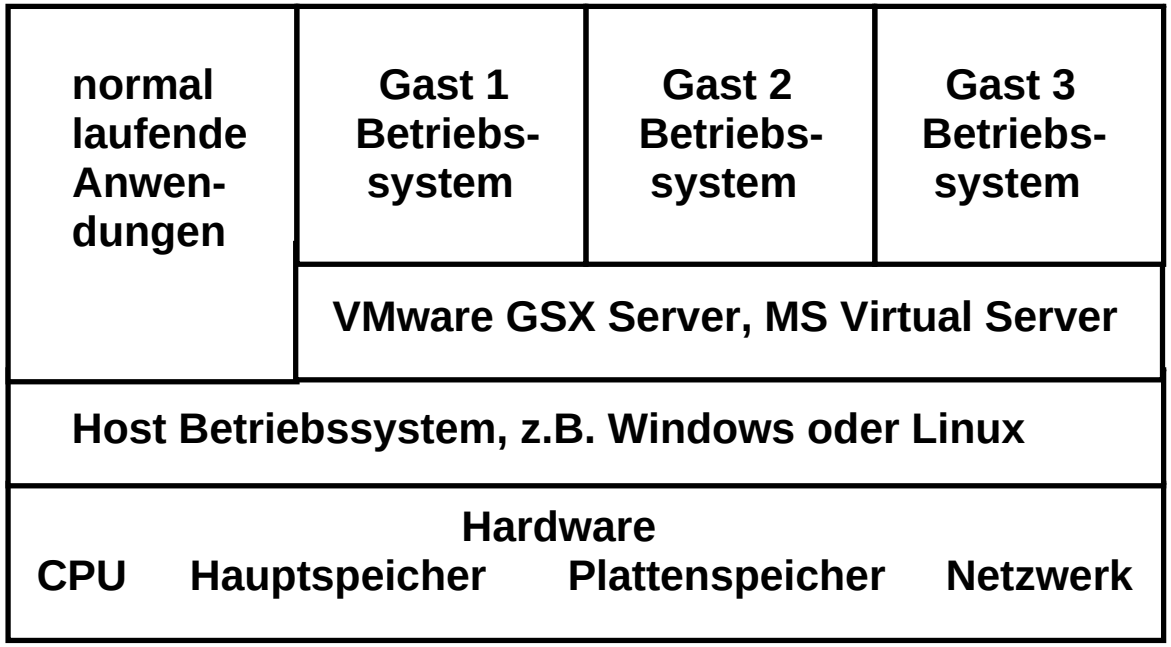


Es ist möglich, jedem Gast System eigene I/O Geräte, wie Plattenspeicher oder Drucker, fest zuzuordnen. Vielfach werden aber auch die I/O Geräte virtualisiert. Beispielsweise können mehrere virtuelle Drucker auf einen realen Drucker abgebildet werden. Virtuelle Plattenspeicher (Minidisks) können als Dateien auf einem realen Plattenspeicher abgebildet werden.

Typ A



Typ B



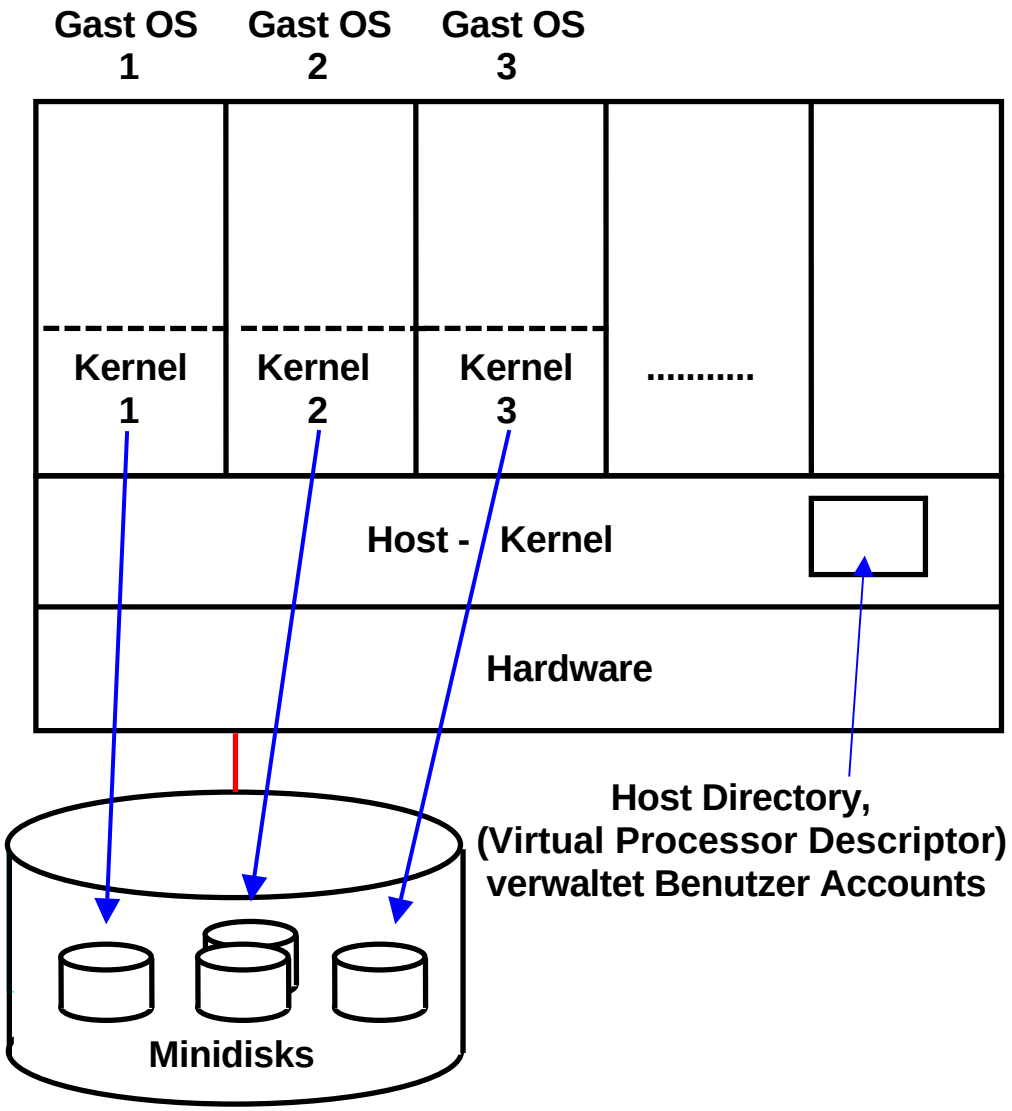
Alternativen für die Virtualisierung

Es existieren zwei Alternativen für die Implementierung.

Typ A ist die bisher beschriebene Konfiguration. Sie wird auf allen größeren Servern eingesetzt, besonders auch auf den Mainframes.

Für den PC existiert als zusätzlich Alternative Typ B. Der Host Kernel läuft als Anwendung unter einem regulären PC Betriebssystem, z.B. Windows oder Linux. Er hat nicht alle Eigenschaften eines selbständigen Betriebssystem Kernels, sondern nutzt statt dessen teilweise Komponenten des darunterliegenden Windows oder Linux Betriebssystems.

Der Vorteil von Typ B ist, dass Anwendungen auch nicht virtualisiert mit besserem Leistungsverhalten laufen können. Als Nachteil laufen virtuelle Maschinen bei Typ B langsamer als bei Typ A.



Im einfachsten Fall werden dem Gast Betriebssystemen Ressourcen wie

- CPU-Zeit,
- Aufteilung auf mehrere CPUs in einem Mehrfachrechner,
- Hauptspeicher,
- Plattenspeicher
- Ein-/Ausgabe-Geräte und -Anschlüsse

fest zugeordnet. Alternativ, z.B. Mainframes, ist auch eine dynamische Zuordnung möglich.

Bei kleinen Systemen mit nur einem einzigen physischen Plattenspeicher können mehrere Platten der virtuellen Systeme als „Minidisks“ auf dem physischen Plattenspeicher emuliert und den einzelnen virtuellen Maschinen zugeordnet werden.

Der Host Kernel enthält ein „Host Directory“, welches die Konfiguration der virtuellen Maschinen verwaltet.

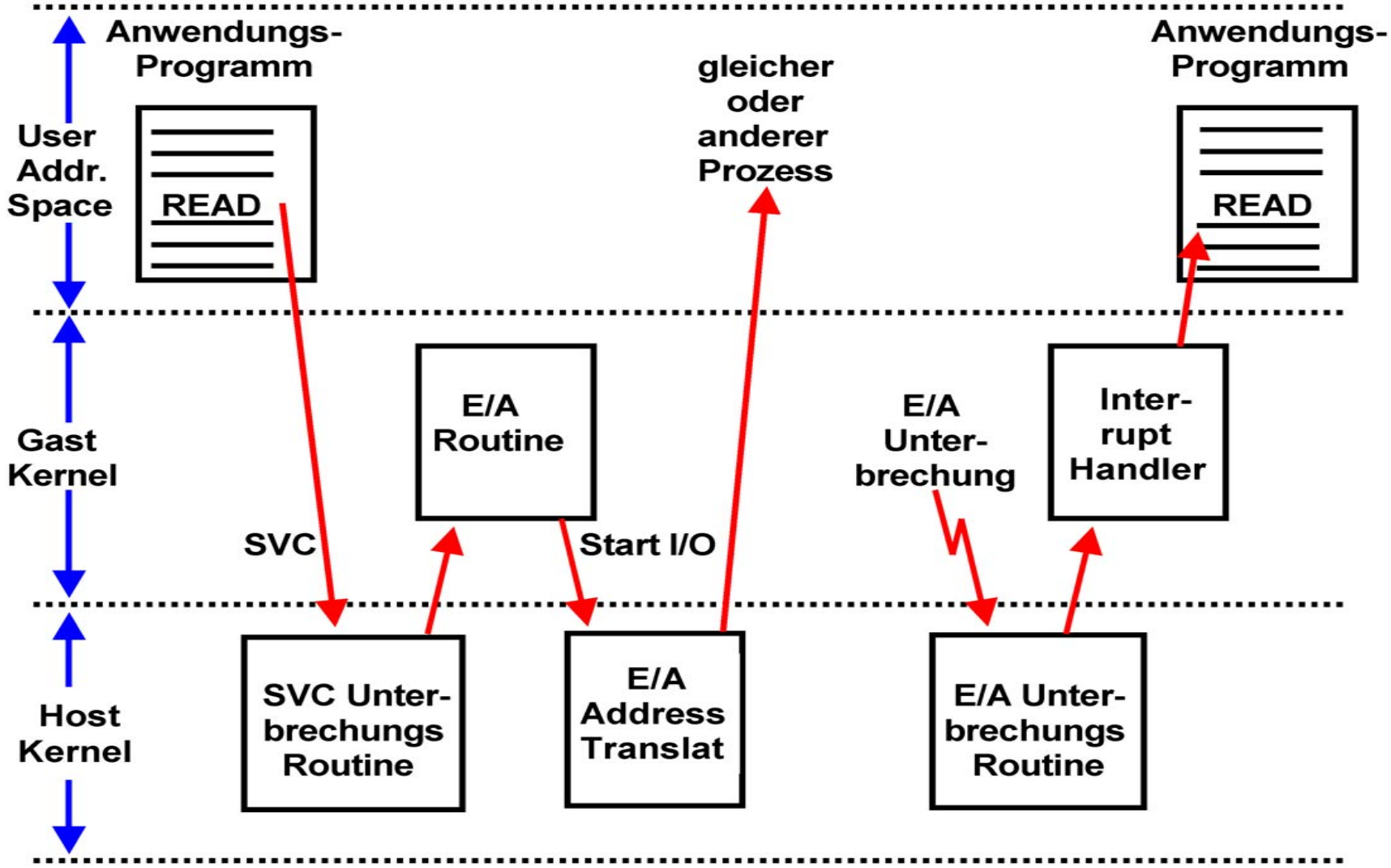
Steuerung der virtuellen Maschine

Alle Gast Maschinen laufen in einem eigenen virtuellen Adressenraum

Der Host Kernel Zeitscheiben-Scheduler übergibt die Kontrolle über die CPU(s) einer Gast Maschine.

Der Kernel der Gast Maschine läuft im User Mode (Problem Mode). Wenn das Programm des Gast Betriebssystems versucht, einen privilegierten Maschinenbefehl auszuführen, führt dies zu einer Programmunterbrechung.

Die Programmunterbrechungsroutine des Host Kernels interpretiert den privilegierten Maschinenbefehl der Gast Maschine soweit als erforderlich und übergibt die Kontrolle zurück an den Kernel des Gastbetriebssystems



Als Beispiel ist eine I/O READ Operation dargestellt, die von einem Anwendungsprogramm einer Gastmaschine ausgeführt wird.

Ausführung einer I/O Operation

Beispiel READ

Die Ausführung einer (Plattenspeicher) READ Operation in einem Anwendungsprogramm führt zum Aufruf einer Library Routine, welche mittels eines SVC (Supervisor Call) Maschinenbefehl in eine Routine des Kernels verzweigt. Diese führt die I/O Operation stellvertretend für den Anwendungsprozess aus.

Eine virtuelle Maschine läuft als Prozess unter dem Host Kernel im User Status, einschließlich des Kernels des Gast Betriebssystems. Erfolgt die READ Operation in einer virtuellen Gast-Maschine, so bewirkt der SVC Befehl einen Aufruf nicht des Gast Kernels sondern des Host Kernels, da nur dieser im Kernel Status läuft. Nur dieser ist in der Lage, den eigentlichen I/O Driver auszuführen.

Auf der anderen Seite verwaltet der Gast Kernel die I/O Buffer oder die I/O Steuerblöcke (z.B. DCB, ECB, UCB unter z/OS). Der Host Kernel delegiert deshalb die Erstellung der I/O Anforderung an den Gast Kernel.

Die so erstellte I/O Anforderung arbeitet aber mit den virtuellen Adressen des Gast Kernels. Der Gast Kernel übergibt deshalb die entgültige Fertigstellung an den Host Kernel (über einen SVC Aufruf), der die I/O Anforderung vervollständigt und an die Plattenspeicher-Steuereinheit weitergibt.

Der Abschluss der I/O Operation wird mit einer I/O Unterbrechung dem Host Kernel übergeben. Dieser übergibt die Unterbrechung an den entsprechenden Unterbrechungshandler des Gast Kernels. Letzterer benachrichtigt die Anwendung, welche die READ Operation auslöste, von deren erfolgreichen Abschluss.