EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Enterprise Computing
Virtualisierung

Prof. Dr.-Ing. Wilhelm G. Spruth
Dipl. Inf. Gerald Kreißig

WS2016/17

Virtualisierung Teil 1

Partitionierung

Skalierung eines Symmetrischen Multiprozessors

Ein Symmetrischer Multiprozessor (SMP) mit zwei statt einer CPU sollte idealerweise 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 Ressource 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.

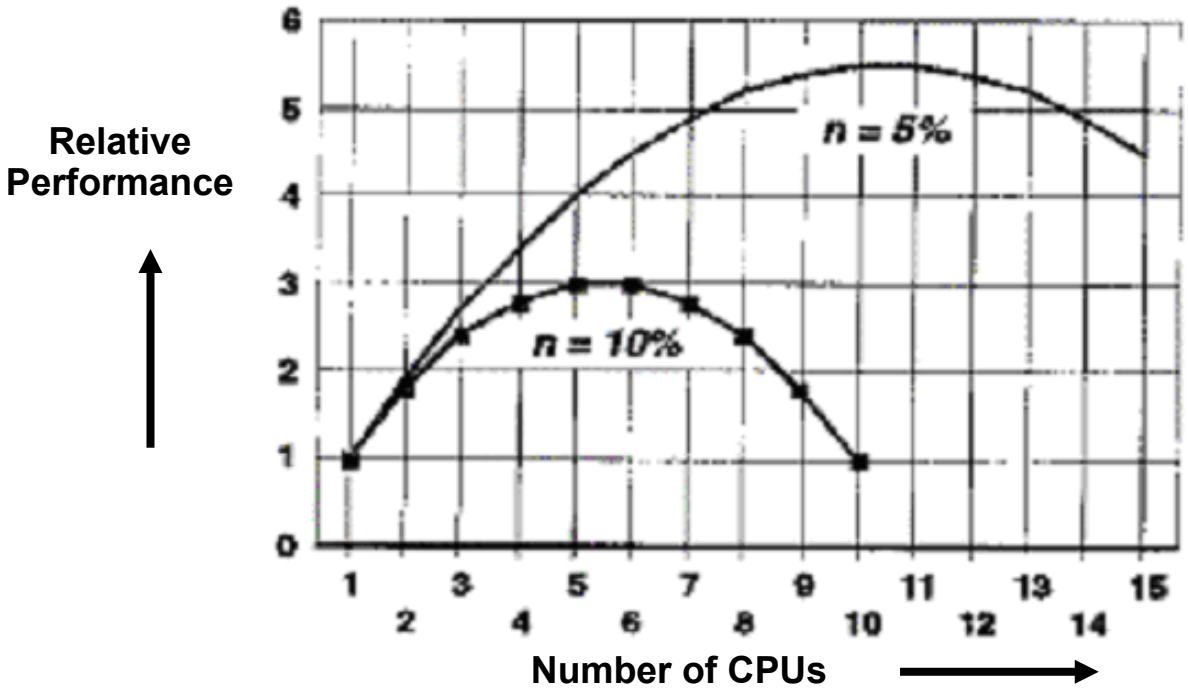
Wir haben in den vergangenen Jahrzehnten Fortschritte für SMP Rechner mit zunächst 2, dann 4, später 8, dann 12, 16 usw. CPUs gemacht, indem Wege gefunden wurden, den Kernel möglichst optimal in mehr und mehr Teile zu gliedern, die durch Locks von der gleichzeitigen Nutzung durch zwei oder mehr CPUs 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 wenige 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 [1]. 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.

[1] Eine weitere Ausnahme sind Rechnerarchitekturen, die auf massive Parallelität ausgelegt sind, z.B. der Tilera64 (Parallelität auf Threading-Ebene) oder GPGPUs von NVidia und AMD (Parallelität auf Datenebene). Diese Systeme skalieren – je nach eingesetztem Algorithmus – problemlos auf bis zu mehrere tausend Recheneinheiten. Sie nutzen den Vorteil aus, dass hier kein Überwacher existiert. Nachteilig ist, dass die Algorithmen für diese Architekturen geeignet sein müssen und auf diesen Rechnerarchitekturen normale Anwendungen – wenn überhaupt – nur sehr ineffizient laufen. Sie sind nicht für die Aufgaben von klassischen CPUs verwendbar.



Die beiden dargestellten Kurven repräsentieren das typische Leistungsverhalten eines SMP als Funktion der Anzahl der eingesetzten 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 begrenzten maximalen Anzahl von 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 zEC12 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.

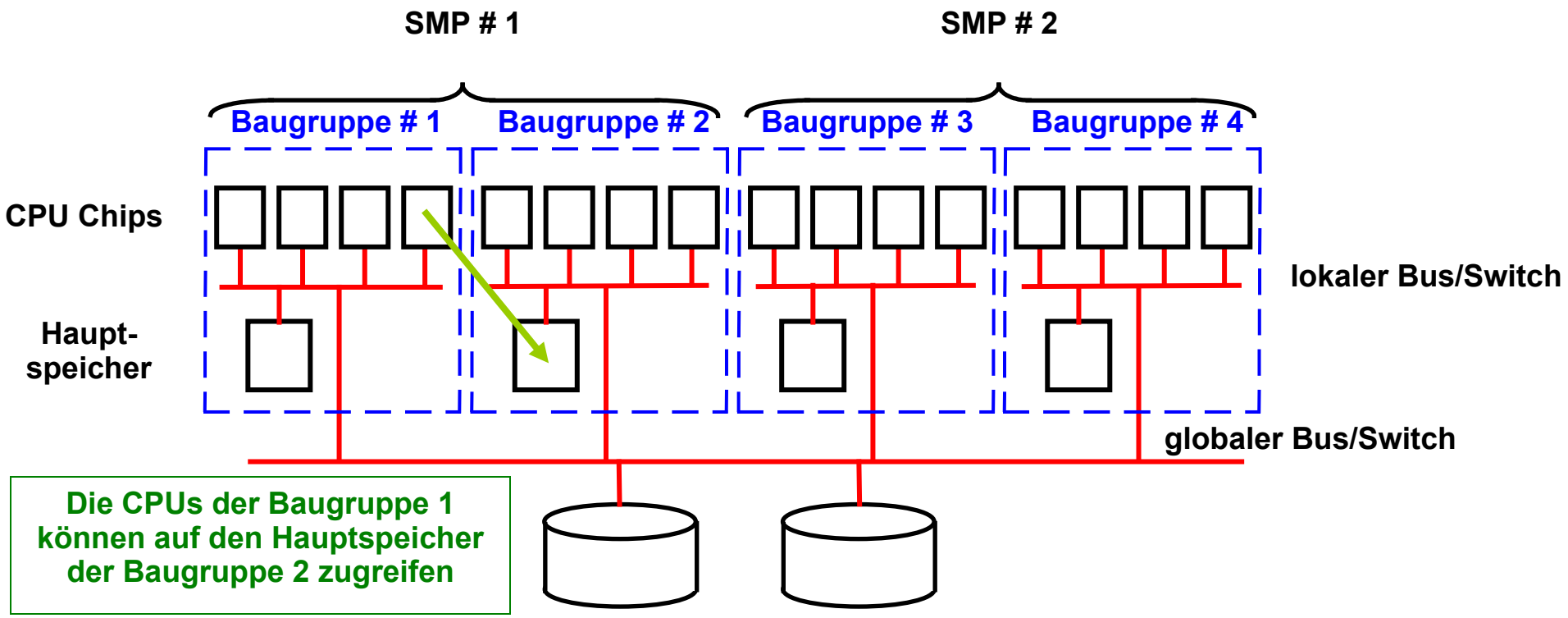
Die System Boards einer Sun 25K oder Cell Boards eines HP Superdome Rechners enthalten jeweils ihren eigenen Hauptspeicher. Im einfachsten Fall partitioniert man den Rechner so, dass jede Baugruppe (System Board oder Cell Board) gleichzeitig ein SMP ist. Die CPUs einer Baugruppe greifen dann nur auf den Hauptspeicher der gleichen Baugruppe zu.

Wenn ein SMP aus mehr als einer Baugruppe besteht, muss eine SMP Betriebssystem Instanz auf den Hauptspeicher aller beteiligten Baugruppen zugreifen können.

Hierzu ist es erforderlich, 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).

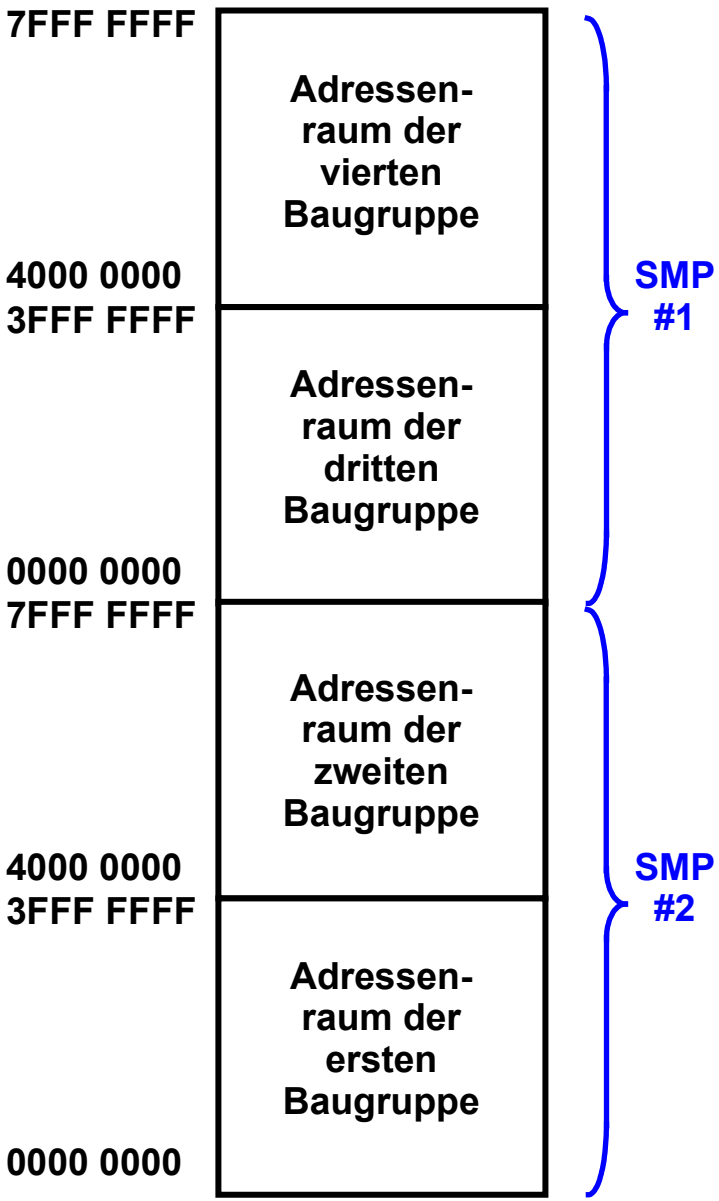
Dies geht aber nur bis zu der erwähnten Grenze von 12 – 16 CPUs. Ein SMP kann zwar aus mehr als einer Baugruppe bestehen, aber bestenfalls aus sehr wenigen Baugruppen. Deshalb besteht ein Rechner mit z.B. 16 Baugruppen aus mehreren SMPs, auch wenn ein SMP aus mehr als einer Baugruppe besteht. Die zu unterschiedlichen SMPs gehörigen Baugruppen werden durch Schalter voneinander getrennt (sog. „harte“ Partitionierung).

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. Jeweils 2 Baugruppen bilden einen SMP. Alle CPUs eines SMPs können jeweils auf den Hauptspeicher der anderen Baugruppe zugreifen.

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 von zwei SMP Rechnern bestehend aus jeweils 2 Baugruppen Jede Baugruppe verfügt über einen Hauptspeicher von 1 GByte.

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

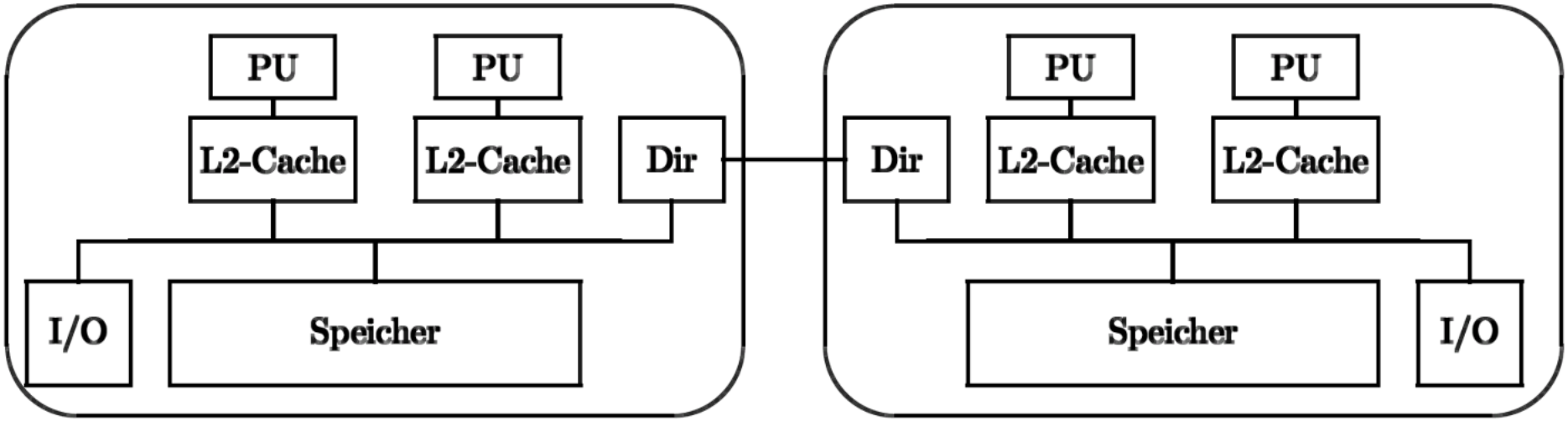
Der zweite der 4 Baugruppen benutzt die Adressen Hex 4000 0000 bis 7FFF FFFF.

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

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

Da alle Prozesse in den SMPs mit virtuellen Adressen arbeiten, ist diese Aufteilung der realen Adressen problemlos möglich.

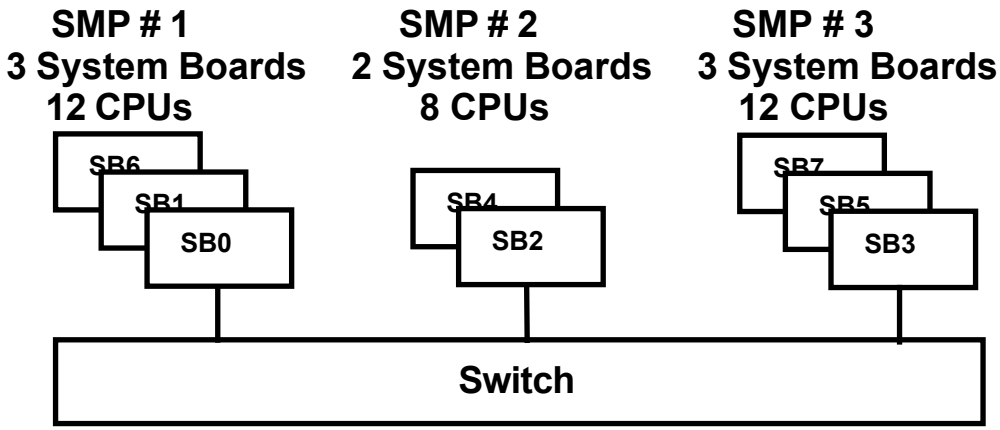
NUMA-Rechner



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 bewirkt die **Dir-Einheit (Directory)**. Diese enthält Informationen über alle auf den externen Baugruppen 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



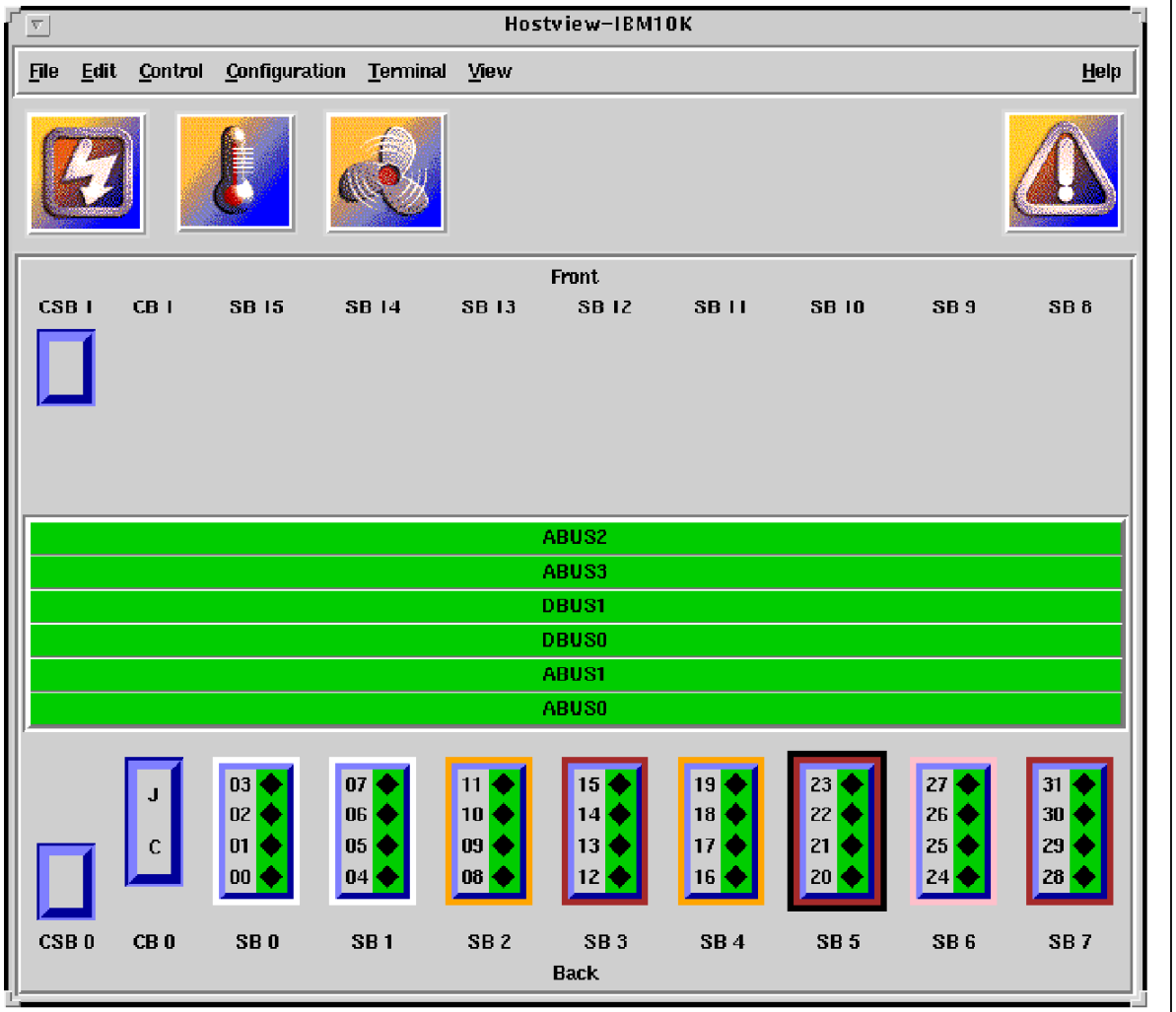
Als Beispiel ist die Aufteilung eines Sun 25K (oder HP Superdome) Servers mit 8 Baugruppen (System Boards) mit je 4 single Core CPU Chips in mehrere parallel laufende SMPs gezeigt.

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. Bei der „Harten Partitionierung“ erfolgt die Aufteilung 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 während des laufenden Betriebes 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 SMP1 überschüssige Kapazität hat.

Der Administrator beschließt, SMP1 das System Board SB6 wegzunehmen und SMP2 zuzuordnen.

Auf der in der Abbildung dargestellten Konsole klickt der Administrator auf das SB6 Icon und zieht es mit der Maus herüber auf den SB2 Icon.

Die Granularität der Zuordnung auf die einzelnen SMPs (Partitionen) war ursprünglich in Einheiten von 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 den Nachfolgersystemen Sun M9000 und M5-32 bieten deshalb zusätzlich eine **Virtuelle Partitionierung** an, auch als „Software Partitioning“ oder „Virtualisierung“ bezeichnet. Bei der virtuellen Partitionierung läuft jeder SMP in einer „virtuellen Maschine“.

Beide Alternativen haben Vor- und Nachteile. Der Benutzer entscheidet von Fall zu Fall, welche der beiden Alternativen er bei einer Neuinstallation einsetzten will. Bei den Sun und HP Servern hatte die virtuelle Partitionierung ursprünglich erhebliche Performanceprobleme. In den letzten Jahren wurden deutliche Verbesserungen erreicht..

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.

Paravirtualization.

Hierbei ist die Betriebssystem-Architektur der virtuellen Maschine nicht vollständig mit der Host Betriebssystem-Architektur identisch. Die Benutzerschnittstelle (Application Binary Interface, ABI) ist die gleiche. Der Gast-Kernel unterstellt jedoch Abweichungen zu der x86-Architektur. Dies verbessert das Leistungsverhalten, erfordert aber Änderungen des Gast-Kernels. Hiervon ist nur ein sehr kleiner Teil des Kernel-Codes betroffen. Beispielsweise existiert ein funktionsfähiger Linux-Port (XenoLinux), dessen ABI mit dem eines nicht-virtualisierten Linux identisch ist.

Ein ähnlicher Ansatz wird von *Denali verfolgt*. Als Gast-Betriebssystem dient *Ilwaco*, eine speziell an den Denali Hypervisor angepasste BSD Version. Denali unterstützt nicht das vollständige x86 ABI. Es wurde für Netzwerk-Anwendungen entwickelt und unterstellt Einzelbenutzer-Anwendungen. Mehrfache virtuelle Adressräume sind unter Ilwaco nicht möglich.

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 Emulator 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 Performanceverlustes 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 Performanceverlust möglichst klein zu halten (früher: Faustformel Faktor 3 heute unter optimalen Bedingungen ca. 20%).

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 Implementierung. 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 offizielles 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.

Virtualisierung Teil 2

Virtual Machine

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**.

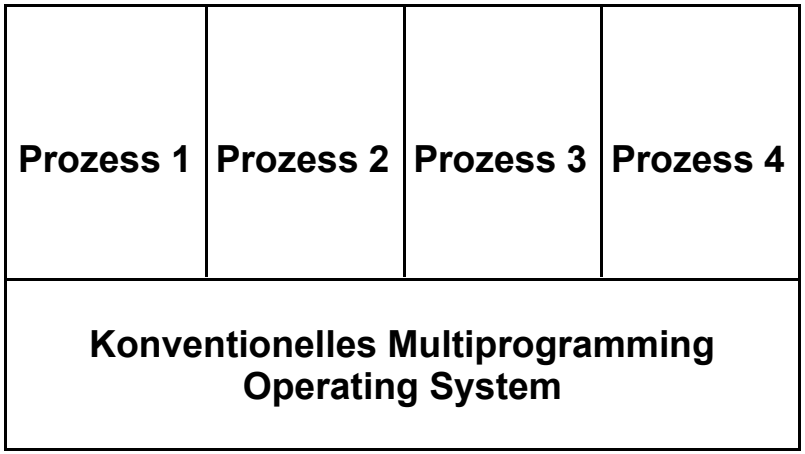
XEN, Virtual Box, KVM, (alle Open Source), ***VMWare*** und ***Microsoft VirtualPC*** Typ 2 bilden mehrere Windows oder Linux Gast Maschinen auf einem Windows oder Linux Host ab. **Oracle VM Server for Sparc** ist eine Weiterentwicklung von XEN.

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

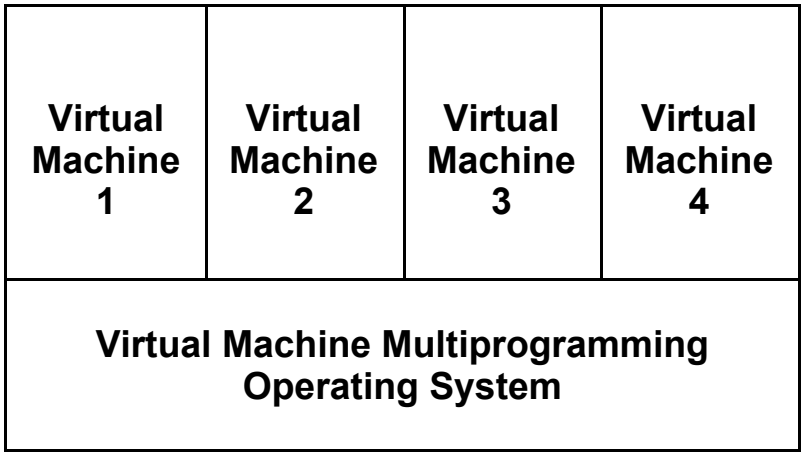
Pacifica (AMD-V) für AMD Prozessoren ist eine zu VT-x nicht kompatible (und nach Ansicht mancher Experten überlegene) Alternative. I/O MMU Virtualization (AMD-Vi und VT-d) erlaubt die direkte Anbindung von peripheren Geräten an die virtuellen Maschinen, dazu gehören auch DMA und Interrupt Remapping.

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

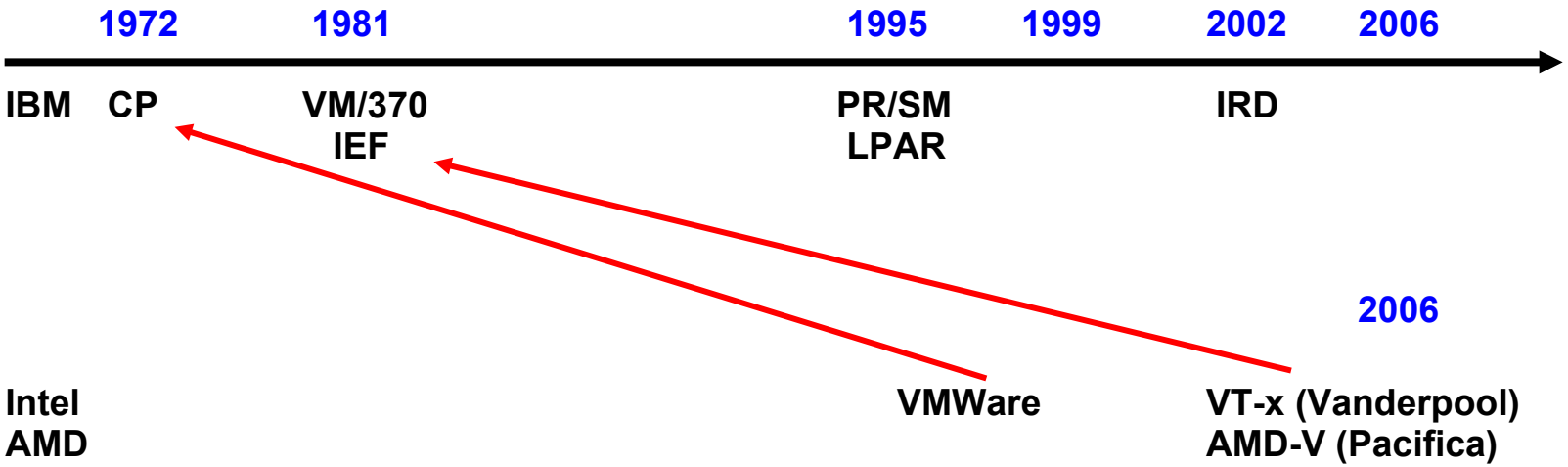


Unter einem Virtual Machine Operating System laufen spezielle Prozesse, von denen jeder eine virtuelle Maschine implementiert. Jede virtuelle Maschine hat ihr eigenes Betriebssystem, unter dem Anwendungen der virtuellen Maschine laufen. Auf zwei virtuellen Maschinen können unterschiedliche Betriebssysteme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

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.

Die Entwicklung der Virtualisierung



Das VM/370 Betriebssystem mit dem CP Hypervisor und dem CMS Gast-Betriebssystem ist seit 1972 verfügbar.
1995 gründeten einige VM/370 Entwickler die Firma VMware. Seit 1999 ist VMWare für die x86 Plattform verfügbar.

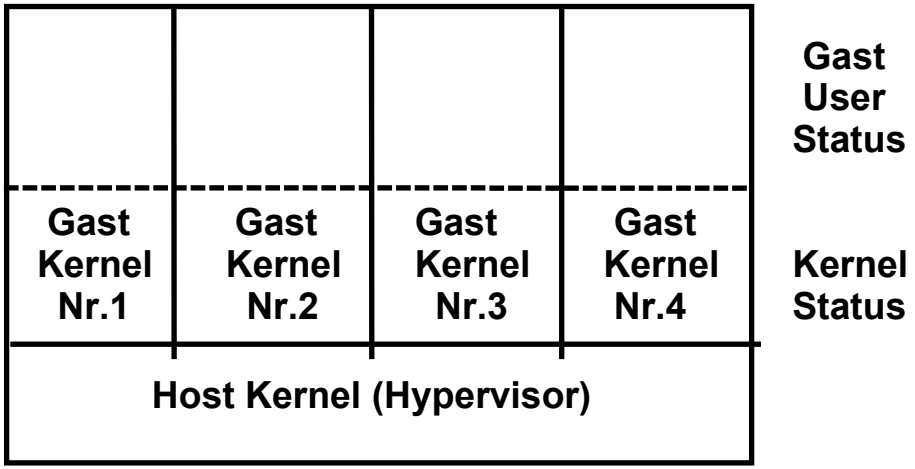
Bereits 1981 hatte IBM für alle S/370 Rechner eine Erweiterung der Hardware Architektur eingeführt, die als **Interpretive Execution Facility (IEF)** bezeichnet wurde, und zu einer erheblichen Leistungsverbesserung der virtuellen Maschinen führte. Eine vergleichbare Architekturerweiterung wurde von Intel und AMD im Jahre 2006 für die x86 (Pentium) Architektur unter dem Namen Vanderpool bzw. Pacifica eingeführt.

VT-x (Virtualisation Technology for x86) ist eine alternative Bezeichnung für Vanderpool. Eine ähnliche Einrichtung für Itanium hat den Namen VT-i. AMD Virtualization, abgekürzt AMD-V ist eine alternative Bezeichnung für Pacifica.

Bereits vorher im Jahre 1995 hatte die S/390 Architektur eine weitere grundlegende Verbesserung unter dem Namen **Logische Partition (LPAR)** eingeführt. Dies wurde 2002 durch eine nochmalige Verbesserung, den **Intelligent Resource Director (IRD)** ergänzt. Der PowerPC führte LPARs 2002 ein.

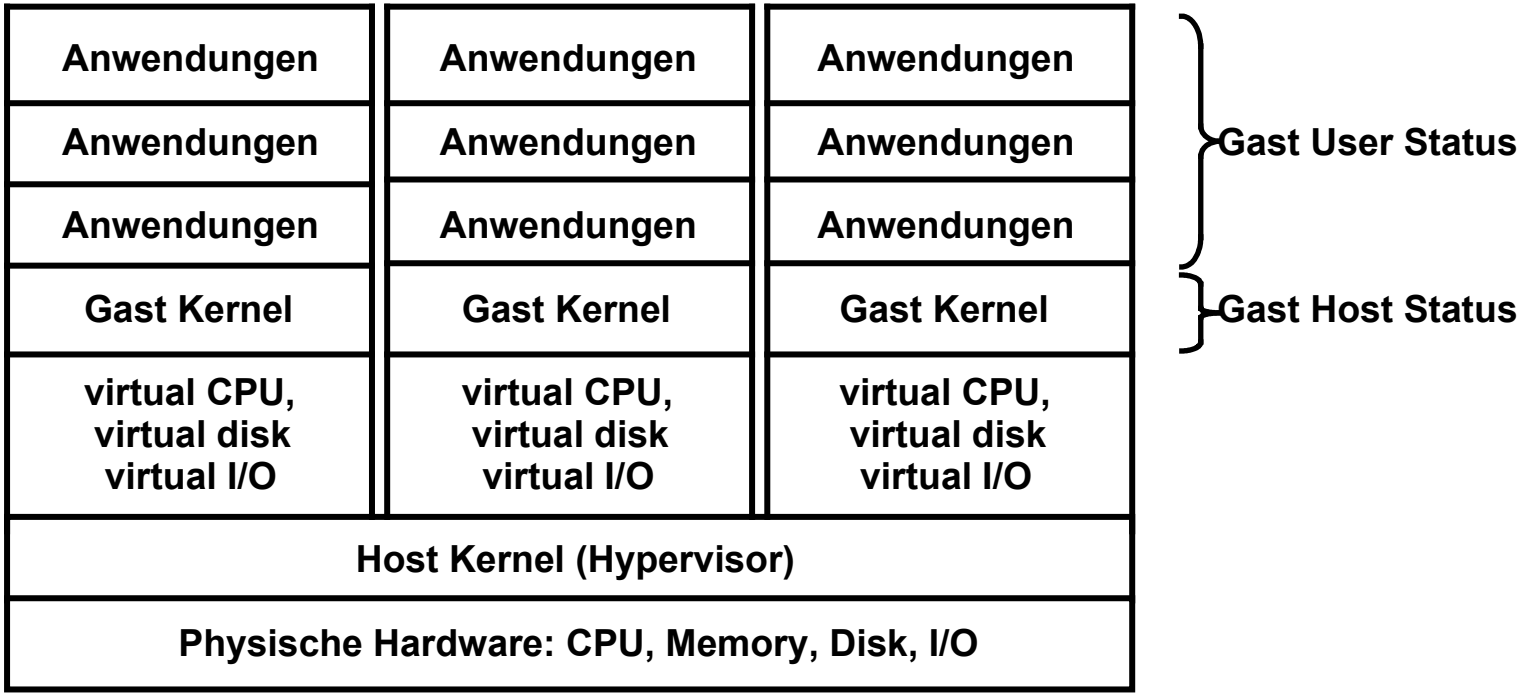
Experten gehen davon aus, dass x86 weitere 10 Jahre benötigen wird, um eine mit LPAR und IRD vergleichbare Funktionalität zu erreichen.

Gleichzeitiger Betrieb mehrerer Betriebssysteme auf einem Rechner

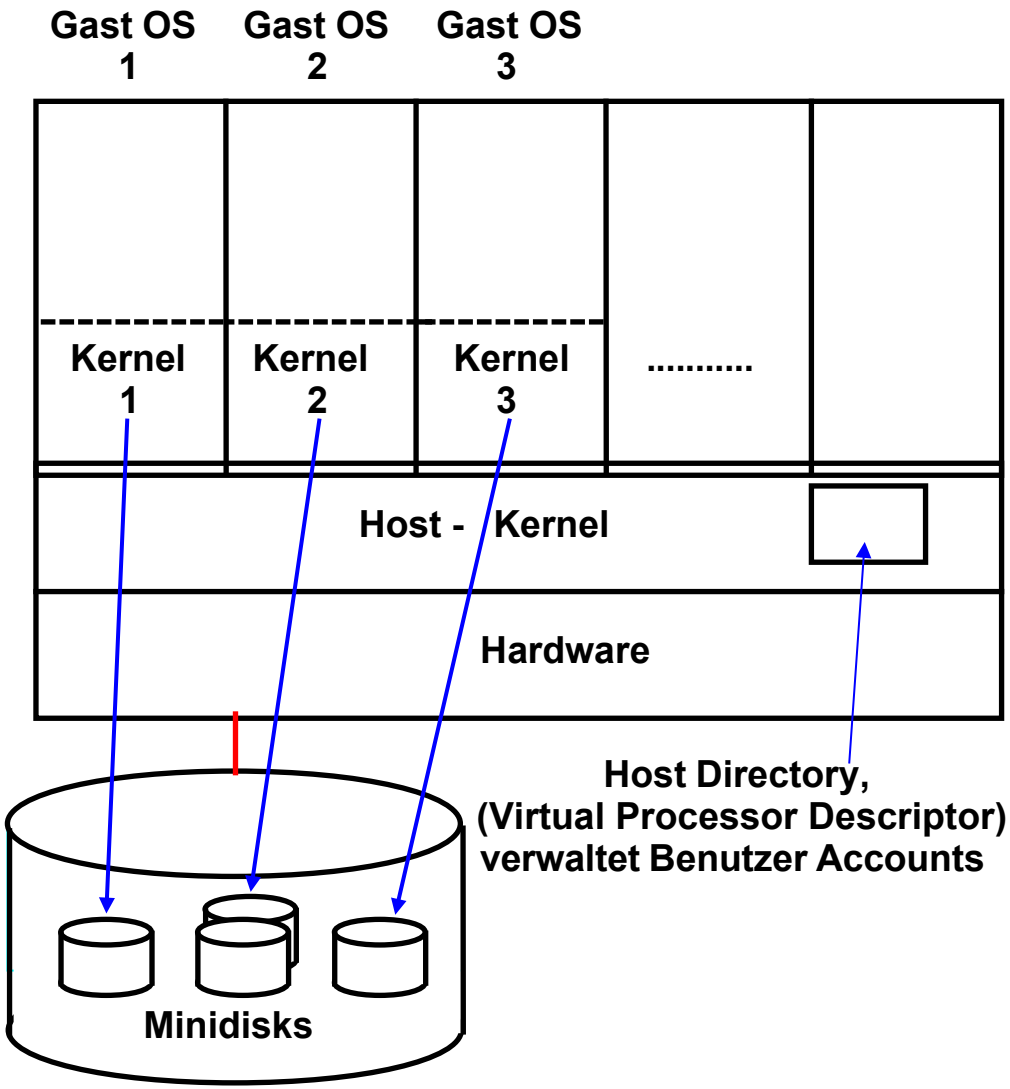


Der Ansatz, mehrere *Gast*-Betriebssysteme unter einem *Host*-Betriebssystem zu betreiben, wird als virtuelle Partitionierung bezeichnet. Beispielsweise kann ein Rechner mit 80 CPUs als 5 SMPs mit je 16 CPUs mittels der virtuellen Partitionierung betrieben werden.

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 mäßig, 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.



Im einfachsten Fall werden dem Gast Betriebssystemen Ressourcen wie

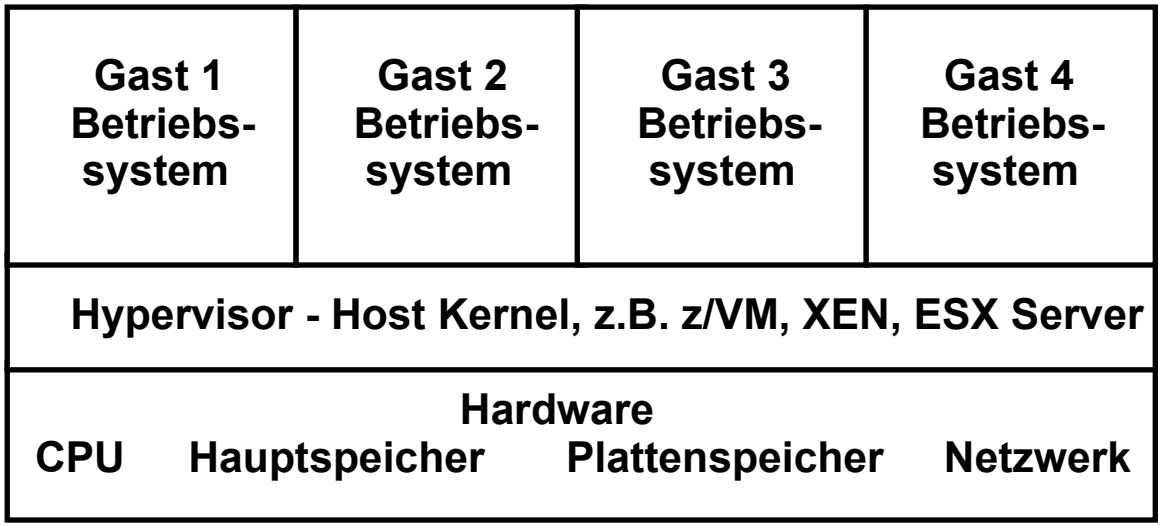
- CPU-Zeit,
- Aufteilung auf mehrere CPUs in einem Mehrfachrechner,
- Hauptspeicher,
- Plattenspeicher
- Ein-/Ausgabe-Geräte und -Anschlüsse

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

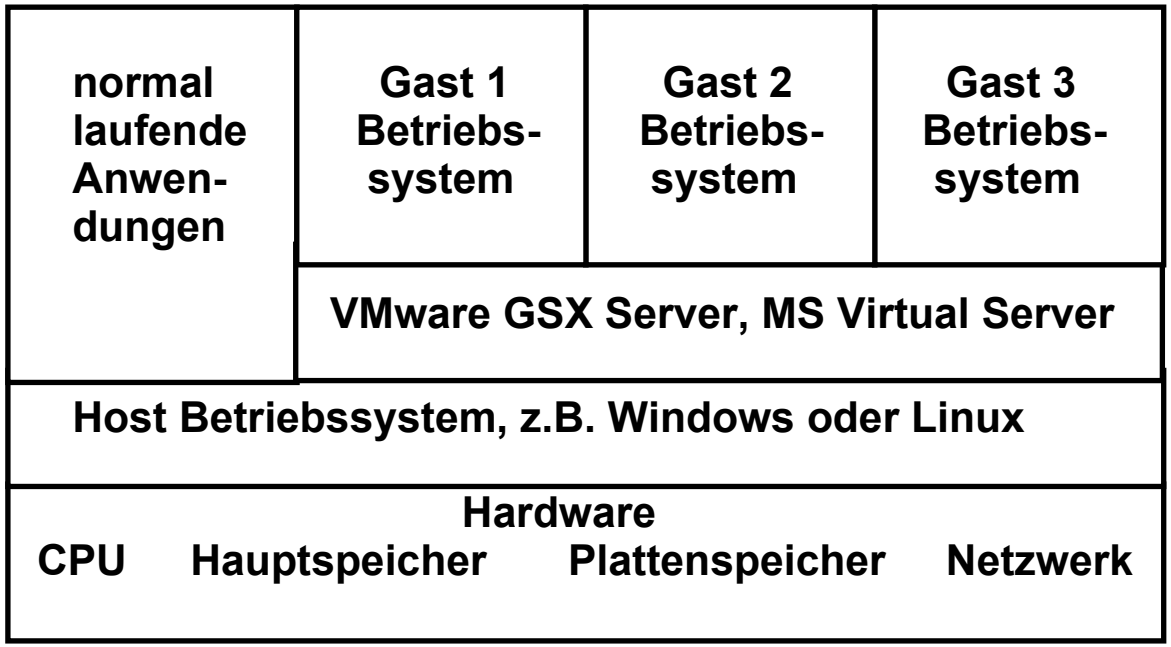
Bei kleinen Systemen mit nur einem einzigen physischen Plattenspeicher können mehrere Platten der virtuellen Systeme als „Minidisks“ auf dem physischen Plattenspeicher emuliert und den einzelnen virtuellen Maschinen zugeordnet werden.

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

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 stattdessen 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.

z/VM (VM/370) Betriebssystem

Für Mainframe Rechner existiert seit 1972 ein spezielles Betriebssystem VM/370, welches heute fast ausschließlich für Virtualisierungsaufgaben eingesetzt wird.

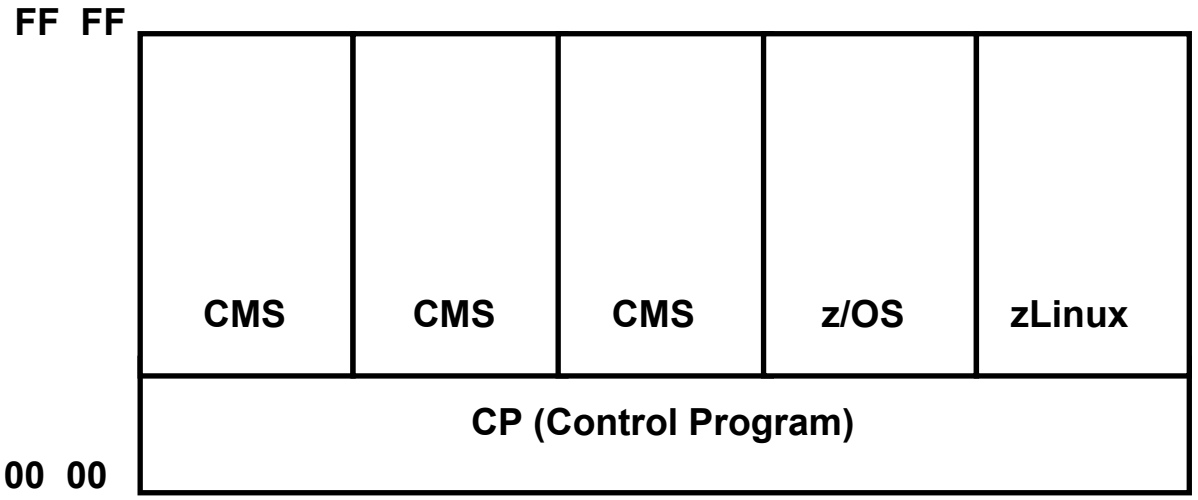
Ein Vorläufer wurde 1965 unter der Bezeichnung CP/40 im IBM Scientific Center in Cambridge/Mass. entwickelt. Eine spezielle Version CP/67 wurde 1967 für das Rechnermodell S/360-67 für den Vertrieb freigegeben.

Seitdem wurden zahlreiche Verbesserungen und Erweiterungen eingeführt. Von besonderer Bedeutung war die Entwicklung der „Interpretive Execution Facility“. (IEF) im Jahre 1981. Die heutige Bezeichnung des Betriebssystems ist **z/VM**.

Unter CP/67 und VM/370 laufen alle Anwendungen grundsätzlich auf virtuellen Maschinen. Der Host Kernel (Hypervisor) wird als **Control Programm (CP)** bezeichnet und läuft im Kernelstatus.

http://www.multicians.org/thvv/360-67.html

z/VM (VM/370) Betriebssystem



Für CP/67 und VM/370 wurde ein eigenes Gast Betriebssystem CMS (Conversational Monitor System) entwickelt. Dieses und alle anderen Gast Betriebssysteme (einschließlich ihrer Kernel Funktionen) laufen im Problemstatus. Privilegierte Maschinenbefehle (z.B. E/A) werden von CP abgefangen und interpretativ abgearbeitet.

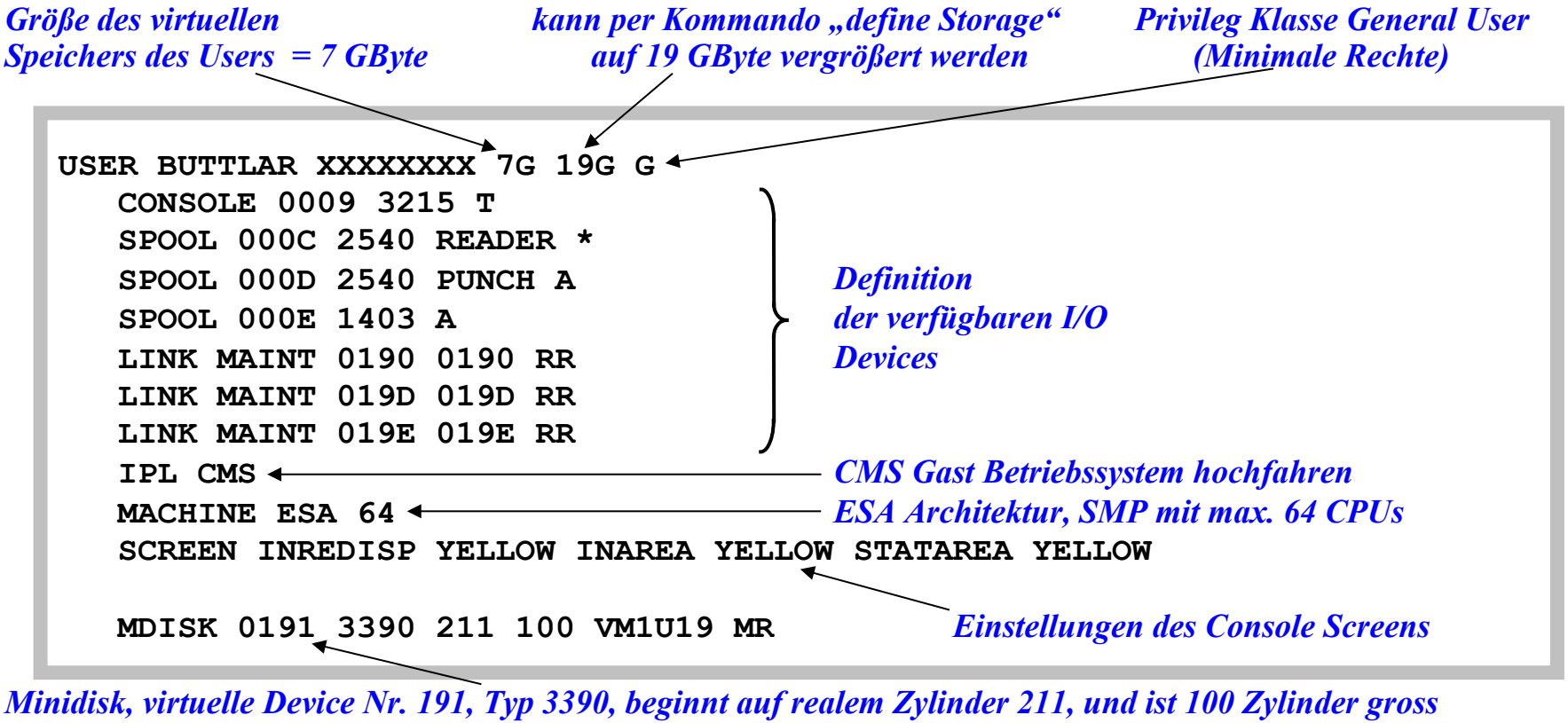
CMS ist ein besonders für die Software Entwicklung ausgelegtes Einzelplatz-Betriebssystem. Für 1000 gleichzeitige CMS Benutzer werden 1000 CMS-Instanzen angelegt.

VM/370 und z/VM erreichen eine uneingeschränkte S/390 und System z Kompatibilität für alle Gast-Betriebssysteme. Der Performance Verlust ist relativ gering (< 5%).

Plattenspeicherplatz wird allen Gastbetriebssystemen in der Form virtueller „Minidisks“ statisch zugeordnet. Hauptspeicherplatz wird dynamisch verwaltet.

Der Kernel des VM/370 Betriebssystems ist gleichzeitig der Hypervisor. Der Rechner läuft immer im Virtualisierungsmodus.

z/VM Welcome Screen



Gezeigt ist der Welcome Screen, mit dem sich ein CMS User in z/VM einloggt. Die Größe des virtuellen Speichers wird für jeden Benutzer eingeschränkt, um den Verwaltungsaufwand klein zu halten. Reader und Punch sind eine Erinnerung an die früher gebräuchlichen Lochkartenleser und –stanzer. Mit dem IPL (Initial Program Load) Kommando wird das CMS Gast Betriebssystem in den virtuellen Adressenraum dieses Benutzers geladen. Es steht eine Minidisk mit der Device Nr. 191 zur Verfügung. Sie ist auf einem realen Plattenspeicher vom Typ IBM 3390 abgespeichert, beginnt auf Zylinder 211 und ist 100 Zylinder groß.

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 Gastmaschine.

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

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

Ausführung einer I/O-Operation
Beispiel READ

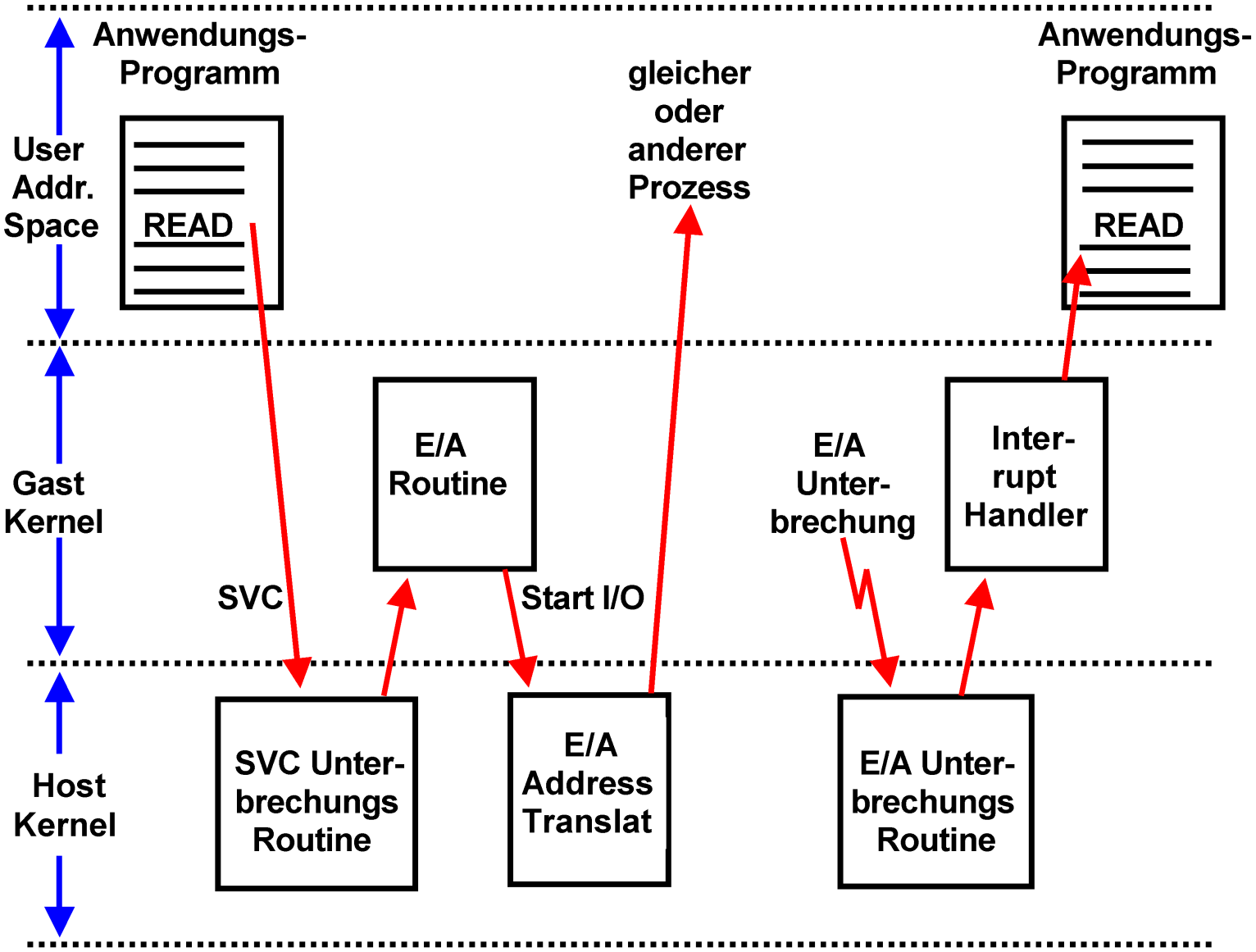
Die Ausführung einer (Plattenspeicher) READ-Operation in einem Anwendungsprogramm führt zum Aufruf einer Library Routine, welche mittels eines SVC (Supervisor Call) Maschinenbefehl in eine Routine des Kernels verzweigt. Diese führt die I/O-Operation stellvertretend für den Anwendungsprozess aus.

Eine virtuelle Maschine läuft als Prozess unter dem Host Kernel im User-Status, einschließlich des Kernels des Gastbetriebssystems. 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 Festplattenspeicher-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 Gastkernels. Letzterer benachrichtigt die Anwendung, welche die READ-Operation auslöste, von deren erfolgreichen Abschluss.



Darstellung einer I/O READ Operation, die von einem Anwendungsprogramm einer Gastmaschine ausgeführt wird

Sensitive Maschinenbefehle

Es muss verhindert werden, dass bei der Ausführung eines Maschinenbefehls innerhalb einer virtuellen Maschine das Verhalten einer anderen virtuellen Maschine beeinflusst wird.

Die Ausführung von **sensitiven Maschinenbefehlen** kann das Verhalten einer anderen virtuellen Maschine beeinflussen.

Die Ausführung von **nicht-sensitiven Maschinenbefehlen** beeinflusst nicht das Verhalten einer anderen virtuellen Maschine.

Einfachste Implementierung:

Alle sensitiven Maschinenbefehle sind gleichzeitig auch privilegierte Maschinenbefehle und können nur vom Host-Kernel ausgeführt werden. So wurde VM/370 in 1971 implementiert.

Probleme der x86 Architektur

Die Firma VMWare portierte VM/370 auf die x86-Plattform und lieferte ihr erstes Produkt in 1999 aus. Unglücklicherweise war die x86-Architektur hierfür wesentlich schlechter als die damalige S/370 Architektur geeignet:

“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.”

G.J. Popek, R.P. Goldberg: Formal Requirements for Virtualizable Third Generation Architectures. Comm. ACM, Vol. 17, Nr. 7, Juli 1974, S. 412-421.

Im Vergleich zu VM/370 sind die VMware ESX und GSX Server benachteiligt, weil einige kritische Eigenschaften in der x86-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 Status ausgeführt werden können.

Wenn ein Gast in ein Control Register schreibt, muss der Host Kernel diesen Maschinenbefehl abfangen, damit nicht das reale Kontrollregister des Hosts verändert wird. Der Host Kernel wird jetzt nur die Effekte der Instruktion für diesen Gast simulieren. Liest der Gast anschließend diese Kontrollregister 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 Kontrollregisters, der nur für den Host sichtbar ist).

Da die x86-Architektur diese Bedingung ursprünglich nicht erfüllte, war es nicht möglich, wie unter VM/370, alle Maschinenbefehle einfach im User Mode auszuführen, und auf Programmunterbrechungen zu vertrauen, wenn auf privilegierten Maschinenstatus Information zugegriffen wird.

VMware's 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; es kann sein, dass bestimmte Anwendungen nicht lauffähig sind.

Host- und Gast Modus

Jeder Rechner unterscheidet zwischen dem **Kernel** Modus und dem **User** Modus. Bestimmte sog. „Privilegierte Maschinenbefehle“ können im Kernel Modus, aber nicht im User Modus ausgeführt werden.

Zur Lösung von Virtualisierungs-Schwierigkeiten führte man zusätzlich einen **Host**- bzw. **Guest** Modus als Architekturerweiterung ein. Mainframes bezeichnen diese Erweiterung als „Interpretive Execution Facility“ (IEF). Die entsprechenden Bezeichnungen sind VT-x (Vanderpool) bei Intel bzw. AMD-V (Pacifica) bei AMD. Obwohl die drei Ansätze sehr ähnlich sind, unterscheiden sie sich in einigen Details. Besonders VT-x und AMD-V sind nicht miteinander kompatibel.

Mögliche Zustände beim Einsatz des Host/Guest Modus

	Kernel Modus	User Modus
Host Modus	+	—
Gast Modus	+	+

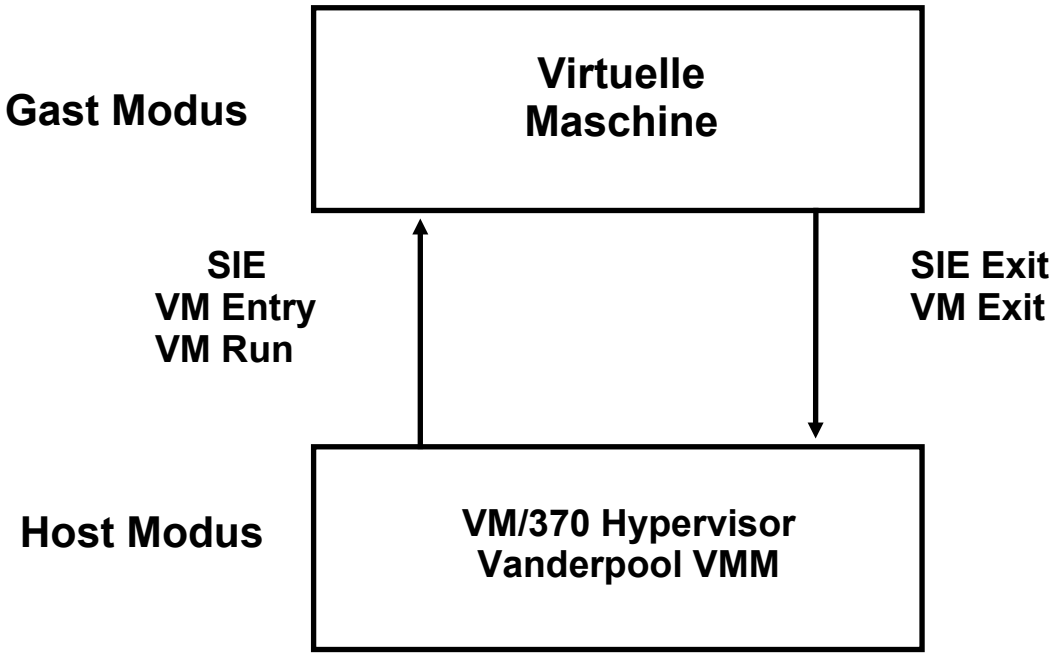
Jeder Rechner unterscheidet zwischen Kernel Modus (supervisor state) und User Modus (problem state). Das unterschiedliche Verhalten wird durch ein Bit im Status Register der CPU (Program Status Word, PSW) definiert.

Interpretive Execution Facility, Vanderpool und Pacifica führten zusätzlich noch eine Unterscheidung Host Modus/Gast Modus mit Hilfe eines zweiten Bits im Statusregister ein. Damit sind die Kombinationen Host Modus/Kernel Modus, Gast Modus /Kernel Modus und Gast Modus/User Modus möglich. Der Hypervisor läuft grundsätzlich im Host Modus; die Kombination Host Modus/User Modus hat in Bezug auf virtuelle Maschinen keine Bedeutung.

Das Problem der sensitiven Maschinenbefehle kann nun dadurch gelöst werden, dass die Hardware sich im Gast Modus anders verhält als im Host Modus. Auch kann die Gast Maschine in ihrem Kernel Modus solche privilegierten Maschinenbefehle direkt selbst ausführen, die nicht sensitiv sind – sie braucht hierfür nicht mehr die Unterstützung des Host Kernels.

Vanderpool bezeichnet den Host Kernel Modus, in dem der Host Kernel (Virtual Machine Monitor, VMM) ausgeführt wird, als *VMX Root Operation*. Jeder Gast läuft in *VMX non-Root Operation.*

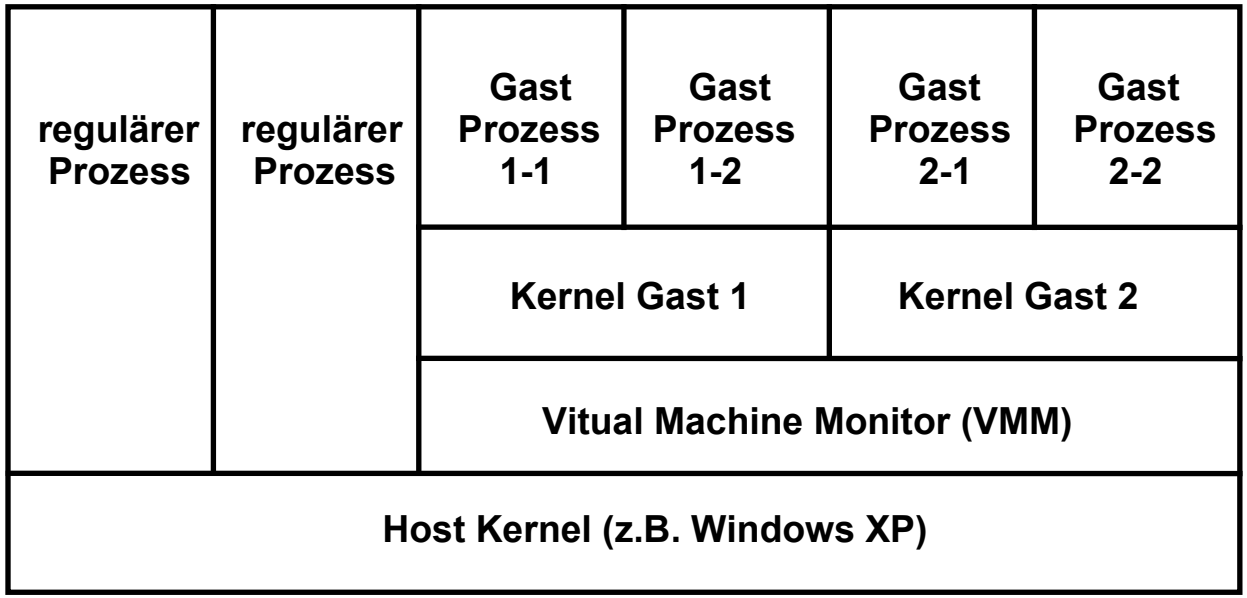
Der Aufruf eines Gastes erfolgt durch einen VM Entry Befehl (VM Run bei AMD VT, SIE bei VM/370). Die Rückkehr zum VMM erfolgt durch einen VM Exit Befehl.



Im Gastmodus können privilegierte, aber nicht sensitive Maschininenbefehle vom Kernel des Gast-Betriebssystems abgearbeitet werden. Das Problem der sensitiven nicht-privilegierten x86 Architektur der x86 Architektur wird durch zusätzliche Vanderpool- bzw. Pacifica-Hardware gelöst, die diese Befehle bei ihrer Dekodierung identifiziert und zwecks Ausführung an den VMM übergibt.

Der **Vanderpool Virtual Maschine Monitor (VMM)** ist eine Erweiterung zu einem vorhandenen Betriebssystem-Kernel, z.B. dem Windows XP-Kernel. Beide zusammen bilden den Hypervisor. Anwendungen können deshalb wahlweise im Virtualisierungsmodus unter einem Gastbetriebssystem oder nicht virtualisiert unmittelbar unter dem Host Betriebssystem laufen:

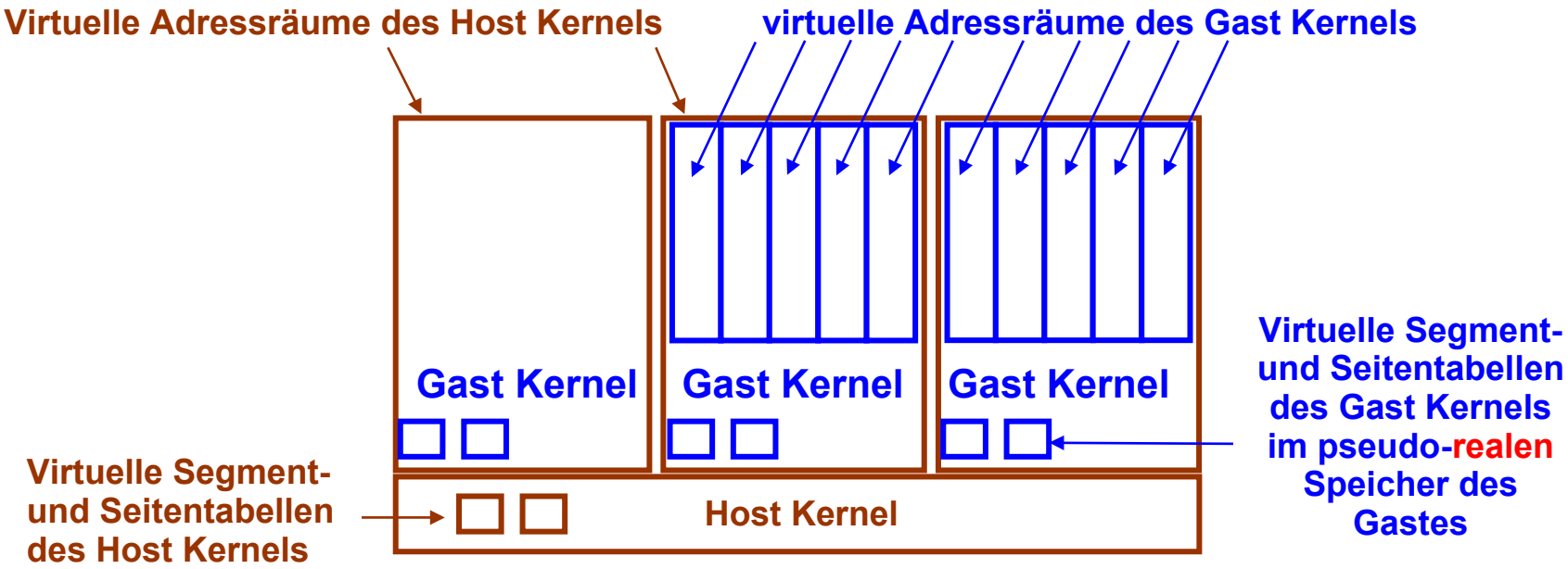
Mögliche Vanderpool Konfigurationen



Der Rechner befindet sich entweder im Virtualisierungsmodus oder auch nicht. Im Virtualisierungsmodus ist er in der Lage, „Virtual Machine Execution“ (VMX)-Operationen durchzuführen und mit Gast-Maschinen zu arbeiten. Ein VMXON Befehl versetzt den Rechner in den Virtualisierungsmodus und aktiviert den Virtual Machine Monitor; mit VMXOFF wird der Virtualisierungsmodus wieder verlassen.

Existierende Virtual Machine Monitore wie VMware, KVM, XEN, Virtual PC, Virtual Box usw. nutzen die Vanderpool bzw. Pacifica Technologie. Damit entfallen die meisten bisherigen x86-Probleme.

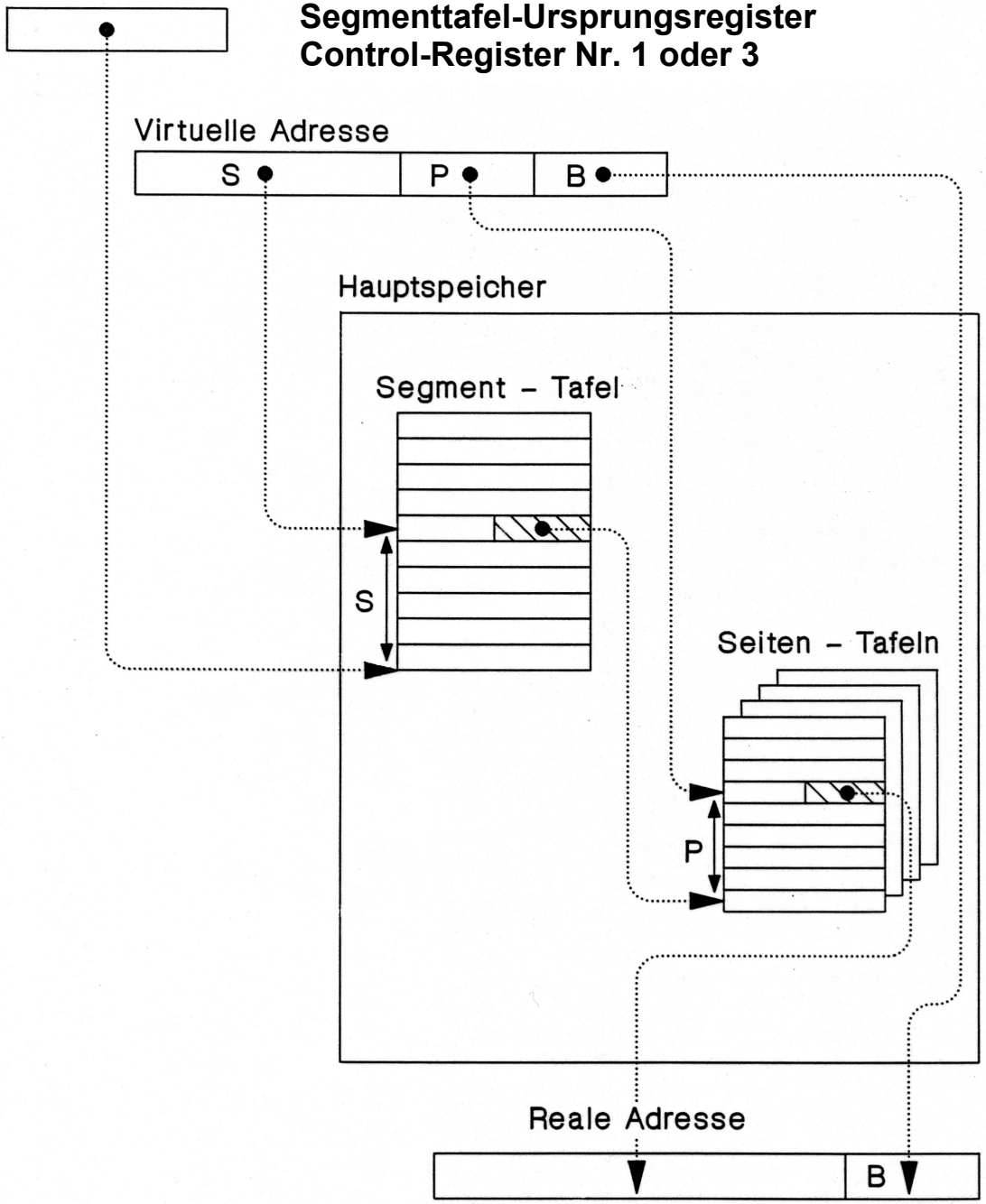
Virtuelle Maschinen mit virtueller Adressumsetzung



Angenommen, wir laden ein Gast Betriebssystem in einen (und nur einen) virtuellen Adressenraum (Address Space) des Host Betriebssystems.

Bei der Einführung von VM/370 war CMS das einzige verfügbare Gast-Betriebssystem. CMS ist ein single User Betriebssystem und verwendet keine virtuelle Adressumsetzung. Das erleichtert die Implementierung.

Praktisch alle modernen Betriebssysteme verwenden eine virtuelle Adressumsetzung. Ein solches Gast Betriebssystem glaubt, der ihm zur Verfügung gestellte virtuellen Adressenraum des Host Betriebssystems ist ein realer Speicher – hier als pseudo-realer Gastspeicher bezeichnet. Es richtet in diesem realen Gastspeicher mehrfache virtuelle Adressenräume (virtuelle Gastspeicher) ein und will seine eigenen Prozesse in diesen virtuellen Adressräumen laufen lassen. Wir brauchen eine Virtualisierung der virtuellen Adressenräume des Gastbetriebssystems!!!



Virtuelle Adressumsetzung S/390

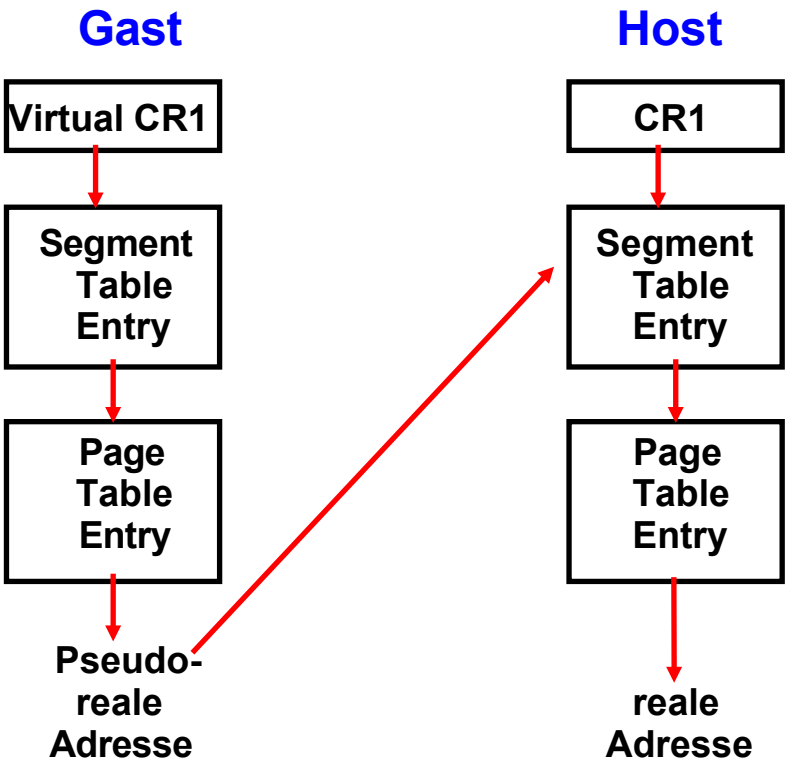
Die Adressumsetzung erfolgt durch zwei Arten von Tabellen (Segment- und Seitentabelle), die im Kernel Bereich des Hauptspeichers untergebracht sind. Die Anfangsadresse der Segmenttabelle steht in einem Control Register der Zentraleinheit, z.B. CR Nr. 1 bei der S/390 Architektur.

Die Adressumsetzung erfolgt bei jedem Hauptspeicherzugriff durch Hardware mit Unterstützung durch die Segmenttabelle und eine Seitentabelle im Kernel-Bereich. Sie kann durch den Programmierer nicht beeinflusst werden.

(Zur Leistungsverbesserung werden die derzeitig benutzten Adressen in einem Adressumsetzpuffer untergebracht.)

Problem: Eine Gastmaschine, die mit Virtueller Adressumsetzung konfiguriert ist, besteht darauf, ihre eigene Adressumsetzung durchzuführen.

Doppelte Adressumsetzung

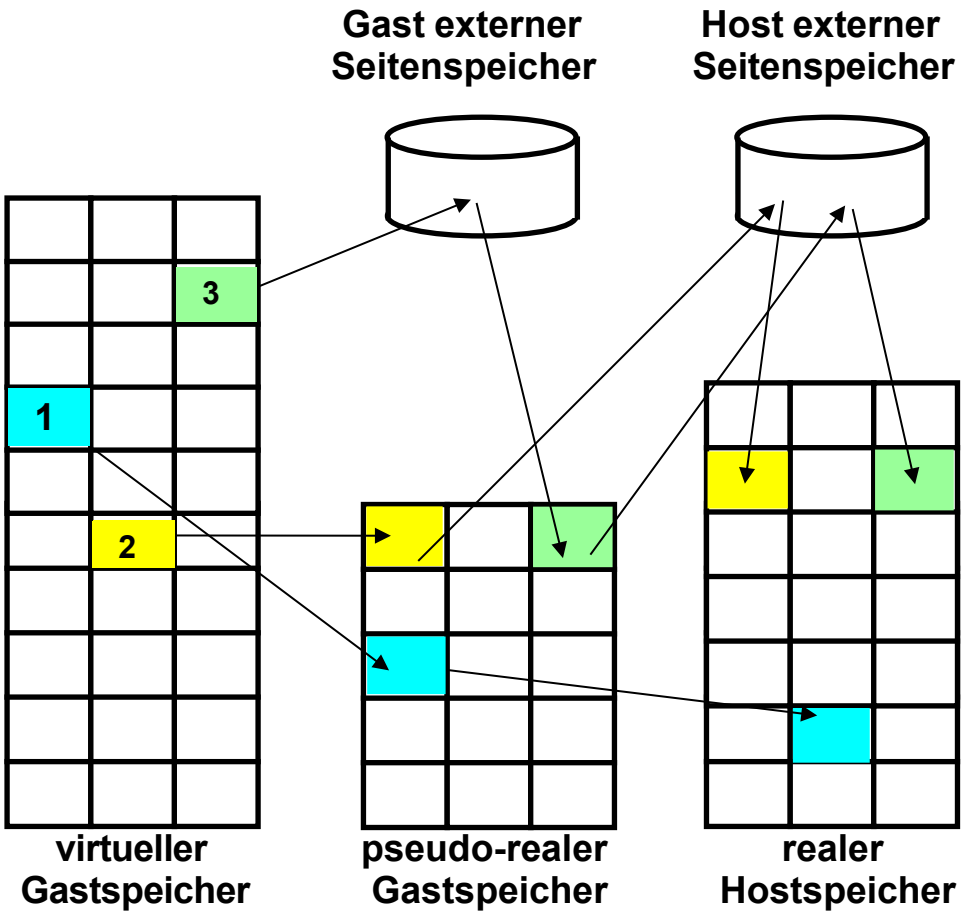


Das Gastbetriebssystem nimmt bei jedem Hauptspeicherzugriff eine virtuelle → reale Adressumsetzung vor. Es unterhält innerhalb des Gast Kernel Bereiches hierfür seine eigenen Segment- und Seitentabellen. Als Ergebnis der Adressumsetzung entsteht eine Rahmenadresse im pseudo-realen Gastspeicher.

Die Rahmenadresse im pseudo-realen Gastspeicher ist aus Sicht des Host Kernels aber eine virtuelle Seitenadresse innerhalb des virtuellen Adressenraums, den der Host Kernel der Gast Maschine als pseudo-realen Speicher zur Verfügung stellt. Bei jedem Hauptspeicherzugriff benutzt der Host Kernel das Ergebnis der Adressumsetzung des Gast Kernels, um die Rahmenadresse im pseudo-realen Gastspeicher in eine echte reale Adresse zu übersetzen. Der Host Kernel verwendet für diese Adressumsetzung seine eigenen Segment- und Seitentabellen.

Da die Adressumsetzung per Hardware erfolgt, kann die doppelte Adressumsetzung nicht funktionieren!

Adressumsetzung zwischen Gast-Kernel und Host-Kernel



Drei alternative Möglichkeiten

Wie erfolgt die Adressumsetzung der Seiten des virtuellen Gastspeichers in Rahmen des realen Hauptspeichers des Host System? Je nach dem Zustand der Seiten treten 3 Fälle auf:

1. Der gewünschte Rahmen befindet sich im (pseudo-)realen Speicher des Gastes und auch im realen Speicher des Rechners selbst (realer Hostspeicher).

2. Der gewünschte Rahmen befindet sich im pseudo-realen Speicher des Gastes, aber nicht im realen Hostspeicher. Der Host-Kernel löst eine Fehlseitenunterbrechung aus und lädt den Rahmen in den realen Hauptspeicher.

3. Der gewünschte Rahmen befindet sich nicht im realen Speicher des Gastes. Der Gast-Kernel löst eine Fehlseiten-unterbrechung aus, die vom Host-Kernel abgefangen wird. Dieser lädt den Rahmen in den realen Hauptspeicher und gleichzeitig auch in den realen Gastspeicher.

Problem der Adressübersetzung für die Virtuelle Maschine

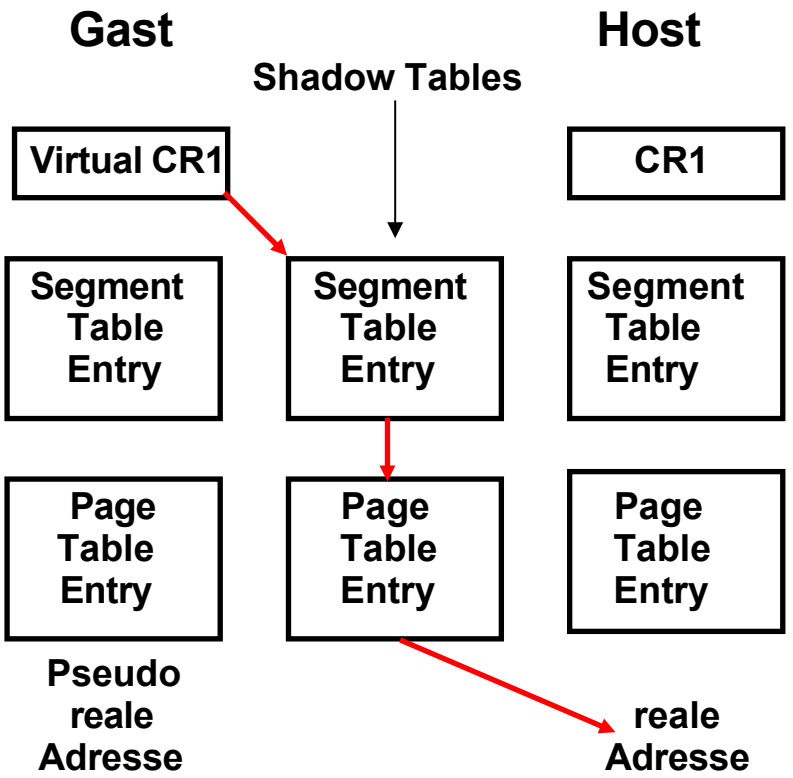
Die virtuelle Adressumsetzung mit Hilfe von Segment- und Seitentafeln erfolgt durch Hardware und kann durch Software nicht beeinflusst werden. Das hier geschilderte Verfahren kann deshalb nicht funktionieren.

Die Anfangsadresse der Segmenttafel steht im Control Register 1 (Mainframe) oder Control Register 3 (x86 Architektur). Für die Gast Maschine ist dies ein virtuelles Control Register, welches den Pointer für die Lokation der Segmenttabelle im Gast-Kernel Bereich enthält. Die Information in dem virtuellen Control Register steht aber nicht in dem physischen Control Register, sondern wird vom Host Kernel irgendwo gespeichert. Mit diesem virtuellen Control Register kann somit die Adressumsetzungs-Hardware nichts anfangen.

Erforderlich wäre eine Änderung der Hardware-Architektur, welche die hier beschrieben doppelte Adressumsetzung ermöglicht. Dies war jedoch in der ursprünglichen x86 (Pentium) Architektur nicht vorgesehen.

Zur Lösung des Problems führte man die sogenannten „Shadow Tables“ ein. Shadow Tables duplizieren teilweise die Einträge in den Segment- und Seitentabellen des Gasts und des Hosts, und müssen mit diesen im laufenden Betrieb synchron gehalten werden.

Shadow Page Tables



VM/370 und VMware erstellen anhand der Gast- und der eigenen Segment- und Seitentabellen sogenannte *Shadow Tables*, die direkt die virtuellen Adressen des Gastes auf reale Adressen des Hosts abbilden. Diese Tabellen liegen im Speicherbereich des Host-Kernels.

Wenn immer eine Gast Maschine (in Wirklichkeit ein Prozess des Host Kernels) die Verfügungsgewalt über die CPU bekommt, muss der Host Kernel die Shadow Tables mit neu berechneten korrekten Werten füllen. Bei jeder Änderung in den Segment- und Seitentabelleneinträgen des Gastes werden diese Änderungen in den Shadow Tables nachvollzogen.
Der Vorteil des Verfahrens ist, es funktioniert. Nachteilig ist, es kostet viel Performance.

Mit der Einführung der Interpretive Execution Facility, Vanderpool bzw. Pacifica kann die Rechner-Hardware für die Bedürfnisse der Virtuellen Maschinen angepasst und erweitert werden. Spezifisch sind damit die Einführung einer doppelten Adressumsetzung und die Eliminierung von Shadow Page Tables möglich.

Hiermit werden die häufigen Updates der Shadow Page Tables eliminiert. Nachteilig sind die zusätzlichen Zugriffe auf Segment- und Pagetables im Hauptspeicher. Der Aufwand hierfür ist in der Regel aber nicht so groß wie der Aufwand für die Shadow Page Table Updates. Der Grund ist, die am häufigsten benutzten Einträge in den Segment- und Seitentabellen werden in einem als DLAT (Directory Look Aside Table) oder TLB (Translation Lookaside Buffer) bezeichneten Cache Speicher abgelegt. Wenn man diesen ausreichend dimensioniert (Tausende von Einträgen), ist der Leistungsabfall vernachlässigbar.

Ende 2007 führte AMD für seine x86-Prozessoren eine neue Virtualisierungstechnologie *Rapid Virtualization Indexing* (RVI) – früher als *Nested Page Tables* bezeichnet – ein. Das Pendant *Extended Page Tables* wurde ein Jahr später von Intel mit den Nehalem-Prozessoren (erste Core i-Generation, i9xx) eingeführt. Mit RVI wird der virtuelle Adressraum eines Gastes direkt auf einen realen Adressraum des Hosts abgebildet. Dabei wird die Shadow Page Table durch eine Nested Page Table ersetzt, die vom Prozessor in Hardware unterstützt wird. Dadurch ist auch keine doppelte Adressumsetzung mehr notwendig. Eine Untersuchung von VMWare ergab, dass mit RVI ein Performancegewinn von bis zu 42% im Gegensatz zur Verwendung von softwarebasierten Shadow Page Tables erzielt werden kann. Ret Hat hat mit Verwendung von RVI eine Performanceverdopplung für OLTP-Benchmarks (Online Transaction Processing) beobachtet.

Weitere Informationen: AMD-V Nested Paging, White Paper, 2008:
http://developer.amd.com/wordpress/media/2012/10/NPT-WP-1%201-final-TM.pdf

Virtualisierung Teil 3

Logische Partitionen

Firmware

Wenn Sie einen Access Point für Ihr privates WLAN kaufen, erhalten Sie eine Box, die u.a. einen Mikroprozessor mit umfangreichem Code (häufig in EPROMs gespeichert) enthält. Das Betriebssystem ist in vielen Fällen eine Linux-Variante.

Als Benutzer können sie ein derartiges Gerät konfigurieren (z.B. eine IP Adresse eingeben). Sie können es jedoch nicht programmieren oder eigenen Code installieren.

Ein derartiges Gerät könnte ganz in Hardware implementiert sein, ohne dass Sie den Unterschied wahrnehmen würden. Mikroprozessor-Code mit derartigen Eigenschaften (vom Benutzer nicht programmierbar, änderbar oder einsehbar) wird als **Firmware** bezeichnet. Auch ein Administrator kann in der Regel Firmware nicht einsehen oder modifizieren, ggf. aber Konfigurierungs-Maßnahmen durchführen. Beispielsweise kann ein Internet-Router einen in Firmware implementierten Browser enthalten, den ein Administrator benutzen kann, um Einstellungen vorzunehmen.

HSA und Firmware

System z benutzt neben den CPUs weitere, unsichtbare Prozessoren, welche den Funktionsablauf steuern. Diese Prozessoren werden – im Gegensatz zu den CPUs – als **System Assist Prozessoren (SAPs)** bezeichnet. Beispielsweise kann ein zEC12 Rechner 101 CPUs und 16 SAPs enthalten.

Auf SAPs können keine Anwendungen laufen, und ein Benutzer kann keine Programme für SAPs schreiben. SAPs werden z.B. für die Ein/Ausgabe Steuerung, für Fehler-Behandlungsroutinen – und für Virtualisierungszwecke – benutzt.

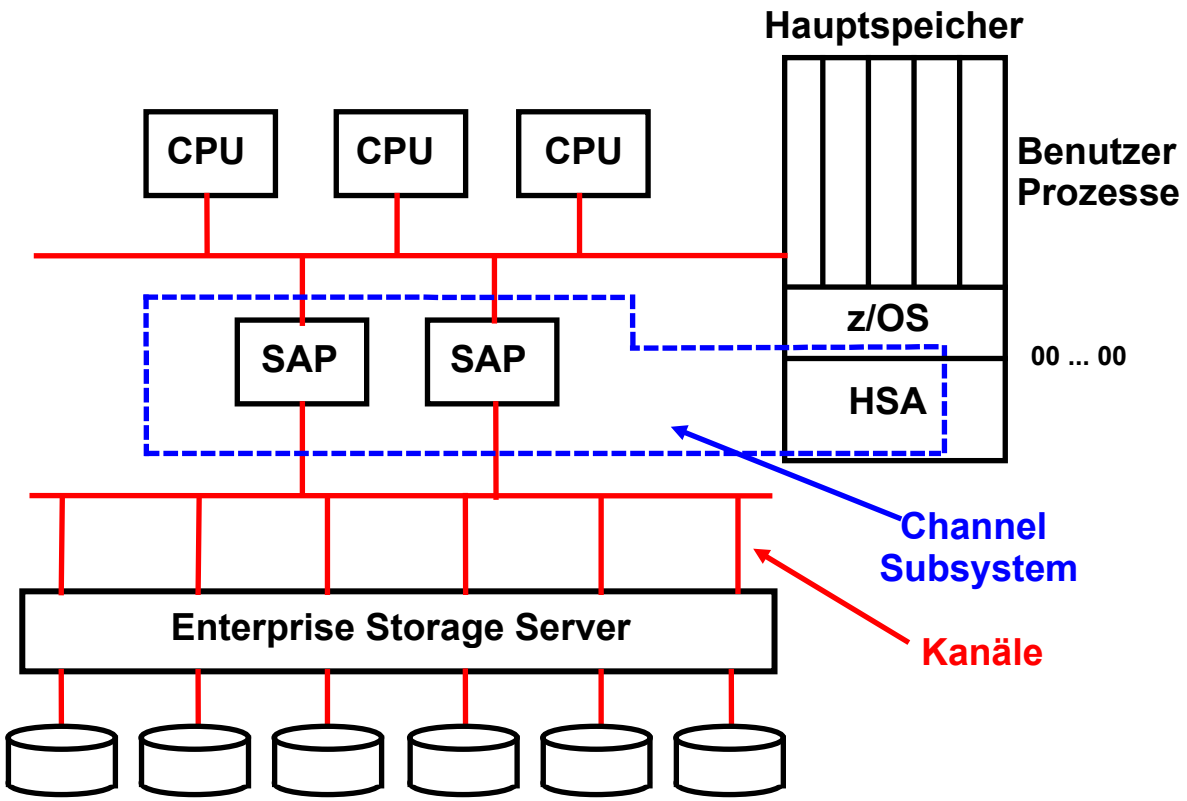
Der von den SAPs ausgeführte Code wird als **Firmware** bezeichnet. Firmware wird ähnlich wie Microcode als Bestandteil der Hardware betrachtet. Der Unterschied zwischen Firmware und Microcode besteht darin, dass Firmware reguläre System z-Maschinenbefehle unterstützt und ausführt. Der Firmware-Code befindet sich in einem getrennten Speicher, der als **Hardware System Area (HSA)** bezeichnet wird.
Firmware wird von IBM auch als „Licensed Internal Code“ (LIC) bezeichnet. Itanium (und DEC Alpha) verwendet statt LIC die Bezeichnumg PAL (Privileged Architecture Library). Itanium enthielt ursprünglich einen in PAL implementierten x86 Emulator.

Heutige Implementierungen verwenden einen einzigen physischen Speicher, der in zwei getrennte logische Speicher aufgeteilt wird:

- Einen realen Speicher, der das Betriebssystem aufnimmt und im dem die virtuellen Speicher der Benutzerprozesse abgebildet werden.
- Eine **Hardware System Area (HSA)**, die Firmware aufnimmt.

Die von den SAPs und ihrem HSA Code ausgeführte Ein/Ausgabe-Steuerung wird als Channel Subsystem bezeichnet. Es bildet das virtuelle I/O-Subsystem, mit dem der Betriebssystem-Kernel glaubt zu arbeiten, auf die reale I/O-Struktur ab. Unabhängig von System- und Benutzercode sind damit umfangreiche Optimierungen der Plattenspeicherzugriffe möglich.

zSeries Ein/Ausgabe-Anschluss

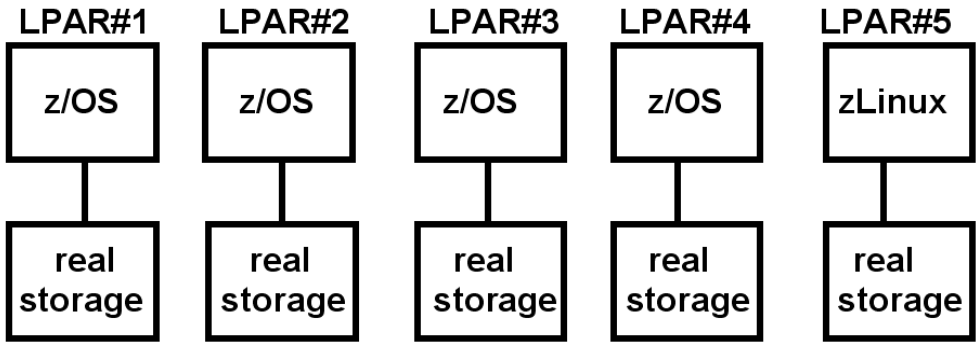


Die HSA (Hardware System Area) ist ein Teil des System z physischen Hauptspeichers. Sie liegt außerhalb des Adressenraums, auf den die CPUs zugreifen können. Der zEC12 Rechner hat eine 32 GByte große HSA.

Der in der HSA untergebrachte Code wird als Firmware bezeichnet. Er wird von SAPs ausgeführt.

SAPs führen z.B. Channel Subsystem Code aus oder implementieren Diagnostische oder Fehlerbehandlungs-Aufgaben. Auch der PR/SM Hypervisor Code wird von SAPs ausgeführt.

PR/SM und LPAR

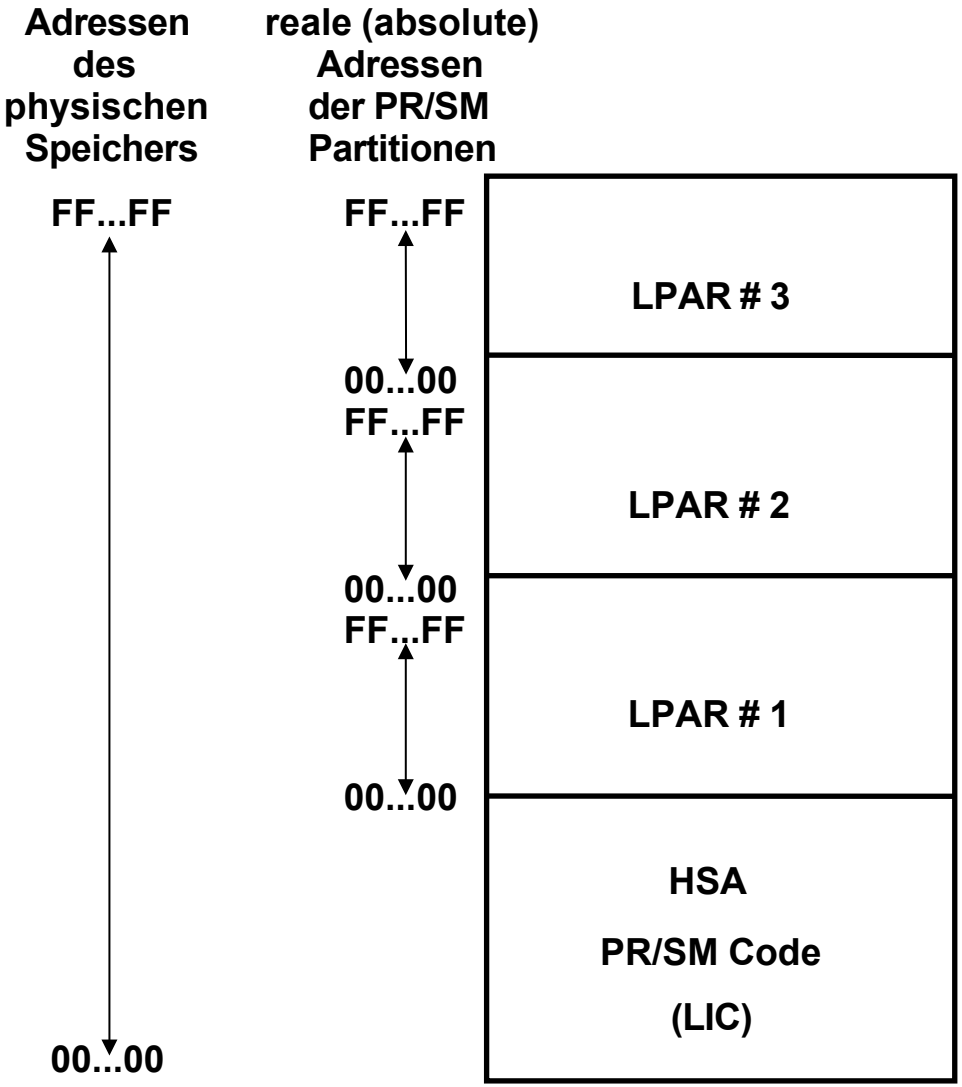


Bei den virtuellen Maschinen wird das Gast-Betriebssystem mit seinen Anwendungen in einem virtuellen Adressenraum des Host-Kernels abgebildet. Nach erfolgter virtueller Adressumsetzung teilen sich alle Gast-Systeme einen einzigen realen Adressenraum. Die Adressumsetzung bedingt Shadow Tables oder eine doppelte Adressumsetzung wie im Fall von Pacifica und der z/VM Interpretive Execution Facility. Eine direkte Zuordnung des realen Adressraumes zu einem Gast wird auch mit AMDs RVI / Intels EPT auf x86-Systemen durchgeführt.

Die System z **PR/SM (Processor Resource/System Manager)** Einrichtung verwendet ein anderes, als **Logical Partition (LPAR)** bezeichnetes Verfahren. PR/SM (ausgesprochen *pri`sm*) hat die Funktion eines Hypervisors (Host Kernel). PR/SM ist als Firmware implementiert, die in der HSA untergebracht ist und von SAPs ausgeführt wird. Für jedes Gastbetriebssystem wird eine eigene LPAR eingerichtet. Jede LPAR verfügt über einen eigenen getrennten realen Speicher. Die PR/SM Hardware stellt mehrfache (bis zu 60) LPARs mit getrennten realen Speicher zur Verfügung.

Da jede LPAR über einen eigenen realen Speicher verfügt, sind Shadow Tables oder eine doppelte Adressumsetzung nicht erforderlich.

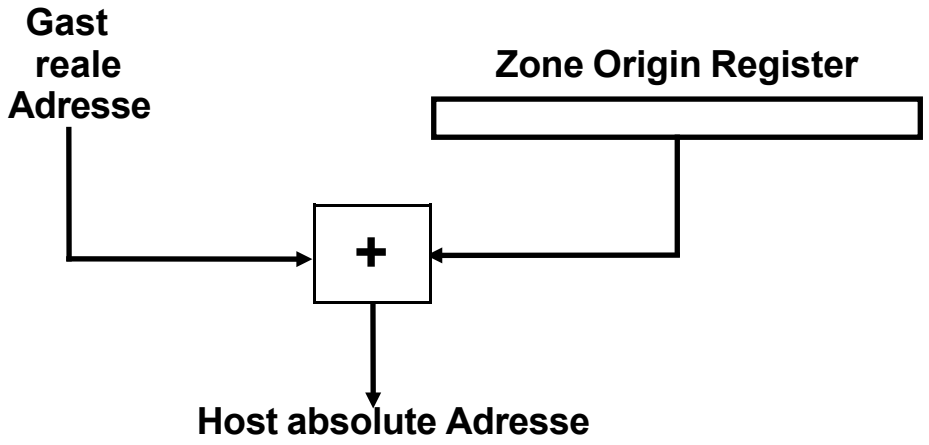
Aufteilung des physischen Speichers in mehrere reale Speicher



Es wäre denkbar, jeder LPAR einen getrennten physischen Hauptspeicher zuzuordnen. System z Rechner verwenden statt dessen einen einzigen physischen Hauptspeicher, der mittels spezieller Logik in einzelnen Partitionen aufgeteilt wird. Jede LPAR erhält eine eigene Hauptspeicher-Partition, deren kontinuierlicher Adressenbereich mit der Adresse 00 .. 00 anfängt.

In der HSA ist Firmware Code (LIC) untergebracht, der unter anderem den PR/SM Hypervisor Code enthält. Die HSA enthält weiterhin Code für die I/O Ablaufsteuerung, dazugehörige Datenstrukturen sowie Code für umfangreiche Diagnostische und Fehlerbehandlungsfunktionen

Umsetzung der Gast Adressen in physische Adressen

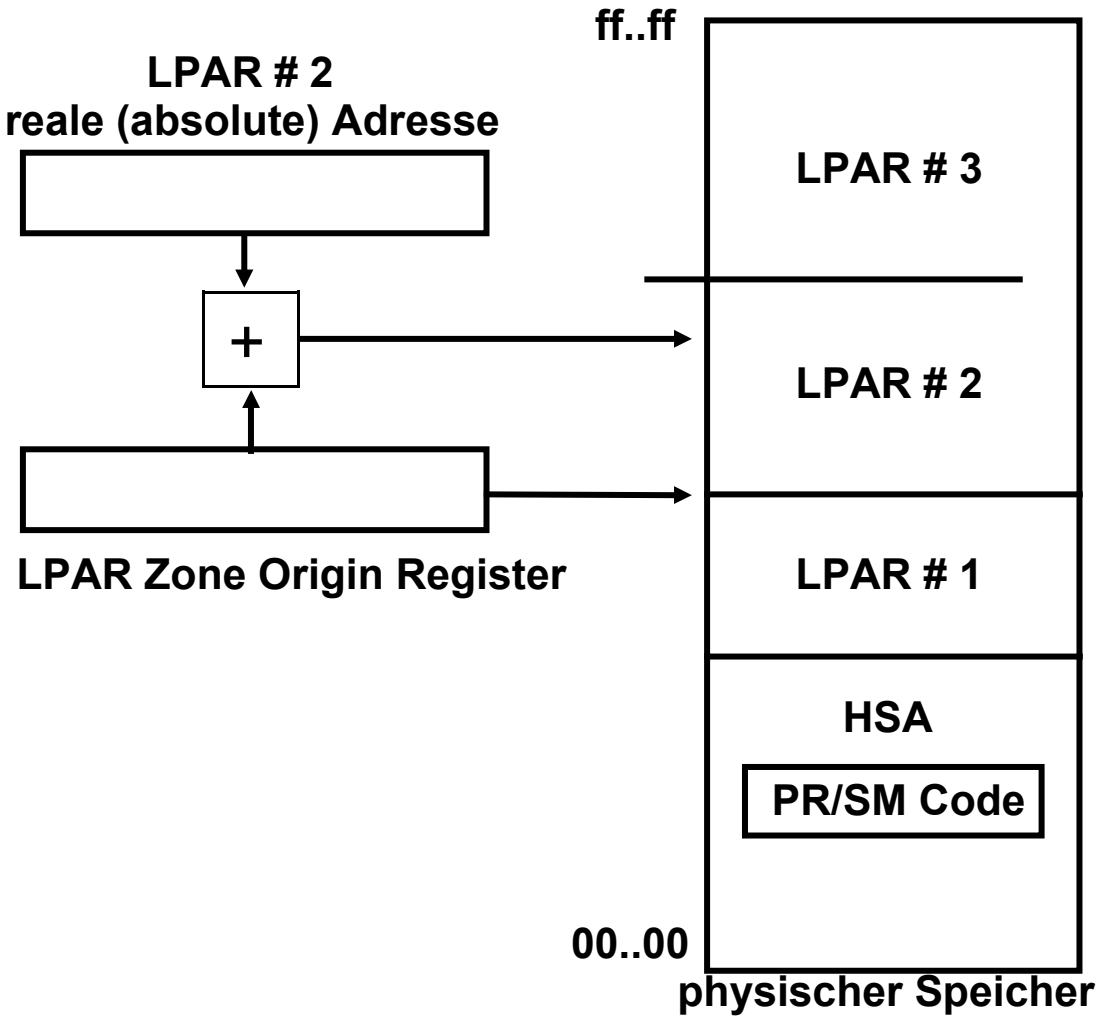


In einem System z Server sind im einfachsten Fall jeder LPAR eine oder mehrere CPUs fest zugeordnet. Jede CPU verfügt über ein „Zone Origin“ Register, in das bei der erstmaligen Konfiguration des Systems der korrekte Adressenwert geladen wird..

Ein weiteres „Zone Limit“ Register in jeder CPU stellt sicher, dass der Gast innerhalb des ihm zugewiesenen physischen Adressenbereiches bleibt.

PR/SM verfügt ähnlich wie ein Betriebssystem-Kernel über eine Zeitscheibensteuerung. Damit ist es möglich, auf einem Rechner mit nur einer einzigen CPU mehrere LPARs zu betreiben.

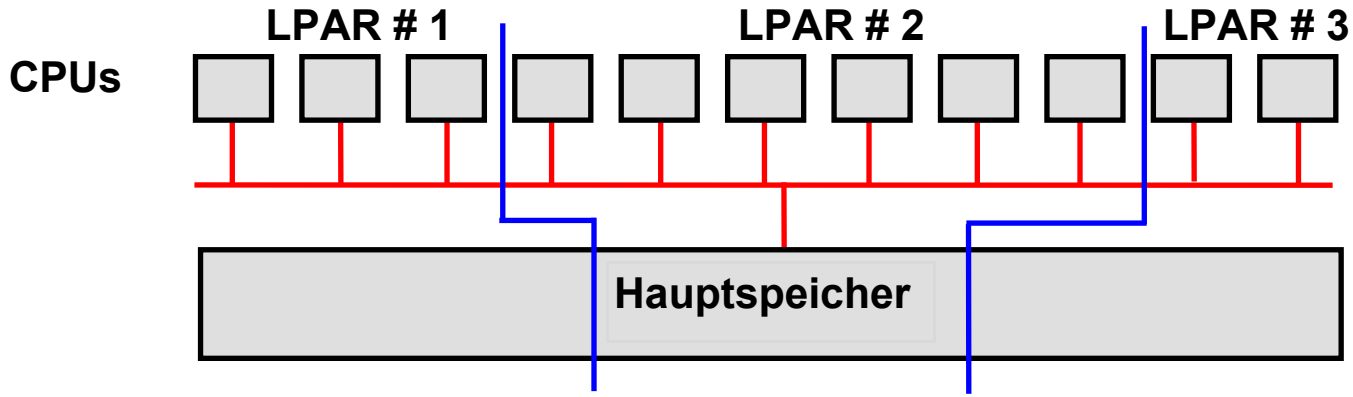
LPAR Speicherplatz-Verwaltung
Basisfall



Jede CPU hat ein LPAR Relocate Register, als LPAR Zone Origin Register bezeichnet. Hier ist die Anfangsadresse des Bereiches im physischen Speicher enthalten, der dieser LPAR zugeordnet ist.

Zusätzlich existiert ein Zone Limit Register, welches die maximale reale Hauptspeicher-Adresse definiert, die dieser LPAR zugeordnet ist.

Aufteilung eines Großrechners in mehrere SMPs



z/OS unterstützt symmetrische Multiprozessoren (SMP) mit bis zu 64 CPUs. Realistisch sind bis zu 24 – 32 CPUs. Bei Unix, Linux und Windows Betriebssystemen liegt die Grenze für Transaktions- und Datenbankanwendungen eher bei 12 CPUs.

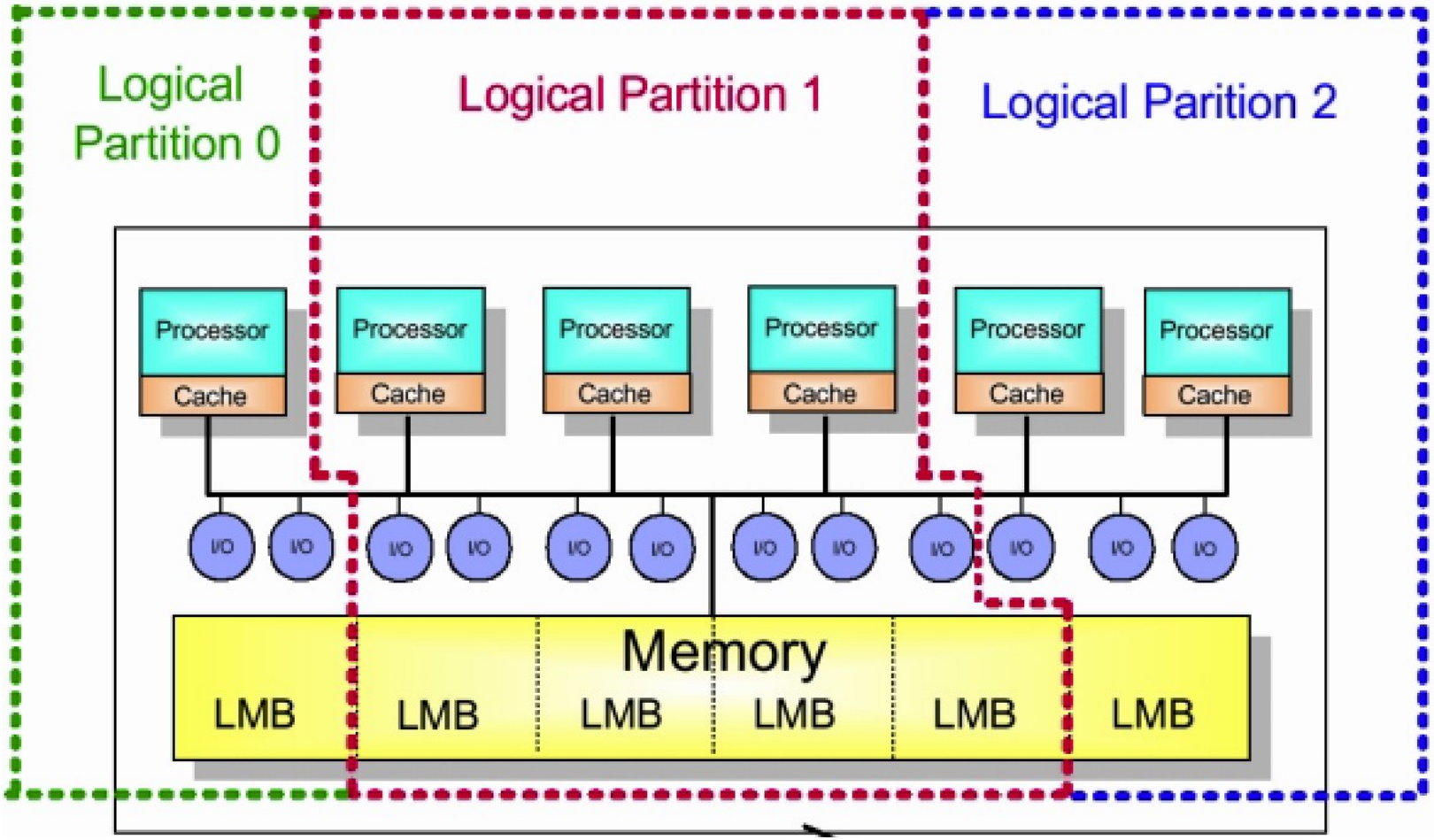
Moderne Großrechner (Systeme) verfügen über wesentlich mehr CPUs. Sie werden deshalb in mehrere SMPs aufgeteilt, die über einen zentralen Switch miteinander kommunizieren.

Die CPUs werden im einfachsten Fall auf die LPARs aufgeteilt und diesen fest zugeordnet.

Der Systemadministrator kann den gesamten Hauptspeicher in unterschiedlichen Größen auf die einzelnen LPARS aufteilen.

Die Aufteilung des physischen Speichers auf die einzelnen LPARs erfolgt (im einfachsten Fall) statisch, und wird bei der erstmaligen Inbetriebnahme des Systems festgelegt.

Logical Partition



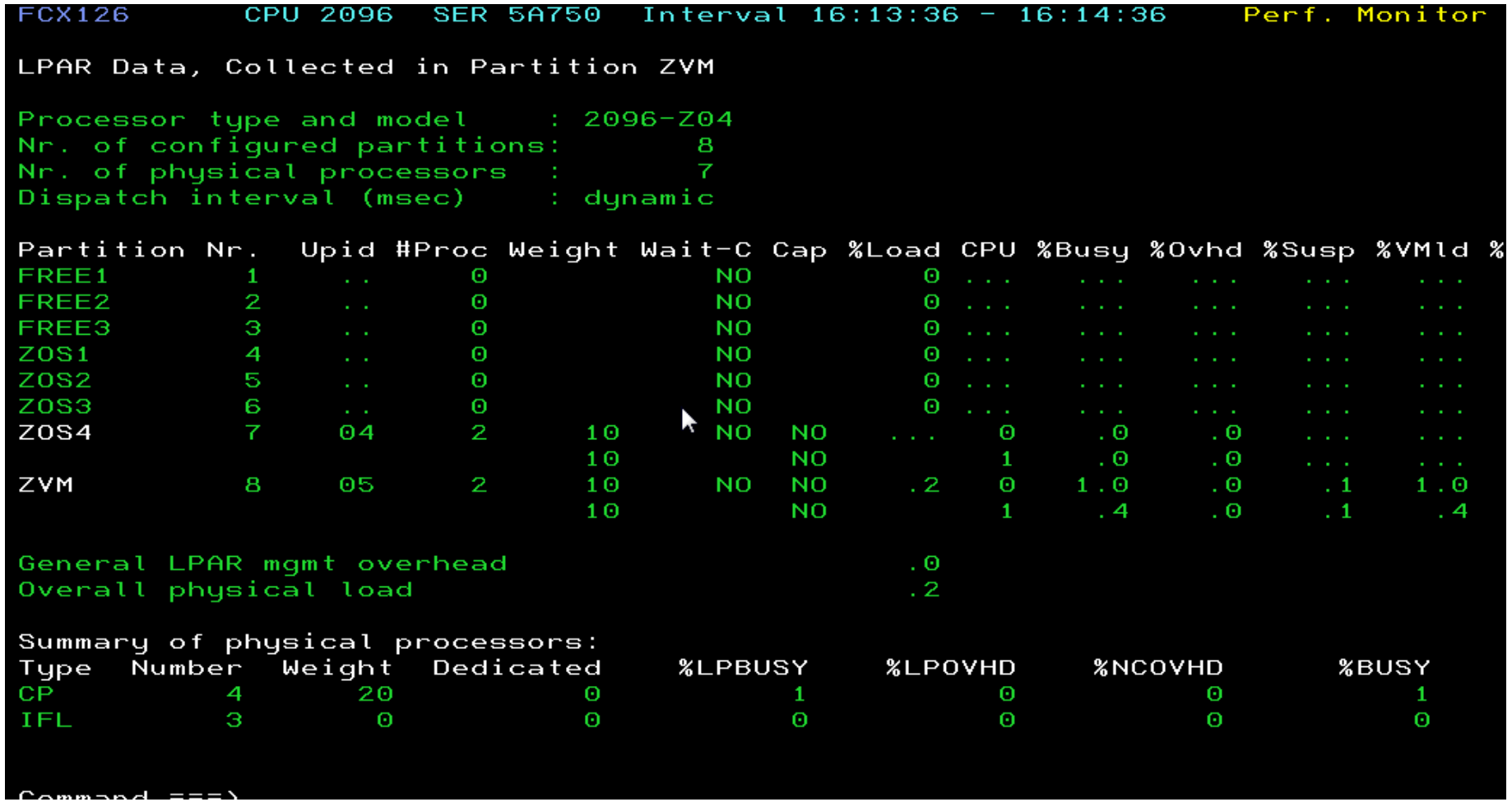
Der vorhandene physische Hauptspeicher besteht aus 64 MByte großen „Logischen Memory Blöcken“ (LMB). Die feste Aufteilung des vorhandenen physischen Hauptspeichers erfolgt in LMB Einheiten.

Die vom Channel Subsystem verwalteten I/O Anschlüsse (Kanäle) und die damit verbundenen Platten- und Magnetbandspeicher werden im einfachsten Fall den einzelnen LPARs fest zugeordnet. Das Gleiche gilt z.B. auch für Ethernet Anschlüsse. Hat ein Rechner z.B. 6 Ethernet Adapter, so kann man der ersten LPAR 4 Adapter, der zweiten LPAR 0 Adapter und der dritten LPAR 2 Adapter zuordnen.

Konfiguration des Tübinger Mainframe Computer

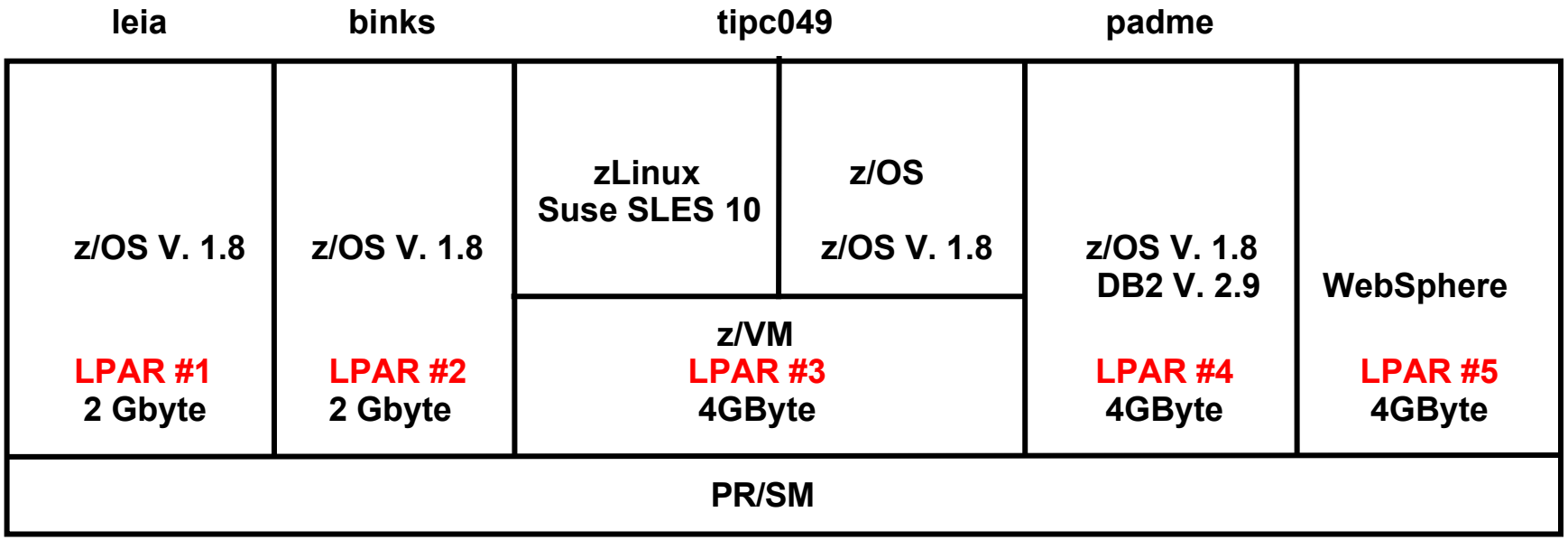
Der Tübinger Mainframe Rechner hat 8 LPAR Partitions, von denen 2 aktiviert sind:

- LPAR 7 z/OS hobbit.cs.uni-tuebingen.de 134.2.205.54 3 GB
- LPAR 8 z/VM legolas.informatik.uni.tuebingen.de 134.2.14.213 4 GB



```
FCX126      CPU 2096  SER 5A750  Interval 16:13:36 - 16:14:36    Perf. Monitor

LPAR Data, Collected in Partition ZVM

Processor type and model     : 2096-Z04
Nr. of configured partitions:        8
Nr. of physical processors   :        7
Dispatch interval (msec)     : dynamic

Partition Nr.  Upid #Proc Weight Wait-C Cap %Load CPU %Busy %Ovhd %Susp %VMld %
FREE1          1     ..     0             NO          0 ...   ...   ...   ...   ...
FREE2          2     ..     0             NO          0 ...   ...   ...   ...   ...
FREE3          3     ..     0             NO          0 ...   ...   ...   ...   ...
ZOS1           4     ..     0             NO          0 ...   ...   ...   ...   ...
ZOS2           5     ..     0             NO          0 ...   ...   ...   ...   ...
ZOS3           6     ..     0             NO          0 ...   ...   ...   ...   ...
ZOS4           7     04     2      10     NO  NO   ...    0    .0    .0   ...   ...
                            10             NO          1    .0    .0   ...   ...
ZVM            8     05     2      10     NO  NO    .2    0   1.0    .0    .1   1.0
                            10             NO          1    .4    .0    .1    .4

General LPAR mgmt overhead                        .0
Overall physical load                             .2

Summary of physical processors:
Type  Number  Weight  Dedicated    %LPBUSY    %LPOVHD    %NCOVHD       %BUSY
CP         4      20          0          1          0          0           1
IFL        3       0          0          0          0          0           0


Command ===>
```

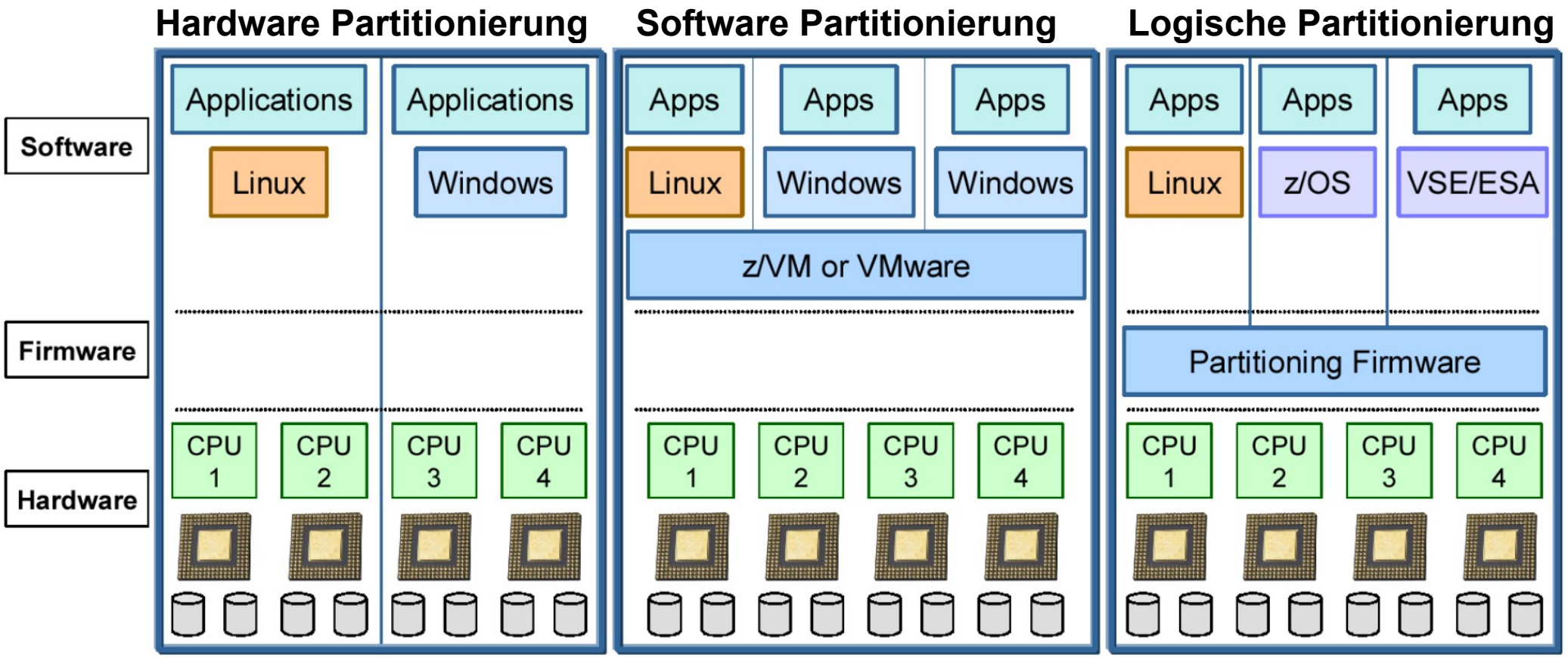
Konfiguration des Rechners jedi.informatik.uni-leipzig.de



Ein Beispiel ist die Konfiguration des Mainframe Rechners des Lehrstuhls Technische Informatik. Wir betreiben 5 LPARs. Jede LPAR hat einen eigenen Ethernet Adapter für Anschluss an das Internet:

LPAR # 1 - 139.18.4.30	**leia.informatik.uni-leipzig.de**	**z/OS 1.8**
LPAR # 2 - 139.18.4.34	**binks.informatik.uni-leipzig.de**	**z/OS 1.5**
LPAR # 3 - 139.18.4.49	**tipc049.informatik.uni-leipzig.de**	**zVM mit zLinux und z/OS**
LPAR # 4 - 139.18.4.35	**padme.informatik.uni-leipzig.de**	**z/OS 1.8**
LPAR # 5 - 139.18.4.33	**Diplomarbeiten und Forschungsprojekte**	**z/OS 1.12**

Zusammenfassung: Unterschiedliche Formen der Partitionierung



Hardware Partitioning ist unter anderem für die Großrechner der Firmen Sun und Hewlett Packard verfügbar.

Software Partitioning (auch als Virtual Partitioning bezeichnet) arbeitet mit einem Hypervisor (Host Kernel) und benötigt im Prinzip keine zusätzliche Hardware Unterstützung (Ausnahme Interpretive Execution Facility, Vanderpool und Pacifica).

Logical Partitioning benutzt Firmware für die Implementierung der Hypervisor Funktionalität. Weiterhin sind die virtuellen Maschinen nicht mehr in virtuellen Adressräumen des Host Kernels untergebracht.

PR/SM and LPAR Sicherheit und Isolation

Zertifizierung: LPARs haben äquivalente Sicherheits-Eigenschaften wie physisch getrennte Rechner.

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) stellt IBM für den Processor Resource/System Manager (PR/SM) des Mainframes das weltweit höchste Sicherheitszertifikat für einen Server aus. Die Bescheinigung nach dem internationalen Standard Common Criteria (CC) für die Stufen EAL4 und EAL5 wurde auf der CeBIT 2003 an IBM verliehen. Mainframes mit PR/SM waren damals die ersten Server, die nach der Evaluierungsstufe EAL5 für seine Virtualisierungstechnologie zertifiziert wurde.

Die Zertifizierung des BSI bescheinigt, dass Programme, die auf einem Mainframe Server in verschiedenen logischen Partitionen (LPAR) laufen, ebenso gut voneinander isoliert sind, als würden sie auf getrennten physikalischen Servern laufen.

Die Logische Partitionierung weist einzelnen Applikationen und Workloads unterschiedliche Bereiche auf dem Server zu und kann diese komplett voneinander abschirmen. So können beispielsweise Web-Anwendungen und Produktionsanwendungen, die in getrennten logischen Partitionen laufen, komplett voneinander isoliert betrieben werden, obwohl sie die physikalischen Ressourcen des zSeries-Servers gemeinsam nutzen.

IBM Presseinformation SG/94/2003, CeBIT 2003: 14. März 2003 -

Virtualisierung Teil 4

Intelligent Resource Director

Intelligent Resource Director
IRD

Der Intelligent Resource Director (IRD) ist eine Erweiterung der LPAR Technologie. Die Erweiterungen bestehen im Wesentlichen aus 4 Einzelkomponenten:

- Dynamische Verwaltung des realen Speichers
- LPAR CPU Management
- Dynamisches Channel Path Management
- Channel Subsystem Priority Queuing

Größe des realen Speichers einer LPAR

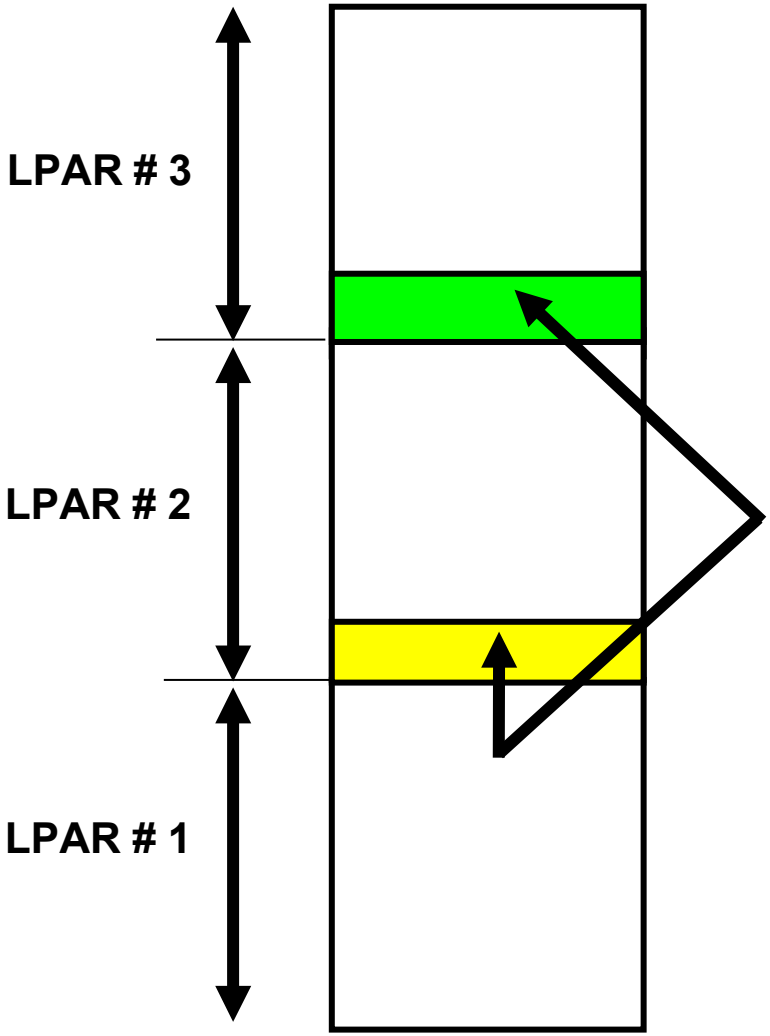
Das Betriebssystem einer LPAR nimmt an, dass sein realer Speicher über einen kontinuierlichen linearen Adressenraum, beginnend mit der Adresse 0x000000, verfügt.

Wird ein Mainframe Rechner neu installiert und konfiguriert, so wird entschieden, wie viele LPARs vorhanden sein sollen, und über wie viel realer Speicher jede LPAR verfügen soll. All diese Parameter werden in einer Konfigurationsdatei abgespeichert und bleiben in einfachen Fällen (z.B. auf unserem eigenen Mainframe Rechner an der Uni Tübingen) während der Lebensdauer unverändert.

Dies bedeutet, dass der LPAR x im physischen Speicher die Adressen von aaaa bis bbbb zugeteilt werden. Der reale Speicher der LPAR verfügt über den linearen Adressenraum von 0x000000 bis 0x00[bbbb-aaaa]. Dieser reale Adressenraum wird linear in den physischen Adressenraum von aaaa bis bbbb abgebildet. Das z/OS Betriebssystem unterstellt, dass sich bestimmte Teile des Kernels auf bestimmten realen Adressen befinden.

Will man den realen Speicher einer LPAR auf Kosten einer anderen LPAR vergrößern, ist es notwendig, alle LPARs herunterzufahren, die Konfigurationsdatei zu ändern, und die LPARs wieder hochzufahren. Es ist jedoch wünschenswert, dass dies während der z.B. 8 Jahre dauernden Lebensdauer des Rechners nie geschehen muss.

Änderung der Größe des realen Speichers einer LPAR



Die mehrfachen realen Speicher der einzelnen LPARs werden in einem einzigen physischen Speicher abgebildet. Jeder LPAR wird ein zusammenhängender (contiguous) Adressenbereich in dem physischen Speicher zugeordnet.

Wenn sich im Laufe eines Tages die Auslastung der LPARs ändert, würde man gerne einer LPAR zusätzlichen physischen Speicher auf Kosten einer anderen LPAR zuordnen.

In dem gezeigten Beispiel wäre es möglich, dass LPAR # 1 auf Kosten der benachbarten LPAR # 2 Speicherplatz erhält. Es wäre nicht möglich, dass LPAR # 1 auf Kosten der nicht benachbarten LPAR # 3 Speicherplatz erhält, weil jede LPAR glaubt, einen eigenen physischen Speicher mit einen linearen Adressenraum mit kontinuierlichen Adressen zu besitzen.

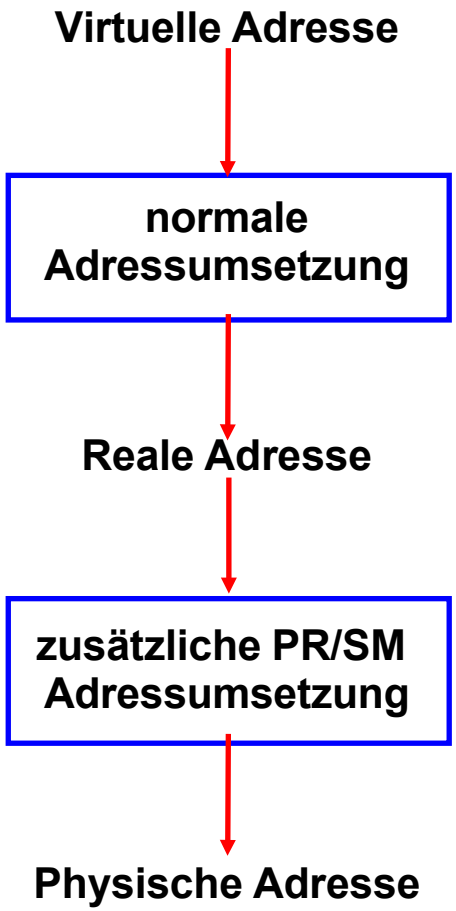
IRD Speicherplatzverwaltung

z/OS Anwendungen verwenden virtuelle Adressen, die mit Hilfe von Segment- und Seitentabellen der normalen virtuellen Adressumsetzung in reale Adressen umgesetzt werden. Die Aufteilung des physischen Speichers auf die einzelnen LPARs erfolgt in 64 MByte großen **Logischen Memory Blöcken (LMB)**.

Bei der als IRD (Intelligent Resource Director) bezeichneten Erweiterung von PR/SM kann mit einem zusätzlichen Mechanismus (der ähnlich wie die virtuelle Speicherplatzverwaltung arbeitet) der Platz, der den einzelnen LPARs zugeteilt ist, dynamisch (während des laufenden Betriebes) in LMB Blöcken von je 64 MByte vergrößert und verkleinert werden.

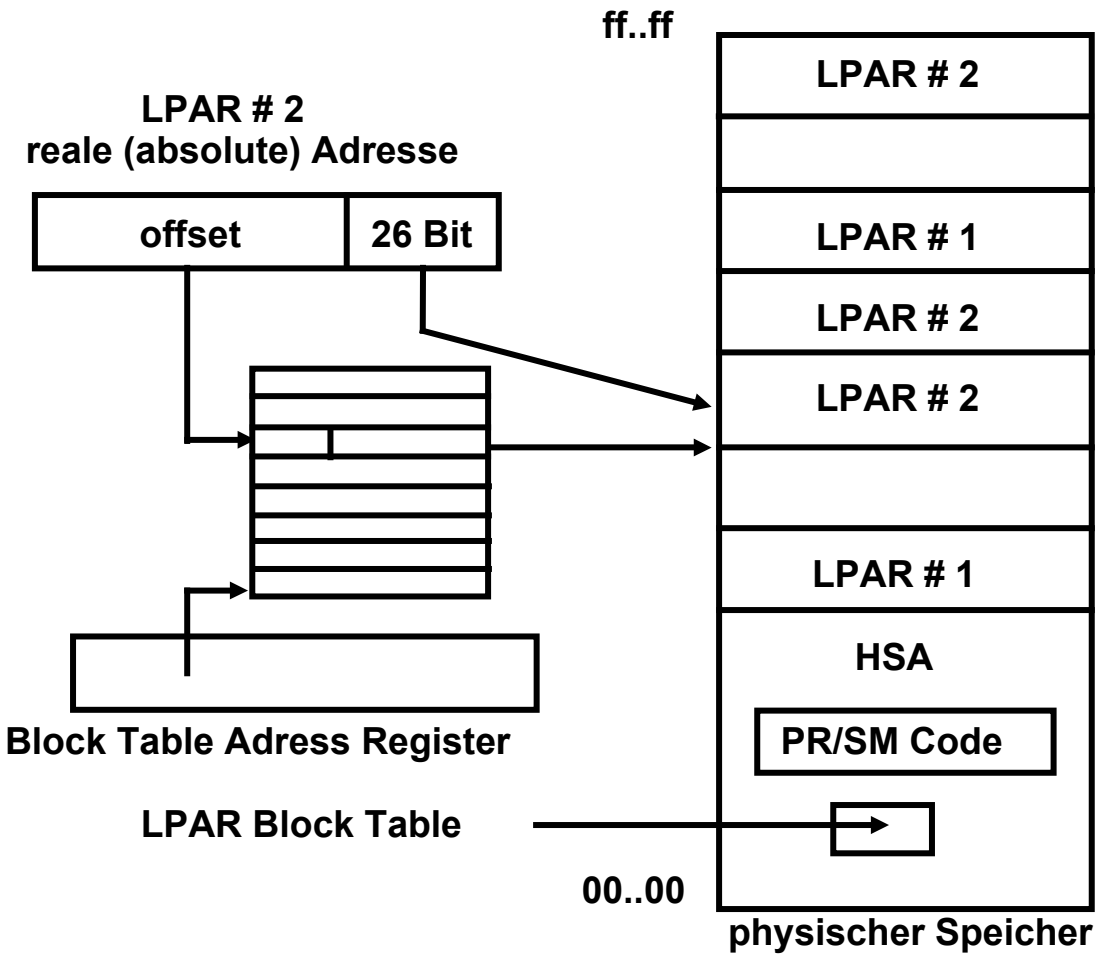
Hierbei ist nicht mehr sichergestellt, dass sich die Menge der LMBs, die einer LPAR zugeteilt sind, in einem zusammenhängenden linearen Teil des physischen Speichers befindet. Damit für die LPARs die Illusion eines kontinuierlichen linearen Adressenraums gewährt bleibt, wird ein Ansatz ähnlich der virtuellen Speicherplatzverwaltung benutzt. Dabei wird wie bei den Seitentabellen der kontinuierliche (contiguous) reale Adressenraum einer LPAR in eine diskontinuierliche Menge von 64 MByte Blöcken innerhalb des physischen Speichers mit Hilfe einer Block Tabelle umgesetzt.

Die Größe des den einzelnen LPARs zugeordneten physikalischen Speicherbereiches kann mit Hilfe einer weiteren Adressenumsetzung dynamisch variiert werden



Die Seiten und Rahmen der zusätzlichen PR/SM Adressumsetzung haben eine Größe von 64 MByte, gegenüber 4 KByte bei der normalen virtuellen Adressumsetzung

IRD Adressumsetzung



Der physische Speicher wird in 64 MByte große LMBs aufgeteilt.

An Stelle des Zone Origin Registers besteht ein Block Table Adressregister. Dieses enthält die Anfangsadresse einer Block Table. Für jede LPAR ist eine derartige Block Table in der HSA enthalten. Die unteren 26 Bit der von einer LPAR erzeugten realen Adresse zeigen auf ein Datenfeld innerhalb eines 64 MByte Blockes. Die oberen Bit der von einer LPAR erzeugten realen Adresse zeigen auf einen Eintrag der LPAR Block Table. Diese enthält die Anfangsadresse eines 64 MByte Blockes im physischen Speicher.

Die Funktion ist vergleichbar mit einer einstufigen virtuellen Address Translation. Jede LPAR hat einen linearen Adressenraum für ihre realen Adressen.

Für jeden 64 MByte Block im physischen Speicher ist ein Eintrag in der LPAR Block Table vorhanden. Deshalb ist ein Fehlseiten-vergleichbarer Mechanismus nicht erforderlich.

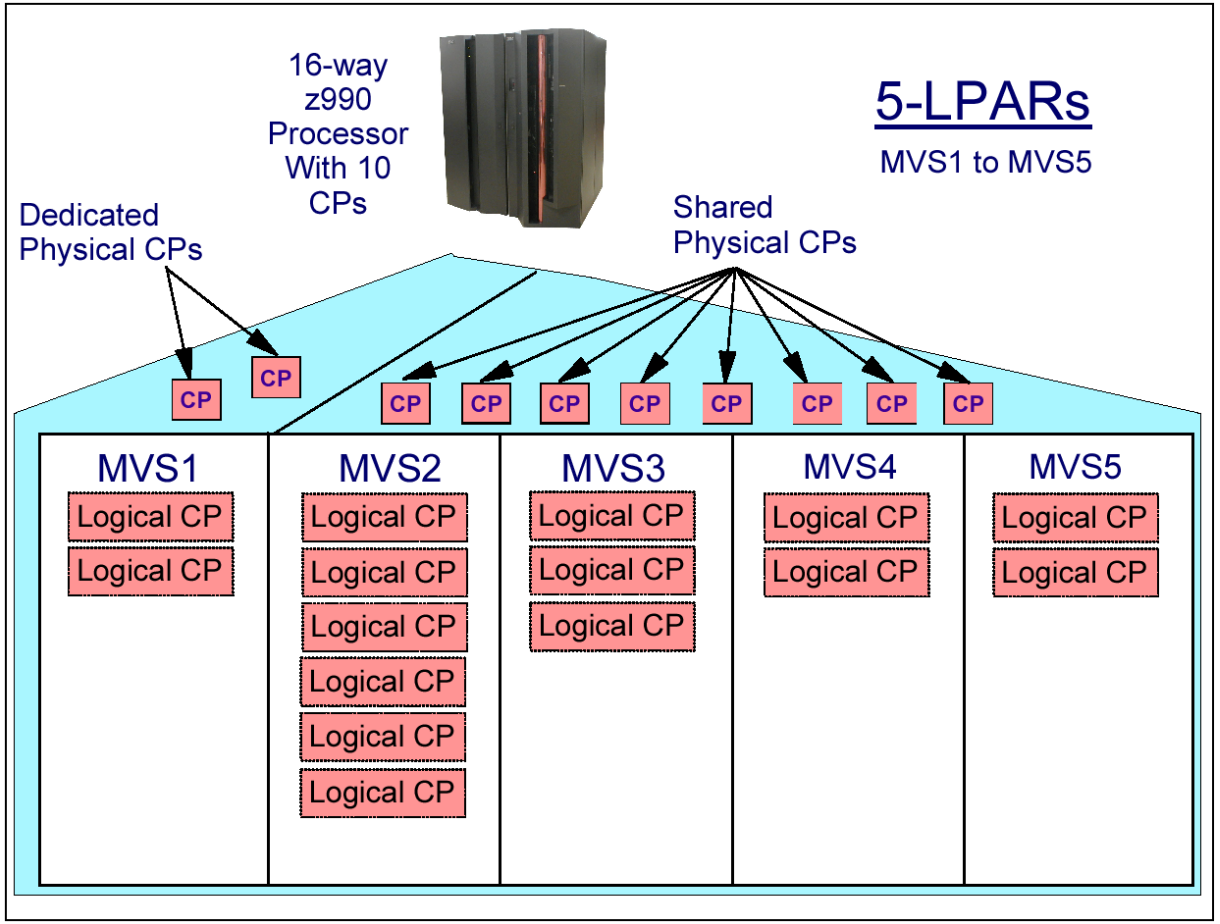
LPAR CPU Management

Bei der Systeminstallation wird ein SMP für eine definierte Anzahl von CPUs definiert. Wird beispielsweise Windows 7 auf einem neuen Quad Core PC installiert, erfolgt die Konfiguration automatisch für 4 CPUs. Bei einer Installation auf einem Dual Core PC erfolgt die Konfiguration automatisch für 2 CPUs.

In den meisten LPARs läuft in der Regel ein SMP mit mehreren CPUs. Im einfachsten Fall werden die physischen CPUs auf die einzelnen LPARs fest aufgeteilt, und das Betriebssystem wird für diese Anzahl von CPUs konfiguriert. Es wäre jedoch wünschenswert, die Aufteilung dynamisch änderbar zu machen, um bei Lastschwankungen während des täglichen oder wöchentlichen Betriebs die Auslastung der CPUs zu optimieren.

Dies kann dadurch geschehen, dass man die Betriebssysteme in den einzelnen LPARs für **logische CPUs** konfiguriert, die dann durch PR/SM auf **physische CPUs** abgebildet werden. Dabei könnte und würde die Anzahl der logischen CPUs in allen LPARs zusammen die Anzahl der physischen CPUs übersteigen.

Wenn eine CPU einer anderen LPAR zugeordnet wird, werden in das LPAR Zone Origin Register und Zone Limit Register neue korrekte Adresswerte geladen. Dies ist eine Aufgabe des PR/SM Hypervisors.

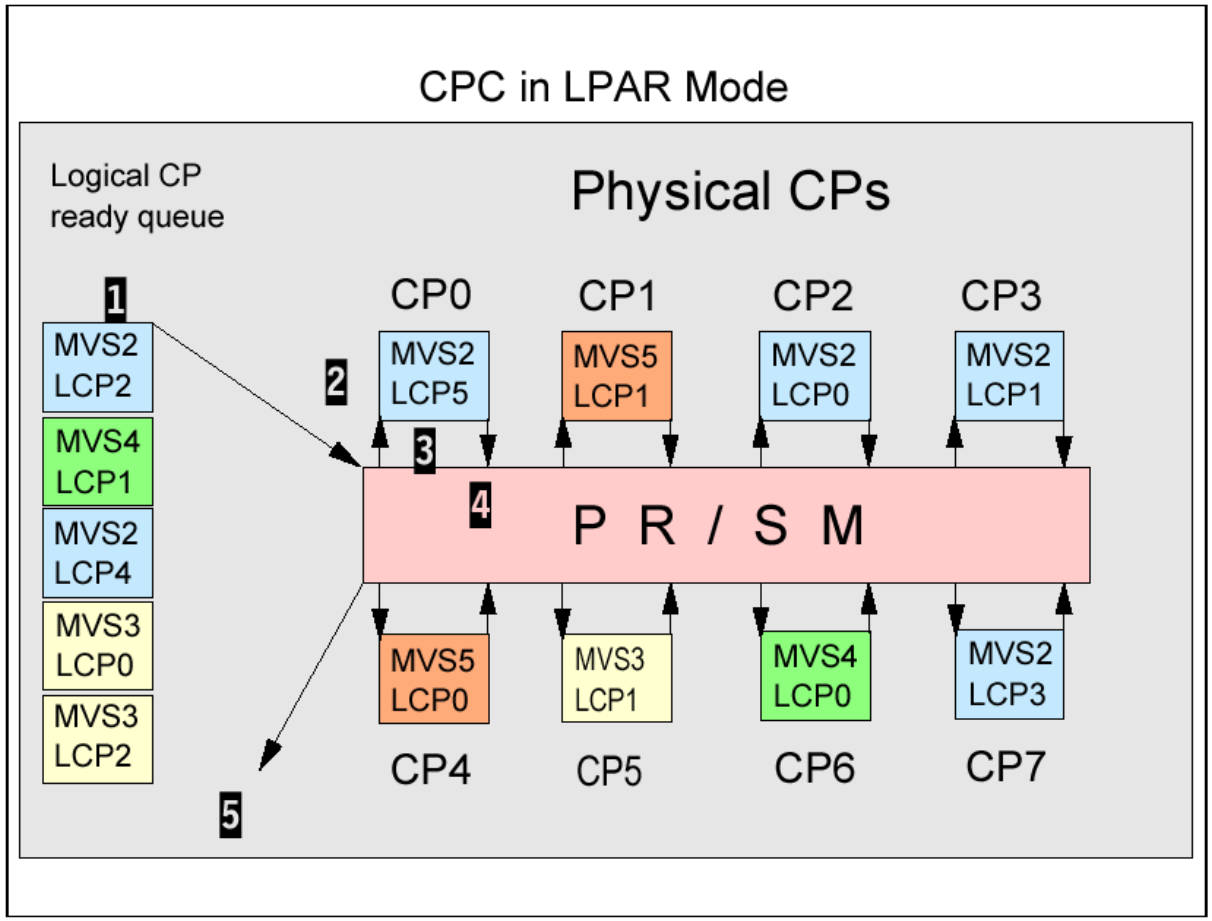


Physische CPUs (CPs) können einer bestimmten LPAR fest zugeordnet, oder von mehreren LPARs gemeinsam genutzt (shared) werden. Bei einer LPAR mit fest zugeordneten CPUs ist eine logische CPU permanent einer physischen CPU zugeordnet. Dies bedeutet weniger Overhead.

Gemeinsam genutzte (shared) physische CPUs erzeugen mehr Overhead. Dies wird überkompensiert, weil eine LPAR CPU Kapazität nutzen kann, die eine andere LPAR gerade nicht benötigt. Wenn ein Betriebssystem in den Warte- (wait) Zustand versetzt wird, gibt es die benutzten CPUs frei.

In dem gezeigten Beispiel sind der LPAR MVS1 zwei CPUs fest zugeordnet. Die vier LPARs MVS2, MVS3, MVS4 und MVS5 teilen sich in die Nutzung von acht physischen CPUs.

LPAR dispatching



Den vier LPARs mit den Betriebssystemen MVS2, MVS3, MVS4 und MVS5 stehen 8 physische CPs (CPUs) zur Verfügung (CP0 … CP7).

Jedes der Betriebssysteme ist als Multiprozessor-BS konfiguriert und glaubt, über eine bestimmte Anzahl (logischer) CPUs zu verfügen. Die Anzahl der logischen CPUs übertrifft die Anzahl der physisch vorhandenen CPUs.

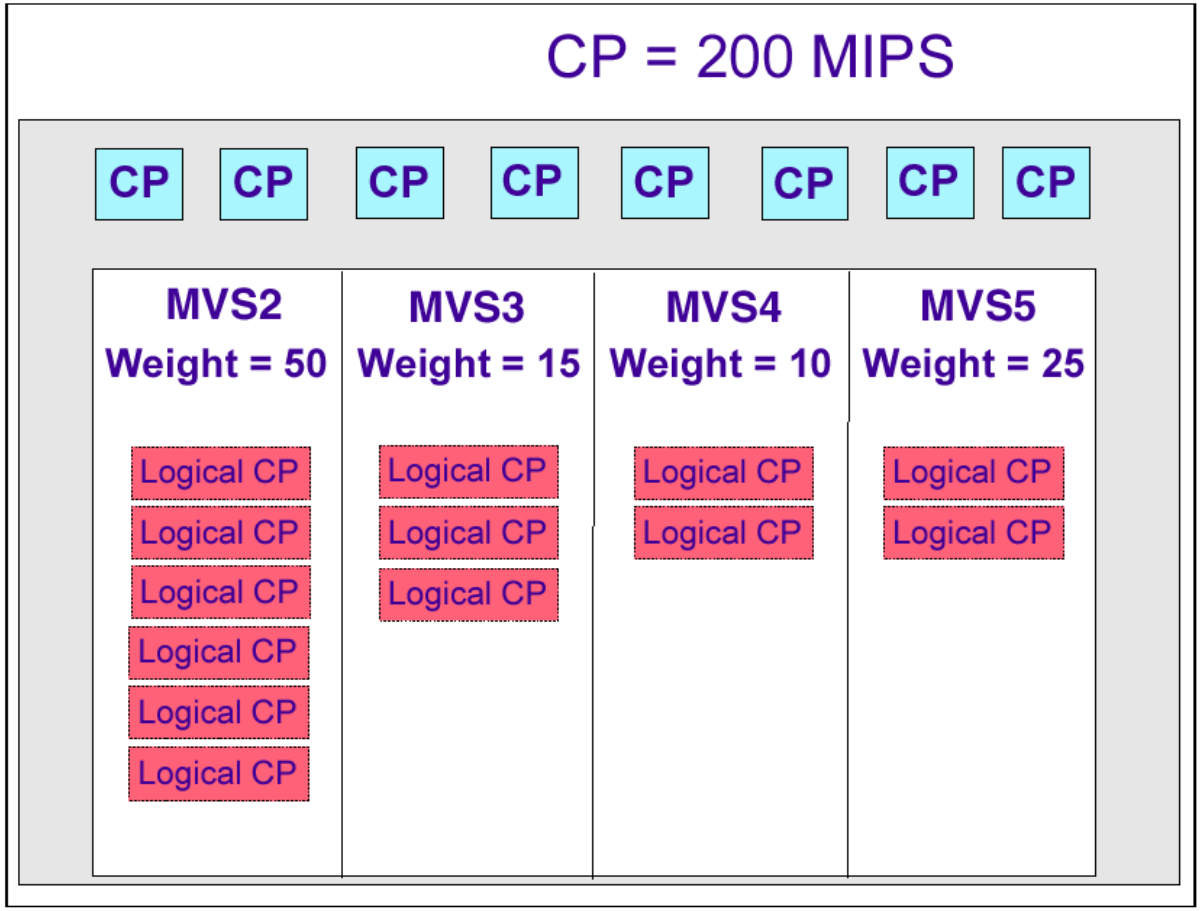
Der PR/SM Hypervisor ordnet die logischen CPs den physisch vorhandenen CPs zu. Er unterhält eine „Ready Queue“ der ausführbaren logischen CPUs aller LPARs. Wenn eine physische CPU frei wird, ordnet ihr PR/SM eine logische CPU aus der Ready-Queue zu.

LPAR dispatching

Der LPAR Dispatching Code ist Teil des PR/SM Hypervisor.

Angenommen, eine physische CPU wird frei, z.B. CP0 in dem obigen Beispiel. Das Ausführung einer logischen CPU durch CP0 erfolgt in den folgenden Schritten:

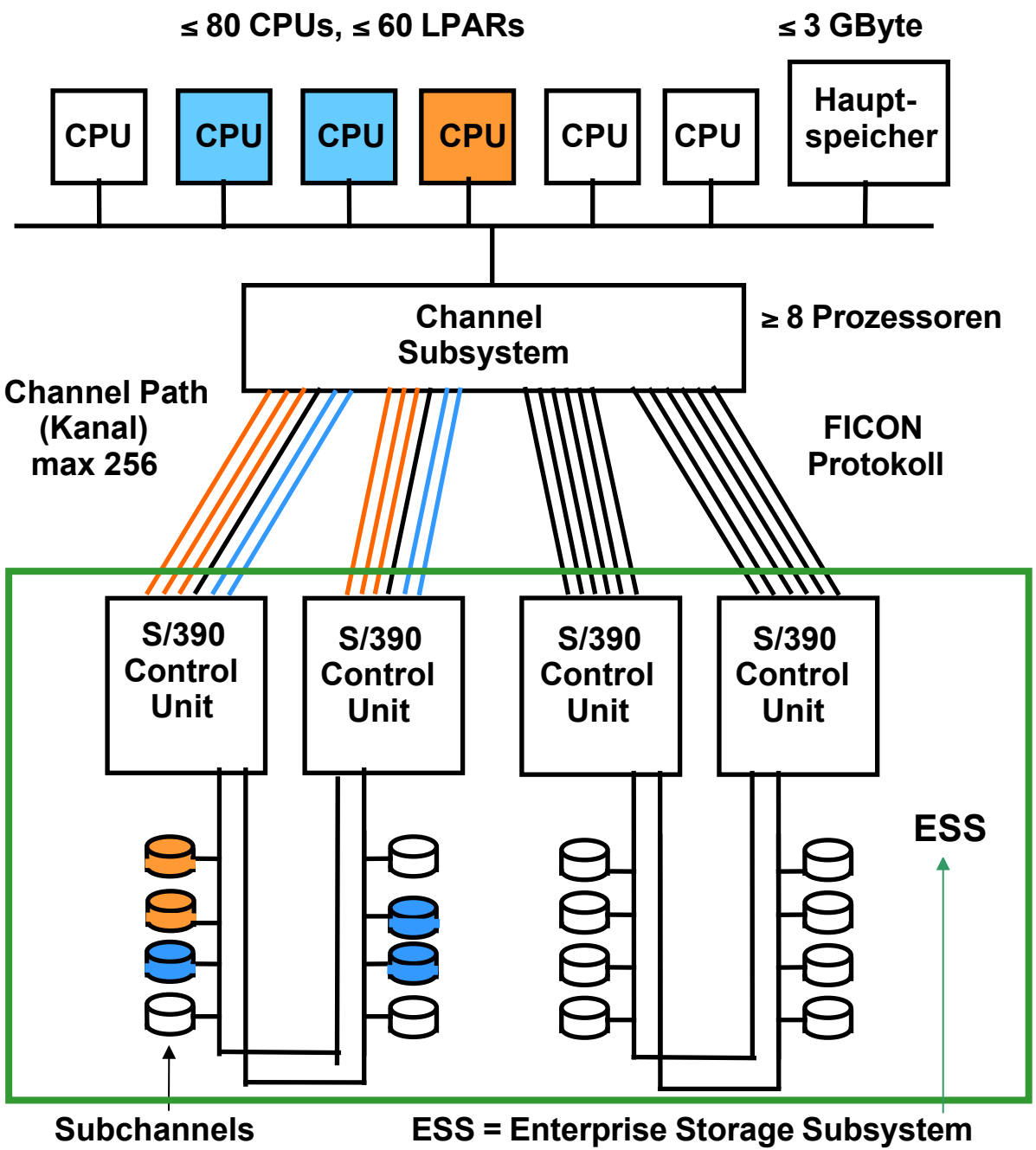
1. Die als nächstens auszuführende logische CPU wird von der logischen CP Ready-Queue ausgewählt, entsprechend dem Gewicht (Weight) der logischen CPU.
2. LPAR Firmware (LIC) ordnet die selektierte Logische CPU (LCP5 von MVS2) auf eine physische CPU (CP0 in der obigen Abbildung) zu.
3. Die z/OS Dispatchable Unit, die auf der logischen CPU (MVS2 logical LCP5) läuft, startet die Ausführung auf der physischen CP0. Sie läuft, bis ein Zeitintervall (typisch zwischen 12.5 und 25 ms) abgelaufen ist, oder ein Wait-Ereignis eintritt.
4. Bei Zeitintervallende wird die Laufzeitumgebung der logischen CP5 abgespeichert. Die Kontrolle geht zurück an die LPAR Firmware, der auf der physischen CP0 wieder startet.
5. LPAR Firmware ermittelt, warum die logische CPU die Ausführung beendete, und reiht diese z.B. in eine wartende Queue ein. Wenn LPC5 wieder ausführbar ist, wird sie wieder in die logische CP Ready-Queue eingereiht. Darauf beginnt Schritt 1 von neuem.



Mit Hilfe von **LPAR Gewichten (weights)** wird die Zuordnung von logischen CPUs auf physische CPUs gesteuert.

LPAR Gewichte bestimmen das garantierte Minimum an physischen CPU Ressourcen, welche eine logische CPU erhält, falls diese sie anfordert. Dieser garantierte Betrag ist gleichzeitig das Maximum, wenn alle logischen CPUs das garantierte Minimum voll ausschöpfen.

Eine LPAR verbraucht möglicherweise weniger als das garantierte Minimum, wenn nicht ausreichend Arbeit anfällt. In diesem Fall steht den anderen LPARs mehr CPU Leistung zur Verfügung.

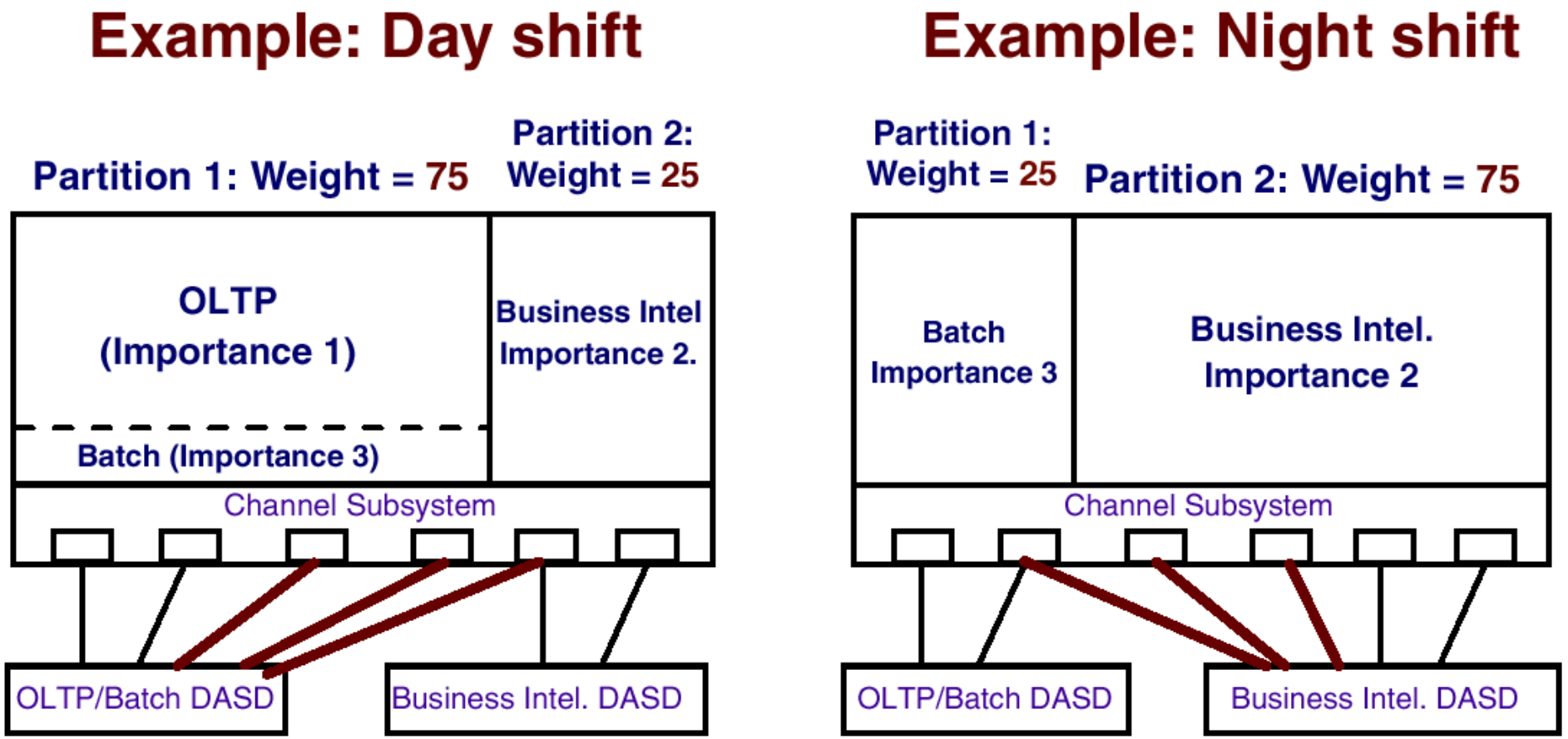


Dynamic Channel Path Management (DCM)

Festplattenspeicher werden über ein Enterprise Storage Subsystem (ESS) angeschlossen, welches S/390 Control Units abbildet.

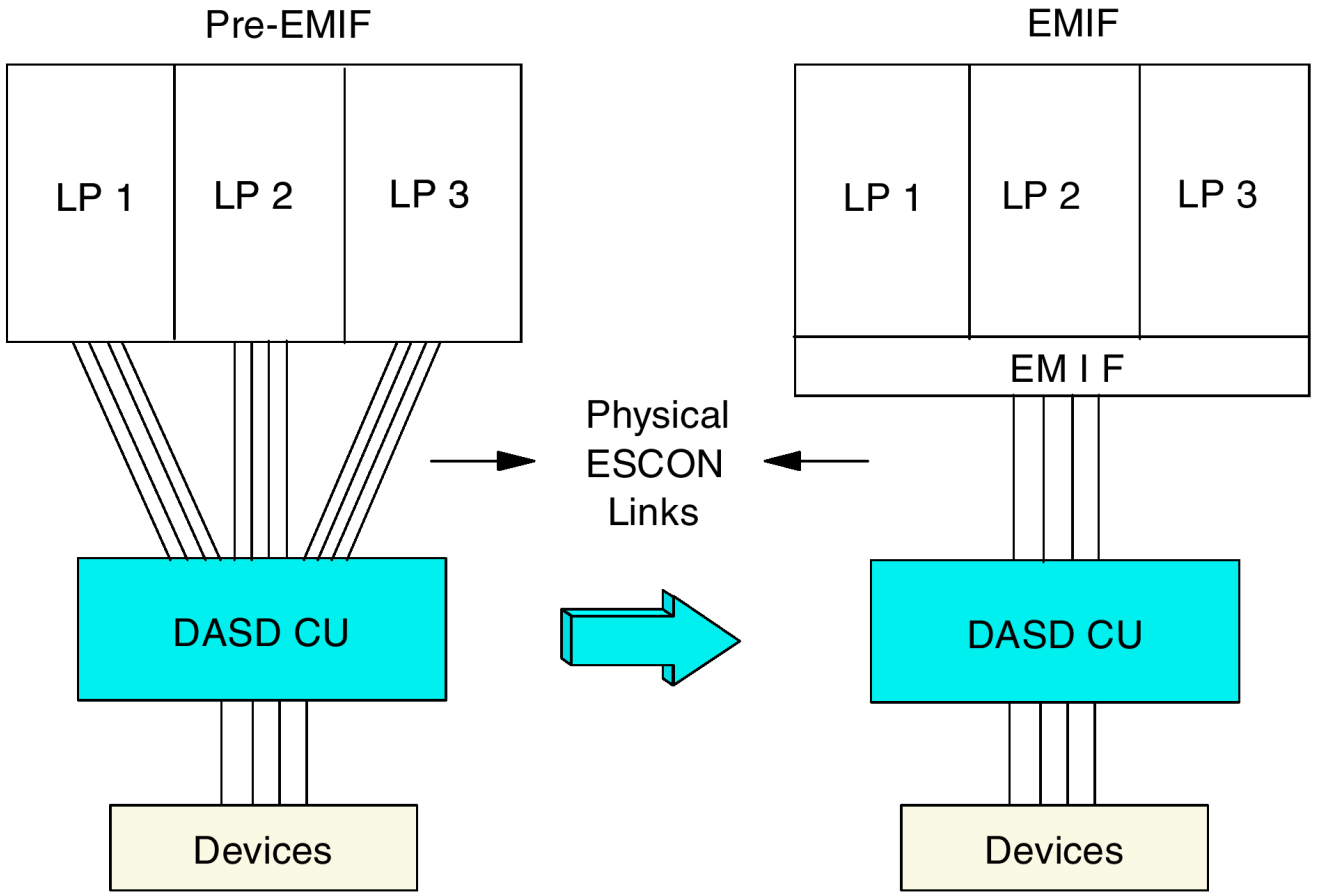
Dynamic Channel Path Management (DCM) ist in der Lage, die Kanal Configuration dynamisch an sich ständig ändernde Auslastungen anzupassen.

Dynamic Channel Path Management (DCM)



In dem gezeigten Beispiel sind während der Tagesschicht die Mehrzahl der Kanäle den OLTP (Online Transaction Processing) Anwendungen zugeordnet. Während der Nachtschicht sind die Mehrzahl der Kanäle einer LPAR für Batch Processing Anwendungen (z.B. Business Intelligence) zugeordnet.

Multiple Image Facility (MIF)



Im einfachsten Fall sind die Kanäle einzelnen LPARs fest zugeordnet.

Mit Hilfe der (extended) Multple Image Facility (EMIF) können Kanäle von mehreren LPARs gemeinsam genutzt werden. Damit kann MIF die Kanäle bestimmten LPARs unter Prioritätsgesichtpunkten dynamisch zuordnen.

An Stelle der Bezeichnung MIF wurde früher auch die Bezeichnung EMIF verwendet.

Channel Subsystem I/O Priority Queuing

Normalerweise werden I/O Anforderungen vom Channel-Subsystem auf einer First-In-First-Out-Basis abgearbeitet.

Wenn eine Arbeitseinheit mit hoher Priorität ihre Bearbeitungsziele wegen Überlastung der I/O Einrichtungen nicht erreicht, kann der **z/OS Work Load Manager** dieser Arbeitseinheit eine höhere Priorität als anderen Arbeitseinheiten zuordnen.

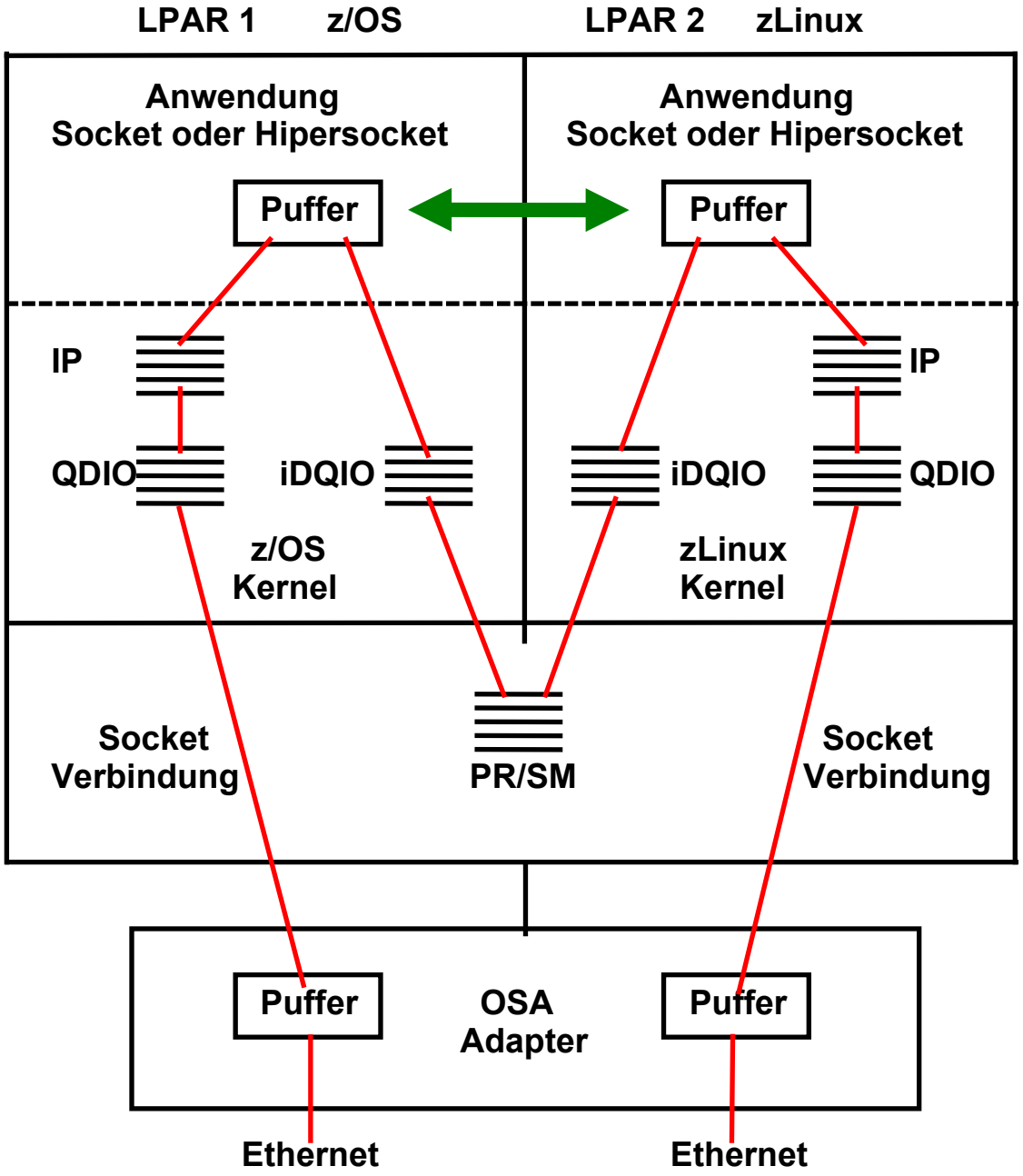
Die **Channel Subsystem Priority Queuing** Einrichtung des Channel Subsystems wird dann diese Arbeitseinheit bevorzugt abfertigen, indem sie hierfür zusätzliche Bandbreite auf Kosten anderer Arbeitseinheiten mit weniger Priorität verfügbar macht.

Hierfür werden Dynamic Channel Path Management und MIF eingesetzt.

Internal Queued Direct I/O

Trotzdem ist es manchmal erwünscht, dass LPARs Nachrichten miteinander austauschen. Hierzu ordnen wir jeder LPAR einen Ethernet Adapter zu.

Ethernet Ports sind in der Form einer I/O Adapter Karte verfügbar. Die „OSA“ Adapter Card unterstützt vier Gbit Ethernet Ports oder zwei 10 Gbit Ethernets Ports. Die Ports können unterschiedlichen LPARs zugeordnet werden. Die OSA Adapter Card enthält außerdem die Funktion einer Contol Unit. Ein als „Queued Direct I/O“ (QDIO) bezeichneter I/O Driver stellt die Verbindung zu z/OS her. PR/SM benutzt das OSI Schicht 5 Internet „Socket“ Protokoll für die Ethernet Verbindung.



Internal Queued Direct I/O

Dargestellt sind zwei LPARs, die eine Nachricht austauschen möchten.

Da sich die LPARs wie getrennte physische Rechner verhalten, kann man jeder LPAR einen Ethernet Adapter Port zuordnen, und die beiden Ports über Sockets und einem Ethernet Kabel miteinander verbinden.

System z verfügt über den OSA Adapter, der als eine Reihe von Ethernet Adapter Ports konfiguriert werden kann.

Das z/OS Communication Manager Subsystem unterstützt eine Ethernet Verbindung mit der „Queued I/O Access Method“ (QIO).

LPARS auf dem gleichen Rechner können stattdessen Internal Queued Direct I/O (iDQIO) verwenden.

HiperSockets

HiperSockets ermöglichen es mehreren LPARs innerhalb des gleichen Rechners (oder Sysplex) miteinander zu kommunizieren, ohne auf ein externes, physisches Netzwerk zurückgreifen zu müssen (LPAR-übergreifende Kommunikation). Die Hochgeschwindigkeitskommunikation über HiperSockets kann auch für die Kommunikation zwischen z/OS, z/VM und „Linux on System z“ in unterschiedlichen LPAR Instanzen eingesetzt werden. Mit dieser Funktion lässt sich innerhalb des Systems ohne eine zusätzliche physische Verbindung ein "systeminternes Netz" aufbauen.

Da die Kommunikation den Hauptspeicher benutzt, erfolgt sie mit Hauptspeicher-Zugriffsgeschwindigkeit. Es bestehen zusätzlich Sicherheitsvorteile, weil die Möglichkeit einer Netzwerk Interception (z.B. Man-of-the-Middle Attacke) von außen entfällt. HiperSockets verbessern Reliability und Availability, weil Network Hubs, Routers, Adapters oder Kabelverbindungen alle im Hauptspeicher virtuell dargestellt werden.

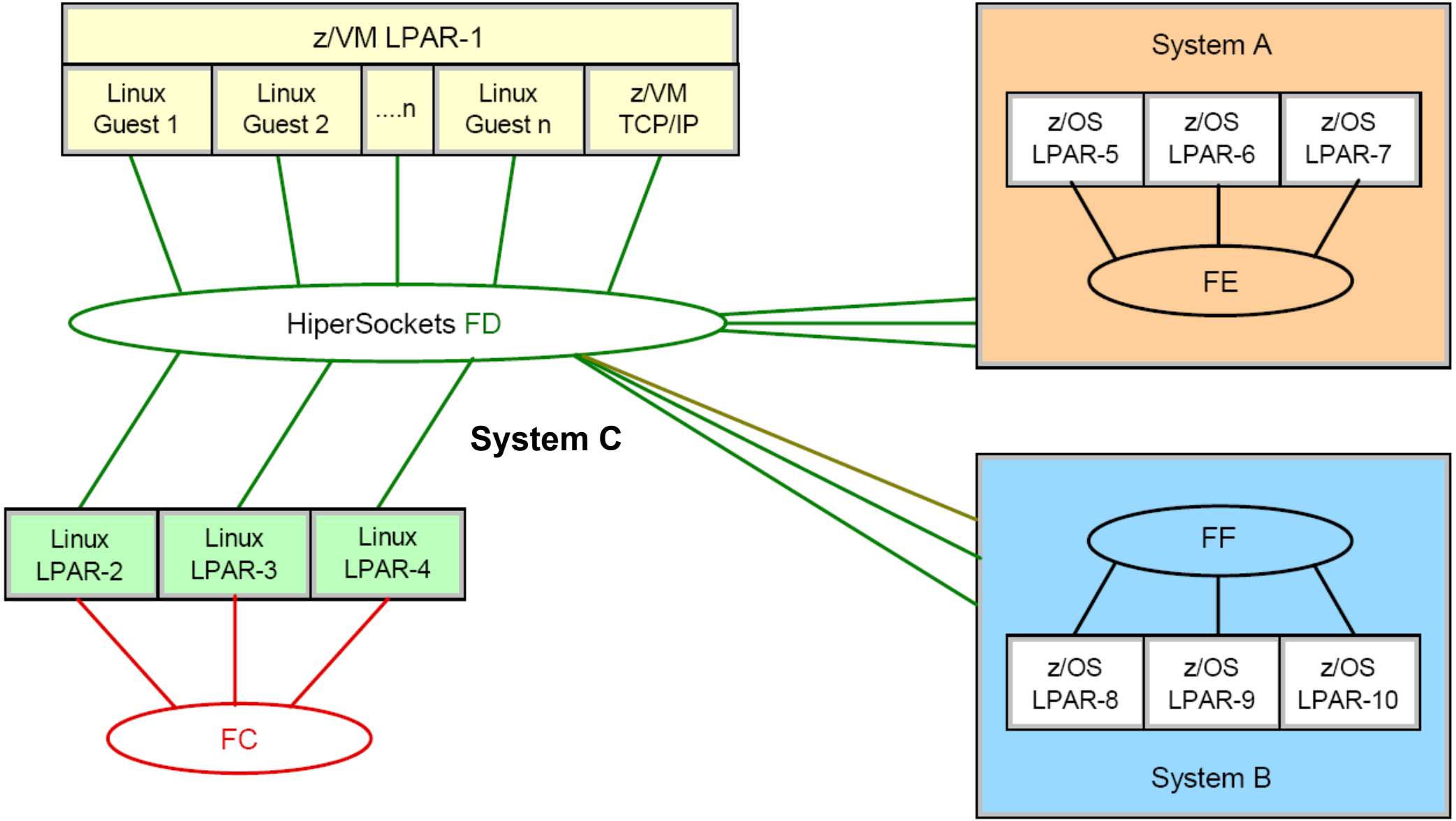
Die HiperSockets Implementierung basiert auf der OSA-Express Queued Direct I/O (QDIO)-Protokoll. Daher werden HiperSockets auch als internes QDIO oder IQDIO bezeichnet. Der Microcode des OSA Express Adapters emuliert die Link Control Schicht (Schicht 2 des OSI Stacks) einer OSA-Express QDIO Schnittstelle.

VSWITCH ist eine Firmware Funktion, die unter Nutzung von iQDIO wie ein OSI Layer 2 Switch arbeitet. Es verbindet ein externes Netzwerk über den OSA Adapter mit den virtuellen Gast Maschinen.

HiperSockets ermöglichen in einem Rechner bis zu 16 “virtual” Local Area Networks (LANs) mit deutlich reduziertem System Overhead.

HiperSockets können für die Kommunikation in einem einzigen physischen System oder zwischen Sysplex-Instanzen eingesetzt werden.

HiperSocket Verbindung mit vier Virtuellen LANs in einer Sysplex Konfiguration mit drei Systemen



Dargestellt ist eine Sysplex Konfiguration mit drei Systemen:

- System A mit LPAR-5 , LPAR-6 und LPAR-7,
- System B mit LPAR-8 , LPAR-9 und LPAR-10,
- System C mit LPAR-1 , LPAR-2, LPAR-3 und LPAR-4.

Die Verbindung erfolgt mit vier virtuellen LANs mit den Bezeichnungen FC, FD, FE und FF.

LAN FC verbindet die zLinux Systeme in LPAR-2, LPAR-3 und LPAR- 4.

LAN FE verbindet die z/OS Systeme in LPAR-5, LPAR-6 und LPAR- 7.

LAN FF verbindet die z/OS Systeme in LPAR-8, LPAR-9 und LPAR- 10.

LAN FD verbindet

- die drei zLinux Systeme in LPAR-2, LPAR-3 und LPAR- 4,
- alle Linux Server, die unter z/VM in LPAR-1 laufen
- Alle z/OS Server in Systen A (LPARs 5 bis 7)
- Alle z/OS Server in System B (LPARs 8 bis 10)
- Den z/VM TCP/IP Stack, der in in LPAR-1 läuft

Virtualisierung Teil 5

Weiterführende Information

Vielleicht interessiert Sie hier ein Video vom IBM Vertrieb

Virtualization with IBM System z
http://www.youtube.com/watch?v=4xL8s8WZUxo&NR=1

Einen Aufsatz “A Comparison of Software and Hardware Techniques for x86 Virtualization” können Sie herunterladen unter

http://www.informatik.uni-leipzig.de/cs/Literature/esiisup/part01.pdf

Falls Sie dies interessiert, enthalten die folgenden Seiten zusätzliche Informationen zu den Virtualisierungsproblemen bei der x86 Architektur.

Backup Information zu VMware

VMware ist ein Produkt mit Virtual Machine Hypervisor Funktionalität für die x86 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 x86 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, mit der 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 Leistungsverschlechterung führen.

Die Nutzung von Host-Betriebssystem-Komponenten verursacht einen Leistungsverlust, insbesondere bei Ein-/Ausgabe-Operationen. 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.

Im Vergleich zu VM/370 sind der ESX Server und VMware benachteiligt, weil einige kritische Eigenschaften in der x86-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.

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 x86 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 privilegierten Maschinenstatus Information 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*

VMware's 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; es kann sein, dass bestimmte Anwendungen nicht lauffähig sind.

Weitere Implementierungen

Ein anderes Produkt für die x86 Plattform ist **VirtualPC** von Microsoft. VirtualPC wurde ursprünglich von Connectix entwickelt.

VirtualBox ist eine von der Firma innotek (von Sun Microsystems übernommen, mittlerweile zu Oracle gehörend) entwickelte Virtualisierungslösung, die es dem Benutzer erlaubt, weitere Betriebssysteme (Gäste) unter einem laufenden System (Host) zu installieren und wie eine normale Anwendung zu nutzen. Als Hostsysteme werden Windows ab XP, Mac OS X, Linux (ab Kernel 2.4) und FreeBSD (ab 7.0), als Gastsysteme neben diesen zusätzlich noch Windows NT und 2000, OS/2, DOS-basierte Betriebssysteme, Linux (ab Kernel 2.2), L4, Solaris, NetWare sowie diverse BSD-Derivate unterstützt. In einer 2010 durchgeführten Umfrage durch LinuxJournal.com und LifeHacker.com, war VirtualBox das populärste Virtualisierungs-Produkt mit über 50% der Stimmen.

Seit dem 15. Januar 2007 steht VirtualBox in zwei Versionen zur Verfügung: Eine Open-Source-Edition (kurz: OSE), die unter der GPL v2 veröffentlicht wird, und eine PUEL-Variante (Personal Use and Evaluation License), die unter bestimmten Bedingungen kostenlos verwendet werden darf. Diese Bedingungen sind recht weit gefasst, so dass der Gebrauch am heimischen PC bedenkenlos möglich ist.

Kernel-based Virtual Machine (KVM) ist eine Virtual Machine Implementierung, die den Operating System's Kernel benutzt. Dies ermöglicht eine bessere Performance als Lösungen, welche User-Space Driver einsetzen, In den meisten Fällen verwendet KVM eine Linux Kernel Virtualization Infrastruktur. KVM unterstützt native Virtualisierung für x86 Pozessoren, welche die Intel VT-x oder AMD-V Erweiterungen unterstützen. Es wurde auch auf die S390, PowerPC, und IA-64 (Itanium) Plattformen portiert. Ein ARM Port findet derzeit statt.

KVM, unterstützt zahlreiche Gastbetriebssysteme, z.B. Linux, BSD, Solaris, Windows, Haiku, ReactOS and AROS Research Operating System. Eine modifizierte Version von Qemu ermöglicht die Benutzung von Mac OS X als Gast.

Die seit Herbst 2010 verfügbare z196 Mainframe Version mit ihrer zEnterprise Blade Extension (zBX) macht umfangreichen Gebrauch von KVM.

Paravirtualisierung

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 Computer Laboratory der University of Cambridge entwickelt wird und ebenfalls auf der x86 Plattform läuft. Um die Probleme mit der ursprünglichen x86 Architektur zu umgehen, wird in einer Version 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 zu der x86 Architektur. Dies verbessert das Leistungsverhalten, erfordert aber Änderungen des Gast-Kernels. Hiervon ist nur ein sehr kleiner Teil des Kernel-Codes betroffen. Derzeitig 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.

Die Firma SW-Soft setzt ihre **Virtuozzo Virtual Private Servers (VPS)** Software vor allem für Hosting Services bei Hosting Providern ein. Als Konkurrent tritt die Firma Ensim mit ihrer konzeptuell ähnlichen Extend Software auf. 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 über keinen 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 läuft. Die VEs sind gegeneinander abgeschottet, können unabhängig voneinander gestartet und beendet werden und besitzen je ein eigenes Dateisystem.

Ein ähnlicher Ansatz wird von ***Denali*** *verfolgt*. Als Gast-Betriebssystem dient ***Ilwaco***, eine speziell an den Denali Hypervisor angepasste BSD Version. Denali unterstützt nicht das vollständige x86 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.

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.