freeX Extra

freeX Extra

Deutschland
€ 9,80

Nr.1
Frühjahr 2008 DS0127600

Ausgabe für den Mainframe

IBM System z

Technologie für die Zukunft

IBM System z

Erfahren Sie, wie Sie Ihr Rechenzentrum effizienter gestalten können:
IBM.COM/**TAKEBACKCONTROL**/GREEN/DE

[1] Benötigt Advanced Power Virtualization, das optional und zu einem Aufpreis erhältlich ist.

Grußwort

Lieber Leser,

Sie denken, der Mainframe sei alt und längst überholt? – Falsch! Der Großrechner der IBM, das »System z«, ist gefragter denn je. Viele Kunden setzen zunehmend auf den Ausbau der bewährten Plattform.

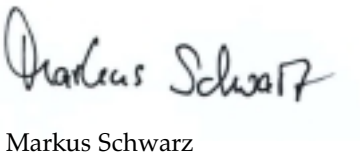
Auszeichnend für diese Plattform sind ...

- die enorme Skalierbarkeit,
- höchste Zuverlässigkeit,
- die Sicherheit für hochsensible Anwendungen,
- die ungeschlagene I/O-Performance
- und der – immer wichtiger werdende – Aspekt der hohen Energieeffizienz.

Die langjährige Innovationsgeschichte des Mainframes erlebt nun durch das System z10 einen neuen Höhepunkt. Schnellere Prozessoren und leistungsstarke Schnittstellen sind optimal auf moderne serviceorientierte Architekturen (SOA) und Information on Demand (IoD) abgestimmt.

Überzeugen Sie sich selbst!

Mit freundlichem Gruß



Markus Schwarz

Business Unit Executive System z Software
IBM Software Group Germany

Inhalt

freeX Extra 1'08 Frühjahr 2008

Die Erfolgsgeschichte

VON ISABEL ARNOLD UND ANDREAS GRÖSCHL

In vielen Unternehmen steht hinter der größten Stahltür eine recht unscheinbar anmutende mannshohe Maschine mit schwarzem Gehäuse: der Mainframe. Heutzutage werkelt auch Tux auf ihm, und Virtualisierung, die in dieser Rechnerklasse schon längst implementiert ist, ist heute auch auf PCs in Mode gekommen.

Bild 1: SAGE (Semi-Automatic Ground Environment), das erste Echtzeitsystem

Die Technik des Mainframe hat die modernen PCs mehr beeinflußt als man gemeinhin ahnt. Der IBM Mainframe vom System/360 erblickte vor 47 Jahren das Licht der Welt, das System z10 ist der jüngste Sproß der Familie. Genau genommen ist das System z der einzige »echte« Mainframe, der den Sprung in die Gegenwart geschafft hat. In »System/360« steht 360 für die 360 Grad eines Kompasses und soll ausdrücken, daß das System für jeden Job und jede Größe geeignet ist.

Fundamente

Die ersten Grundsteine für System/360 wurden bereits durch Großprojekte in den späten fünfziger Jahren gelegt. Auftraggeber für eines der ersten Projekte war – wie so oft – das Militär. Nachdem die Sowjetunion 1949 die erste Atombombe zündete, sah die Regierung der USA dringenden Bedarf nach einem Frühwarnsystem, das 1958 unter dem Namen SAGE (Semi-Automatic Ground Environment) aus der Taufe gehoben wurde. Zu Spitzenzeiten betreuten siebentausend IBM-Mitarbeiter die 250-Tonnen-Computer, von denen jeder aus etwa 60000 Vakuumröhren bestand. SAGE gilt als das erste Echtzeitsystem seiner Zeit.

Bild 2: Das IBM System/370 Model 125

Die Kosten für die Entwicklung des System/360 in den sechziger Jahren betrugen ganze fünf Milliarden US-Dollar (entspricht etwa 30 Milliarden US-Dollar heute). Es war das größte privat finanzierte Projekt, das je unternommen wurde. Die IBM mußte sogar Aktienanteile im Wert von 370 Millionen US-Dollar verkaufen, um die Löhne der Mitarbeiter auszahlen zu können. Das Fortune Magazin titelte dies später als »IBM's $5,000,000,000 Gamble«.
Nachdem das System/360 etabliert war, folgt eine lange Liste weiterer Entwicklungen und Innovationen. Die größten Highlights der folgenden 25 Jahre waren:
1966 wird IMS (Information Management System) für das Apollo-Programm entwickelt. Bei der Mondlandung von Apollo 11 im Jahr 1969 verwaltete das hierarchische Datenbanksystem

unter anderem die umfangreichen Stücklisten.
1968 erblickt der bis heute am häufigsten eingesetzte Transaktionsmonitor CICS (Customer Information and Control System) das Licht der Welt. CICS bot erstmals die Möglichkeit, Online-Transaktionen zu verarbeiten.
1970: Nachdem man sich neun Jahre zuvor wegen der noch unausgereiften Technologie gegen vollständig integrierte monolithische Schaltkreise entscheiden mußte, waren diese in der Zwischenzeit ausgereift. Im Juni wird das darauf basierende System/370 vorgestellt, das mit den neuen 128-Bit bipolaren Chips fünfmal schneller als System/360 ist. Zusätzlich arbeitete der 370 als einer der ersten Computer mit virtueller Speichertechnologie.
1972: IBM adaptierte das Prinzip der Virtuellen Maschine für den Mainframe. VM ist ein schlankes Betriebssystem, das die Systemressourcen virtualisiert und einer Anzahl von Gast-Betriebssystemen zur Verfügung stellt.
1988 wird nach IMS ein zweites, diesmal relationales Datenbanksystem mit dem passenden Namen DB2 angekündigt.
1990: System/390 wurde angekündigt und war bereits 353-mal schneller als das ursprüngliche System/360. Weitere Neuerungen waren unter anderem die Verwendung von Glasfaserkabeln und integrierte Krypto-Prozessoren.
Trotz ständiger Neuerungen schien der Mainframe zu Beginn der neunziger Jahre aus der Mode gekommen zu sein. Der Trend bewegte sich weg von der Idee eines größeren zentralen Computers mit einem Meer an Terminals hin zu dezentralen Umgebungen mit Netzwerken aus Computern aller Größen. Zwar führte IBM in ihrer Produktlinie bereits kleinere Computer wie RS/6000 (das heutige *System p*) oder AS/400 (heute *System i*) ein, dennoch klärt das nicht die Frage über die zukünftige Positionierung des Mainframes. Einige Kritiker halten das Ende des »Big Iron« für besiegelt, beispielsweise schrieb 1991 die amerikanische Zeitschrift InfoWorld: »Ich sage voraus, daß der letzte Mainframe am 15. März 1996 abgeschaltet wird.«
Sie irrten. Was hat den Dinosaurier unter den Computern vor dem vorausgesagten Aussterben bewahrt? Zum einen natürlich diejenigen, die die Plattform nicht verlassen können oder auch gar nicht verlassen wollen und besserer Zeiten harren. Und zum anderen eine längst überfällige Neuerfindung des technologischen Kerns von innen heraus. Erstmals wird die bisher ausschließlich für den PC verwendete (und zu diesem Zeitpunkt im Vergleich zur bipolaren noch recht langsame) CMOS-Prozessortechnologie im Mainframe eingesetzt. Das IBM-Labor in Böblingen verbessert und verfeinert CMOS, wodurch schließlich die Bipolar-Technologie komplett abgelöst werden kann.

Bild 3:
Das neueste IBM-Großrechnermodell System z10

e-Business made by System z

Inzwischen heißt der IBM Mainframe System z10. Er hat mit bis zu 64 Prozessoren, bis zu 1536 GByte RAM, Linux und einzigartigen Fähigkeiten im Bereich Virtualisierung und Workload-Balancing mit dem System/360 von 1964 augenscheinlich nicht mehr viel gemein. Doch die Architektur ist noch immer dieselbe, weswegen Lochkarten-Jobs von vor 45 Jahren auch auf dem System z10 lauffähig wären – vorausgesetzt, man bekommt irgendwo einen Lochkartenleser her. Moderne Anwendungen nutzen neben CICS und IMS natürlich die Möglichkeiten von J2EE und Linux, was sowohl die hochtrabenden Visionen der Architekten in Sachen service-orientierte Architekturen fördert, als auch die Träume der Administratoren, ganze Serverparks per Knopfdruck verwalten zu können und statt Hunderter wenig ausgelasteter Maschinen Hunderte virtuelle Systeme in einer einzigen schwarzen Box optimal ausnutzen zu können, beflügelt. Der Mainframe hat seinen Platz in der neuen Welt des e-Business und der Hochleistungs-PC-Server gefunden und ist doch in zwei Aspekten der alte geblieben: Die Weltwirtschaft tickt weiterhin auf ihm als dem Rückgrat großer Unternehmen und er bildet weiterhin die Spitze des Premium-Segments auf dem Servermarkt. ◆

Literatur:

http://www.ibm.com/software/os/zseries/pdf/360Revolution_0406.pdf
http://www.ibm.com/servers/de/eserver/zseries/time-line/
John Young: Exploring IBM's New Age Mainframes, Maximum Press, ISBN 1-885068-05-0

Mainframe-Leistung auf Leiterbahnen

VON MARTINA SCHMIDT

Nach einer langen Entwicklungszeit wurde der Prozessor des Mainframe einem Redesign unterworfen. Ganz gegen die Tradition wurde nun auch Wert auf rechenintensive Operationen gelegt – der Mainframe hat sich damit vom I/O-Künstler zum Allround-Server entwickelt.

Neunhunderteinundneunzig Millionen Transistoren, sechs Kilometer Kabel auf einer Fläche von 21,7 mal 20,0 Millimetern – das ist der neue Mainframe-Mikroprozessor z10.

Der Nachfolger des System z9 EC-Chips verfügt über vier Kerne, wobei jeder der vier Kerne einen Instruktionscache mit 64 KByte besitzt, der Daten-Level-1-Cache ist 128 KByte groß und der Level-2-Cache hat 3 MByte zur Verfügung, außerdem sind Beschleuniger für Verschlüsselung, Datenkompression und dezimale Gleitkommaarithmetik verbaut.

Design

Der Chip unterstützt nun Taktraten von über 4 GHz, womit er doppelt so schnell wie das Vorgängermodell ist. Auch die Leistung der Systemschnittstellen hat sich deutlich verbessert: Jeweils zwei symmetrische Multiprocessing Hub-Ports (SMP) mit einem Durchsatz von 48 GByte pro Sekunde, vier Memoryports mit 13 GByte pro Sekunde und zwei I/O-Ports mit einem Durchlaß von 17 GByte in der Sekunde sorgen für die enorme Geschwindigkeit des Datentransfers. Insgesamt weist der Chip 8765 Pins auf.

Der Prozessor wurde gemeinsam mit dem POWER6-Chip entwickelt, weshalb einige Designmerkmale wie beispielsweise die 65-Nanometer-Fertigungstechnologie »Silicon-on-Insulator« (SOI), das logische Design, die Ausführungseinheit, die Gleitkommaeinheiten und die Bustechnologie sich entsprechen. Auch das Design der Pipeline, die auf hohe Frequenz, geringe Latenz und weitestgehend in-order ausgelegt ist, ist identisch. In anderen Aspekten unterscheiden sich die beiden Prozessoren stark voneinander. Zu nennen sind diesbezüglich zum Beispiel die Cachehierarchie und SMP-Topologie, sowie die Protokolle und die Chip-Organisation. Die verschiedenen Befehlssätze resultieren in sehr verschiedenen Kernen.

Der z-Prozessor besitzt einen umfangreichen CISC-Befehlssatz und ist dabei immer noch rückwärtskompatibel bis hin zur Architektur des System/360 der IBM aus den sechziger Jahren. Der Instruktionssatz besteht aus achthundertvierundneunzig z-Instruktionen, von denen dreiviertel komplett in Hardware implementiert sind.

Der Chip unterstützt 24-, 31- sowie 64-Bit-Adressierung, multiple Adreßräume für eine hohe Interprozeßsicherheit und verschiedene Zahlenformate. Außerdem sind seine Virtualisierungsfähigkeiten legendär.

Die Erweiterungen des Chips gegenüber seinem Vorgänger sind:

- Über fünfzig neue Instruktionen, um die Effizienz kompilierten Codes zu verbessern.
- Optimierter Soft- und Hardwarecache.
- Unterstützung von 1 MByte großen Pageframes.
- Volle Hardwareunterstützung für dezimale Gleitkomma-Arithmetik.

In der Industrie führend sind seine Fehlererkennung und -behandlung, sein Design ist auf eine feingranulare Redundanz und ausgefeilte Kontrollmechanismen ausgelegt.

Vom I/O zum Allroundserver

Die Pufferung des Prozessorstatus erlaubt eine präzise Core-Wiederholung für beinahe alle Hardwarefehler und bei einem eventuellen Fehler im Kern einen dynamischen, transparenten Übergang auf einen anderen Kern. Ein Fehlerbehebungscode (ECC) auf den Level-2- und -3-Caches sowie eine genaue Prüfung der Paritäten in insgesamt über zwanzigtausend Kontrollinstanzen auf dem

Chip garantieren, daß der z10 nie Daten verliert und auch nie ausfällt. Obwohl der Prozessor über Möglichkeiten für symmetrisches Multi-Processing verfügt, gibt es einen dedizierten Begleitchip, den sogenannten SMP Hub Chip. Er verbindet verschiedene Prozessorchips (z-Prozessoren sowie weitere Hub-Chips) mit einer Bandbreite von 48 GByte pro Sekunde pro Prozessor, außerdem bietet er einen 24 MByte großen Level-3-Cache. Der Hub Chip besteht aus 1,6 Milliarden Transistoren, 242 MByte SRAM, außerdem noch einmal drei Kilometern Kabel, und er ist dimensioniert auf 20,8 mal 21,4 Millimeter.

Warum sind diese Details so bemerkenswert?

Zum einen deshalb, weil mit dem neuen Prozessor die nächste Generation Mainframes von IBM auf einem Gebiet punktet, auf dem sie bis jetzt noch keine Spitzenposition innehatte. Gemeint sind recheninstensive Workloads, die bisher eher als untypisch für diese Klasse von Computern galten. Der neue Chip bietet die Möglichkeit, den Mainframe als einen echten Allround-Server einzusetzen, ohne die typischen Vorzüge eines Mainframes wie Ausfallsicherheit und Virtualisierung aufzugeben.

Zum anderen, weil »pro Blech« nun noch mehr Last verarbeitet werden kann und das bei nur moderat höheren Stromkosten. Diese Tatsache paßt zu der aktuellen Diskussion um »Green IT« sehr viel besser als viele andere Ansätze, auch wenn man dem Mainframe in diesem Bereich dies in der Vergangenheit vielleicht nicht zugetraut hätte. ◆

Die beigelegte CD-ROM

Wie die gesamte freeX Extra ist auch die beigelegte CD nur einem Thema gewidmet: dem Mainframe.

Zwar werden etliche System z betreffende Themengebiete in den gedruckten Beiträgen in diesem Heft bereits angesprochen, aber wir möchten auch den Lesern, die an weiterführenden Informationen interessiert sind und/oder bereits mit der Materie vertraut sind, tiefergehende Informationen bieten, als in dieser Zeitschrift behandelt werden können. Wir haben deshalb eine CD-ROM mit vielen weiteren spannenden Dokumentationen beigelegt. Sie enthält 78 technische Dokumente, die weitere Informationen, konkrete Hilfestellung, Howtos sowie Tips und Tricks bieten.

Im Ordner »Linux on z« befinden sich nicht nur Dokumente zu den Linux-Distributionen auf System z, sondern auch Tips und Hilfestellung zu z/VM, dem Virtua-lisierungs-System auf System z.

Das Betriebssystem z/OS – das auch im Heft beschrieben wird – dürfte dem Unix-Experten nicht unbedingt geläufig sein. Deshalb wird in einer Serie von Redbooks das »ABC der z/OS-Systemprogrammierung« behandelt. Auch Dokumente über die Spezialitäten sowohl von J2SE als auch J2EE auf z/OS können hier gefunden werden, da nicht nur der WebSphere Application Server, sondern auch beispielsweise CICS einen EJB-Container besitzt.

Darüber hinaus hat die WebSphere Brand einen eigenen Ordner auf der CD spendiert bekommen, da das Zusammenspiel der klassischen Mainframe-Komponenten wie DB2, IMS, CICS und dem Betriebssystem z/OS mit der J2EE-Welt eine gewisse Komplexität nach sich zieht, die es mit einer ganzen Reihe technischer Dokumente zu beschreiben gilt.

Software ist auf der CD nicht zu finden. Das hat zwei Gründe:

Zum einen ist jede freie Software für den Mainframe auch von der IBM-Website zu bekommen, was der Aktualität wegen allgemein zu empfehlen ist.

Zum anderen ist das Software-Management auf z/OS sehr individuell und kaum ein z/OS-Systemprogrammierer möchte ein Tool auf dem Kernsystem einer Unternehmens-IT installieren, das er von einer Heft-CD bekommen hat.

Der Kosten-Nutzen-Faktor

von Christine Urban

Hört man »Mainframe«, denkt man automatisch an hohe Kosten. Dieses Urteil gehört längst der Vergangenheit an, denn bei einer realistischen Berechnung der Gesamtaufwendungen zeigt sich oft sogar das Gegenteil!

Die Kosten für Hard- und Software des Mainframes stehen im Ruf, sehr hoch zu sein. Wie sich im Lauf dieses Beitrags herausstellen wird, gehört dieses Urteil längst der Vergangenheit an. Eine realistische Kostenberechnung der IT-Infrastruktur darf nämlich nicht nach dem oft gebräuchlichen TCA-Modell (Total Cost of Acquisistion) erfolgen, vielmehr ist in diesem Bereich das TCO-Modell (Total Cost of Ownership) aussagekräftiger. Wendet man es an, wird deutlich, daß der Mainframe immer attraktiver wird, je ganzheitlicher die Kosten betrachtet werden. Ein Mainframe muß nämlich mit allen anfallenden Kosten analysiert werden, anstatt daß auf der Ebene einzelner Projekte nur die Anschaffungs- und Wartungskosten für Soft- und Hardware kalkuliert werden.

Jeder kennt sicher folgendes Szenario: Man steht im Supermarkt vor einem Regal mit einer Vielzahl ähnlicher Produkte. Als Käufer achtet man oft auf den Preis und entscheidet häufig zugunsten des günstigsten Produkts. Bei solchen kleinen Investitionen wird nicht lange überlegt, sondern kurzerhand entschieden, ein langfristiges Denken braucht hier nicht einsetzen.

Werden die Investitionen größer und steht beispielsweise die Anschaffung eines neuen Notebooks an, nimmt der Entscheidungsaufwand zu. Es werden verschiedene Hersteller, Lieferanten und Preise verglichen, um das optimale Angebot für den eigenen Geldbeutel zu finden. Erst bei sehr großen und bedeutsamen Kaufentscheidungen, wie zum Beispiel beim Erwerb eines Autos, werden auch alle mit der Anschaffung mittelbar und unmittelbar zusammenhängenden Kosten in die Kalkulation mit einbezogen. Neben den einmaligen Kosten für die Anschaffung und Sonderausstattung sind dies auch die Folgekosten für Versicherung und Treibstoff, sogar die Ersatzteile verschiedener Fahrzeugtypen werden oft miteinander verglichen.

TCA und TCO

Der betriebene Aufwand hängt im also privaten Bereich im wesentlichen von der Höhe einer Investition ab. Unternehmen entscheiden ähnlich über ihre Investitionen. Bei geringen Investitionen ist die Vorgehensweise überwiegend vom operativen Geschäft geprägt. Meist wird nur bei strategischen und mittel- bis langfristigen Investitionen im Detail analysiert und bewertet. Bei einer nachlässigen und unvollständigen Berechnung kann es jedoch passieren, daß eine anfangs geringe Investition im nachhinein deutlich höhere Kosten verursacht als ursprünglich angenommen.

Abhilfe kann hier nur die Berechnung wirklich aller anfallenden Kosten schaffen. Diese Art der Kostenrechnung ist in Unternehmen besser bekannt unter der von Gartner Group entwickelten Betrachtung der »Total Cost of Ownership« (kurz: TCO). Ein Beispiel verdeutlicht die Wichtigkeit dieser Kostenbetrachtung: Ein Unternehmen möchte im Zuge des Unternehmenswachstums die bestehende IT-Infrastruktur vergrößern. Dazu bestellt der IT-Leiter einen kostengünstigen Server und fügt ihn in das bestehende Rechnernetzwerk ein. Der Server kostet deutlich weniger als ein Mainframe, scheinbar wurde also alles richtig gemacht. Nun

wächst das Unternehmen weiter und in regelmäßigen Abständen bestellt der verantwortliche IT-Leiter mehr Server und integriert sie in die bestehende, stark wachsende Serverlandschaft. Dies führt schließlich dazu, daß die Zahl der bisherigen Administratoren nicht mehr ausreicht und mehr Personal eingestellt werden muß.

Berechenbare Skalierungskosten

Zudem kommen immer mehr unternehmenskritische Daten zusammen, wodurch die Ansprüche an die Sicherheit wachsen und zusätzliche Investitionen in Sicherheitsmaßnahmen notwendig sind. Die Server nehmen mehr Platz weg und die Stromrechnung steigt kontinuierlich, bis das Rechenzentrum schließlich aus allen Nähten platzt. Diese Geschichte könnte noch endlos fortgeführt werden. Die Kosten des ersten zusätzlichen Servers sind lange nicht mehr aktuell und die Folgekosten haben die anfangs niedrigen Anschaffungskosten zunichte gemacht. Es wird deutlich, daß es sinnvoll ist, Folgekosten wie auch Alternativen – selbst wenn sie anfangs als viel zu teuer scheinen – genau zu betrachten und diverse Szenarien durchzuspielen.
Vergleicht man im Rahmen einer TCO-Betrachtung die Kosten der vorher beschriebenen Serverlandschaft mit denen eines Mainframes, so wird man – teils überraschend – herausfinden, daß der anfangs augenscheinlich teuere Mainframe leicht gegen kostengünstigere Server gewinnen kann.
Unterschiedliche Studien zeigen auf, daß der Mainframe zwischen fünf bis sechzig Prozent teurer als Linux-, Unix- oder Windows-Servern ist. Wird der Mainframe dagegen mit dreißig SUN-Servern oder dreihundert PC-Servern unter Linux verglichen – was ein weitaus realistischeres Vergleichsszenario ist –, erzielt der Großrechner dreißig bis sechzig Prozent geringere Kosten.
Analysiert man diese Ergebnisse genauer, werden die Unterschiede zu

anderen Plattformen noch deutlicher sichtbar.
Bisher konnten verteilte Server ein großes Plus bei den Softwarelizenzkosten für sich verbuchen, da diese auf dem Mainframe deutlich höher waren. Durch neue Spezialprozessoren für begrenzte Anwendungsgebiete (unter anderem IFLs (Integrated Facility for Linux) für alle Anwendungen unter Linux) wurden die Softwarekosten neu strukturiert und sind nun mit den Softwarekosten für verteilte Server durchaus wettbewerbsfähig.
Im Hinblick auf die Hardwarekosten haftet dem Mainframe noch das Vorurteil an, sehr teuer zu sein. Aber auch hier hat sich einiges getan und die Kosten für Hardware sind gesunken, so daß es sich auch hier unbedingt lohnt, einen Mainframe mit verteilten Servern zu vergleichen.
Außerdem lassen sich die Personalkosten durch den Einsatz eines Mainframes deutlich senken. Je größer der Workload auf einem Mainframe, desto größer die Differenz. Um alle Server in einer großen heterogenen Serverlandschaft zu verwalten und zu pflegen, bedarf es deutlich mehr Mitarbeiter als für einen Mainframe, der alle Applikationen konsolidiert beherbergt.
Durch spezielle Kühltechnologien verbraucht ein Mainframe heutzutage nur etwa ein Fünftel des Stroms, den ein vergleichbarer Workload auf einem verteilten Server benötigt. Das bedeutet einerseits, daß der Mainframe deutlich kostengünstiger ist. Zum anderen ist dies auch ein wichtiger Schritt in Richtung Green IT, die heutzutage politisch zunehmend an Bedeutung gewinnt.

Spitzenwerte bei Servern

Sinnvoll ist eine TCO-Betrachtung aber nicht nur, um sich schon vor einer Investition mit daraus resultierenden Folgekosten vertraut zu machen, sondern auch um Veränderungen der unterschiedlichen Kostenblöcke und Kostenanteile zu erkennen. Obwohl die Kosten für Hardware in den letzten Jahren gesunken sind, nehmen die Gesamtkosten von IT-Lösungen zu. Die Ursache liegt in einer Verschiebung der Kostenblöcke. Während vor rund zehn Jahren die Hard- und Softwarekosten zusammen achtzig Prozent der Gesamtkosten darstellten, tragen diese Faktoren heute nur noch zu einem Anteil von vierzig bis maximal sechzig Prozent an den Gesamtkosten bei. Deutlich zugenommen hat dafür der Anteil der Personalkosten. Sie betragen fast fünfzig Prozent der Gesamtkosten. Diesen Veränderungen sollte man sich vor Investitionen bewußt sein, da hier deutlich wird, daß eine reine Betrachtung der Anschaffungskosten längst nicht mehr ausreicht.
Eine Betrachtung der Total Cost of Ownership ist zwar anfangs eine Herausforderung, da es unter Umständen aufwendig ist, für verschiedene Alternativen geeignete vergleichbare Kosten zu finden. Im nachhinein wird man aber merken, daß sich die Anstrengungen gelohnt haben. Beim privaten Autokauf scheut man ja meist auch keine Mühe, um das Beste für sich zu finden. ◆

Der Pinguin und das große Blech

VON MICHAEL STÖRCHLE

Zu Beginn dieses Jahrtausends ergab sich in der IT-Szene eine zunächst seltsam anmutende Liaison: Der Mainframe und Linux fanden zueinander. Dinosaurier und Pinguin, eine auf den ersten Blick eigenartige Mischung. Wir werfen einen zweiten Blick darauf.

Linux auf dem Mainframe, offiziell »Linux on IBM System z« genannt, ist die Verbindung der Stärken von Linux und des Mainframes. Es ist kein Versuch, eines von beiden neu zu erfinden. Für Linux – ebenso wie für andere Mainframe-Betriebssysteme – sind die systemimmanenten Vorteile des Mainframes wie Stabilität, Zuverlässigkeit und Skalierbarkeit ein direkter Mehrwert. Die besondere Attraktivität von Linux auf dem IBM System z zeigt sich vor allem in Verbindung mit den exzellenten Virtualisierungsmechanismen des Mainframes. Hier wird die symbiotische Beziehung zwischen Dinosaurier und Pinguin besonders klar: Hunderte Linux-Instanzen, die in virtuellen Maschinen auf einem real vorhandenen Mainframe betrieben werden. Sozusagen eine Pinguinfarm. Oder, um bei dem Begriff Dinosaurier zu bleiben, eine agile Raptorenherde.

Von Dinosauriern und Pinguinen

Linux on IBM System z gliedert sich bezüglich der Installationsmöglichkeiten nahtlos in die Reihe der anderen Mainframe-Betriebssysteme ein. Es läßt sich in einer LPAR (Logical Partition) oder unter z/VM betreiben. Die Anzahl der Linux-Instanzen ist prinzipiell unbegrenzt, wird in der Realität jedoch durch eine begrenzte Zahl der zur Verfügung stehenden LPARS beziehungsweise durch die unter z/VM zur Verfügung stehenden Ressourcen limitiert. Generell ist der Einsatz unter z/VM empfohlen, da hier eine effizientere Virtualisierung und damit eine höhere Systemauslastung realisiert wird: Ressourcen können granularer als auf der LPAR-Ebene zwischen den einzelnen Linux-Instanzen verteilt und geteilt werden, ebenso sind die Linux-Instanzen dank zahlreicher Tools des Hypervisors leichter administrierbar. Die Hardware des Mainframes wird auch unter Linux on IBM System z im vollen Umfang genutzt. OSA-Adapter, Hipersockets, Crypto-Karte oder die IEEE-Floating-Point-Instruktion sind nur einige Beispiele hierfür. Die Architektur des IFL (Integrated Facility for Linux), der oft irreführend als »Spezialprozessor für Linux« bezeichnet wird, ist identisch mit normalen Mainframeprozessoren und nicht speziell für Linux gebaut. Vielmehr handelt es sich um eine kostengünstige Alternative zu normalen Mainframe-Prozessoren, da – bedingt durch eine Microcode-Restriktion – nur Linux-Workload und gegebenenfalls das zugehörige z/VM abgehandelt werden kann. Dieser Effekt ist lediglich auf der Kostenseite sichtbar, weshalb man hier auch von einer »kaufmännischen Lösung« spricht.

Linux ist Linux ist Linux – dieser Grundsatz, nämlich eine möglichst identische Betriebssystemumgebung für viele Hardware-Plattformen, trifft auch auf die Mainframe-Variante zu. Linux on IBM System z ist eine vollständige Portierung von Linux auf

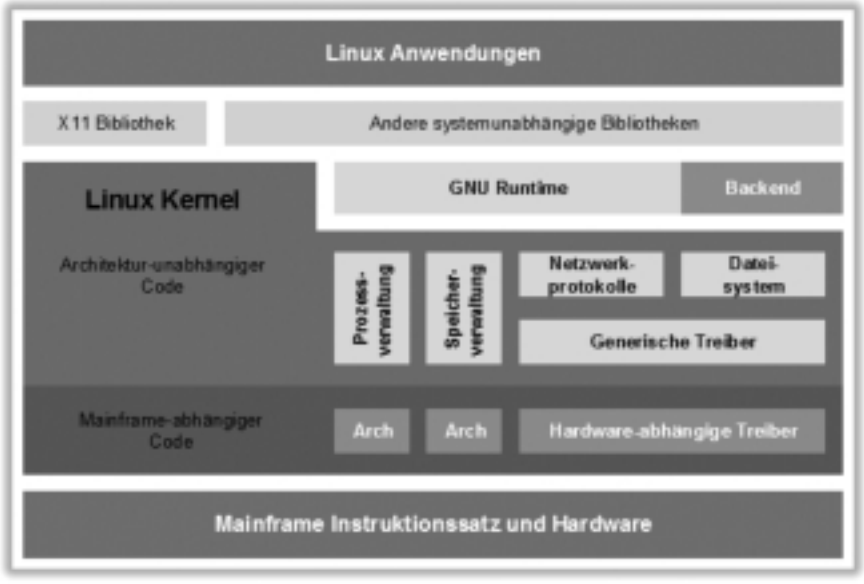


Bild 1: Aufbau des Linuxkernels (Mainframe)

die Mainframe-Hardware ohne jegliche API-Emulation. Daher basiert die komplette Betriebssystemumgebung auch auf ASCII, die sonst auf dem Mainframe übliche EBCDIC-Codepage sucht man vergebens. Ebenso gibt es keine ASCII/EBCDIC-Konversionen wie sie unter dem z/OS-Service USS (Unix System Services), einer kompletten Unix-Umgebung unter z/OS, notwendig ist.

Bei der Portierung des Linuxkernel-Codes auf die Mainframe-Architektur mußte naturgemäß der Quelltext angepaßt werden. Die Unterschiede im Sourcecode von Linux on IBM System z zur IA32-Version bestehen jedoch lediglich auf der untersten Ebene der Linux-Kernelarchitektur, die direkt mit der Hardware kommuniziert. Diese Besonderheiten der Architektur werden zum Zeitpunkt der Kernel-Kompilierung eingebaut. IBM hat hierzu mainframe-eigene Kernel-»Ergänzungen« entwickelt, auf die der verwendete C-Compiler aufgrund des Arch-Flags verzweigt. Beispiele hierfür sind bestimmte Hardwaretreiber und spezielle Instruktionen. Änderungen im allgemeinen Linux-Sourcecode oder der Kernelstruktur gibt es nicht. Daher ist es auch keine Überraschung, daß diese systemspezifischen »Lines of Code« nur in etwa 4,1 Prozent des gesamten Quelltexts ausmachen. Bei den wichtigen Systemkomponenten wie gcc, glibc oder binutils sind es unter einem Prozent.

Zwangsläufig stellt sich die Frage, was die Mainframe-Variante grundlegend von der IA32-Version auf der untersten Ebene der Linux-Kernelarchitektur unterscheidet. Für Anwendungsentwickler sind diese Spezifika in der Regel von geringer Bedeutung und müssen lediglich bei hardwarenahen Anwendungen berücksichtigt werden. Aus diesem Grund ist eine echte Cross-Plattform-Entwicklung unter Linux möglich.

Linux on IBM System z gibt es als 64-Bit und als 31-Bit-Versionen. Letztere Variante steht einer 32-Bit-Version auf der IA32-Architektur gegenüber. Diesen Unterschied in der Adressierbarkeit des Speichers bedingt die Historie des Mainframe. Adressen wurden schon immer in 32-Bit-Wörtern hinterlegt, allerdings wurden bis zum Jahr 1983 lediglich 24 Bit genutzt. In diesem Jahr wurde die 31-Bit-Adressierung eingeführt. Das »verlorene« Bit, auch »most significant bit« genannt, wird als Flag dafür benutzt, ob die folgenden 24 oder 31 Bit als Adresse interpretiert werden sollen. Aufgrund des »verlorenen« Bits ergibt sich zwangsläufig ein geringerer Speicherbereich, den Linux on IBM System z (zwei Gigabyte) im Vergleich zu seinem IA32-Pendant (vier Gigabyte ohne PAE) adressieren kann. Eine weitere Folge ist, daß die Speicheradressen, in denen Executables und Libraries geladen werden, auf den beiden Architekturen unterschiedlich sind. Auch hier gilt der Grundsatz, daß diese Umstände Anwendungsentwickler in der Regel nicht berühren. Sie können aber bei der Verwendung von Pointern bedeutsam werden, ebenso falls mmap() in Verbindung mit dem Parameter MAP_FIXED genutzt wird. Momentan löst die 64-Bit-Version immer mehr 31-Bit-Installationen ab, weshalb diese Betrachtungen in der Praxis immer weiter an Bedeutung verlieren.

Anpassungen am Kernel

Ein weiterer grundlegender Unterschied besteht in der Art und Weise, wie Zahlen, die aus mehreren Byte bestehen, im Speicher abgelegt werden, der sogenannten Byte-Order. Während auf der IA32-Architektur Little Endian (das niederwertigste Byte wird an der niedrigsten Speicheradresse abgelegt) als Byte-Order verwendet wird, ist beim Mainframe Big Endian die Byte-Order. Dieser Unterschied ist für Anwendungsentwickler allerdings nur bei Pointermanipulationen relevant.

Architekturbedingte Unterschiede zwischen verschiedenen Plattformen sind oft die Knackpunkte bei Anwendungsmigrationen von der Entwicklungsplattform auf andere Hardware-Architekturen. Genau hier ist eine der großen Stärken von Linux, denn es gestattet eine flexible Cross-Plattform-Entwicklung von Anwendungen: Die plattformspezifischen Anpassungen im Linux-Kernel sind, wie bereits aufgezeigt, marginal und für Anwendungsentwickler in der Regel irrelevant. Daher gibt es einen Codestream für mehrere Plattformen.

Der Einsatz von Linux auf einem Mainframe macht vor allem dann Sinn, wenn Anwendungen die klassischen Stärken des Mainframes benötigen: Durch Virtualisierungstechniken stehen Mechanismen zur Verfügung, die eine maximale Nutzung der vorhandenen Hardwareressourcen im Sinne einer Serverkonsolidierung gestatten. Cloning- und Verwaltungstechniken gestatten darüber hinaus einen sehr kurzfristigen Rollout bei minimalem administrativen Aufwand. Im Bereich Sicherheit ist der Mainframe unangefochten am Markt. Durch Funktionen wie Cryptohardware oder Resource Access Control Facility wird einem Linux-System auf dieser Architektur eine breite Palette an Sicherheitslösungen und -zertifizierungen angeboten, was einen Einsatz in sensiblen, unternehmenskritischen Bereichen legitimiert. Verfügbarkeit, Verläßlichkeit und Ausfallsicherheit des Systems sind zwei weitere wichtige Punkte – auch in diesem Bereich ist die Mainframe-Hardware herausragender Marktführer. Für den Produktivbetrieb ist darüber hinaus die Performance entscheidend. Die Mainframe-Hardware punktet in diesem Bereich vor allem dann, wenn es nicht nur auf reine Prozessorleistung, sondern auf ein ausgewogenes Gesamtsystem ankommt. Denn die Leistungsfähigkeit eines Systems im Produktivbetrieb ist nicht über die Taktung des Prozessors, sondern über die Antwortzeit der Anwendung definiert. ◆

Literatur

Linux on IBM System z:
http://www-03.ibm.com/systems/z/os/linux/

Linux for Big Iron:
http://linuxvm.org/

Mainframe in a Nutshell

von Martina Schmidt, Thomas Schulze und Michael Störchle

Den Mainframe, die Mutter aller Serversysteme, kann man nicht in einem Satz beschreiben oder erklären. Das grundlegende Architekturkonzept, innovative Ideen und modernste Technik in einem System, das so groß wie ein durchschnittlicher Kleiderschrank ist, sind eine Welt für sich.

In der Informatik findet man viele Ansätze, um Rechner beziehungsweise deren Architektur in Familien einzuteilen. Eine der wohl bekanntesten ist die Flynsche Notation. Darin findet man die meisten heutigen Serversysteme, unabhängig von Hersteller oder Betriebssystemfamilie, im Bereich der Multiprozessorsysteme. Schnell kann man zu der Meinung kommen, daß alle Server im Prinzip gleichartig aufgebaut sind und sich nur in der Anzahl an Prozessoren, dem Prozessorhersteller und der Farbe des Gehäuses unterscheiden.

Ein genauerer Blick auf die unterliegende Architektur zeigt jedoch erhebliche Unterschiede zwischen heutigen Serversystemen auf. So wurden Mainframes klassisch primär auf I/O-Durchsatz sowie Sicherheit und Verfügbarkeit hin entwickelt, wodurch sich eine Clusterverfügbarkeit von 99,999 Prozent erklären läßt. Der Mainframe wurde in seiner fünfundvierzigjährigen Geschichte auf den gleichzeitigen Betrieb mehrerer Betriebssysteminstanzen optimiert, um die teuren Systeme optimal auszunutzen. Das ist auch heute noch der Grund, warum Virtualisierung und Konsolidierung zwei Stärken des Mainframe sind.

Shared Everything

Die Entwickler entschieden sich für ein Shared-Everything-System, in dem die generischen Ressourcen wie Prozessor, Speicher und die Eingabe-/Ausgabegeräte zwischen den einzelnen Betriebssystemen geteilt werden können. Um Deadlocks bei exklusiven Hardwarezugriffen zu vermeiden, mußte eine Managementschicht eingezogen werden, deren Algorithmen bis heute kontinuierlich weiter optimiert wurden, und in den achtziger Jahren in den Microcode (die Firmware, oder auch Licence Internal Code, LIC) der Systeme eingebettet wurde.

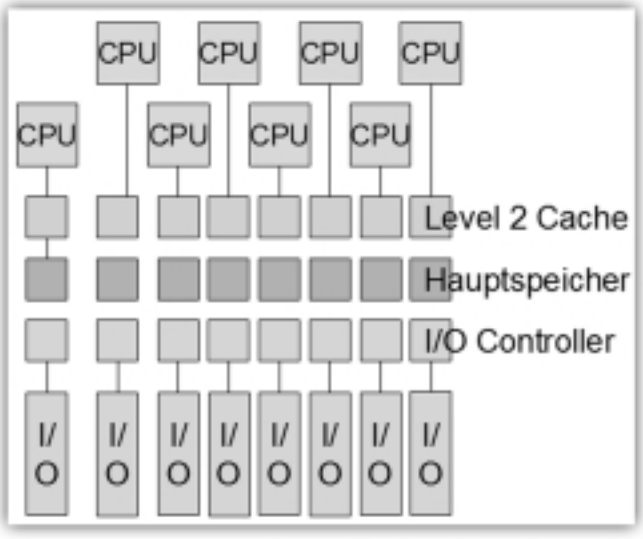


Bild 1: Architektur von Shared Everything

Die Shared-Everything-Architektur zeichnet sich dabei durch folgende wesentliche Merkmale aus:

- Hohe Auslastung: Da immer mehrere Applikationen gemeinsam um die vorhandenen Ressourcen konkurrieren, sind diese sehr hoch ausgelastet, ohne viel an Performance einzubüßen, da der Designansatz von Beginn an auf Multi-User ausgerichtet war.
- Mixed- (dynamisches) Workload geeignet:

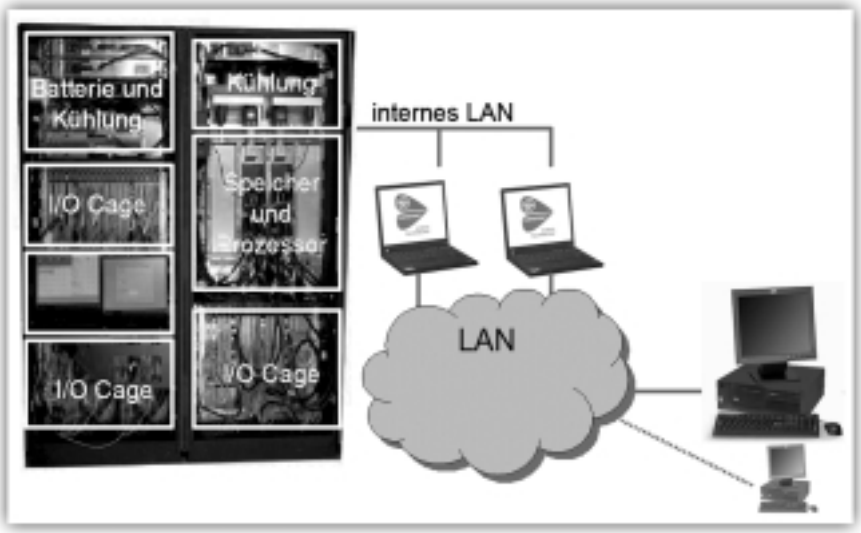


Bild 2: Aufbau des System z

Durch die vorhandenen Protokolle zur Ressourcenverteilung und Synchronisation können mixed Workloads auf den Systemen mit nur minimalen Performanceeinbußen ablaufen.

- Konsolidierung: Shared-Everything-Systeme eigenen sich sehr gut zur Konsolidierung von kleineren, wenig genutzten Systemen. Solche Systeme nutzen nur wenige Ressourcen, die bei einem Idlen des konsolidierten Systems ausgelagert werden können, so daß sie quasi keine Last erzeugen und erst bei Bedarf wieder in den Speicher geladen werden.

Komponenten heutiger Mainframes

Die Bauteile eines Mainframes kann man in mehrere Gruppen einteilen. Zum einen gibt es die Infrastruktur wie Kühlung, Batterien (optional), zum anderen Komponenten wie die Netzwerkanschlüsse und die eigentlichen Rechnerkomponenten.
Während in früheren Zeiten ein Mainframe ganze Räume füllte, benötigen die aktuellen Systeme lediglich einen Stellplatz von zirka zweieinhalb Quadratmetern. Auf dieser Fläche befinden sich dann zwei miteinander verbundene Schränke. Das Topmodell bringt es dabei mit 54 Prozessoren, 512 GByte Speicher und 1024 Netzwerkanschlüssen auf ein Gesamtgewicht von zwei Tonnen. Der Betrieb erfordert einen Rechenzentrumsboden sowie eine entsprechend ausgelegte Kühlung. Das System selbst verfügt über eine duale Kühlung, Ventilatoren und eine Flüssigkeitskühlung, die die Prozessoren selbst über einen geschlossenen Kreislauf im Betrieb herunterkühlt.
Zuerst benötigt man zur Steuerung des Mainframes und seiner hardwarenahen Funktionen Support-Elemente (SE), die sich direkt im Gehäuse des System z befinden, aus Gründen der Ausfallsicherheit zwei Stück. Sie sind durch ein privates IEEE-802.3-Netzwerk angeschlossen, können jedoch über paßwortgesichertes Remote Login oder Virtual Private Network von Arbeitsplatzrechnern aus aufgerufen werden. Bei den Support-Elementen handelt es sich um paßwortgesicherte Laptops, die jeweils mit einem eigenständigen, geschlossenen Betriebssystem versehen sind.

Infrastruktur

Zusätzlich zu den Support-Elementen werden weitere Konsolen angeschlossen. Die Hardware Management Console (HMC) bietet zusätzlich zur Steuerung des System z mittels Login auf den Support-Elementen die Möglichkeit, Betriebssystemkonsolen oder zusätzliche Anwendungen laufen zu lassen, sowie Backups der Konfigurationsdateien anzulegen.
Hier angeschlossen befindet sich auch die Remote Support Facility, ein Modem, über das Maschinendaten direkt an die IBM transferiert werden können. Hierbei handelt es sich ausschließlich um Daten, die die Hardwarekonfiguration und den Status des Systems betreffen. Sollten Fehler im System selbst auftreten, unabhängig davon, ob es sie durch Aktivierung von Ersatzkomponenten beheben kann, wird über die Telefonleitung der Technische Außendienst der IBM informiert, der dann entsprechend reagieren kann.
Gravierende Fehler können so in der Regel frühzeitig erkannt und behoben werden, bevor sie die Produktion beeinflussen. Da hauptsächlich unternehmenskritische Anwendun-

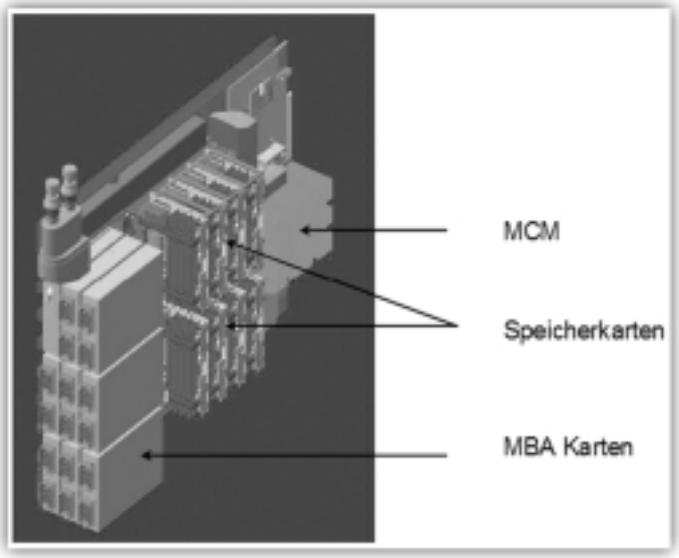


Bild 3: Aufbau eines System-z-Buchs

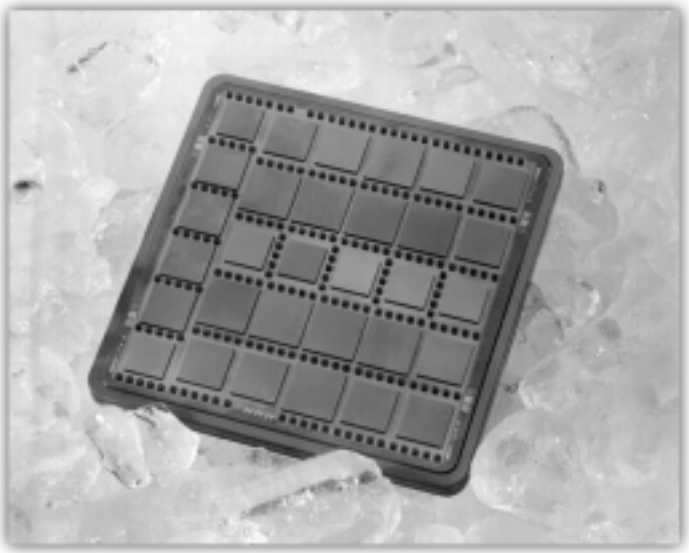


Bild 4: Das Multi-Chip-Modul

gen auf dem Mainframe laufen, ist eine solche präventive Wartung extrem wichtig.
Die eigentliche Recheneinheit der System z befindet sich in einem sogenannten Buch (*Book*). In ihm befinden sich die Prozessoren, sowie Schnittstellen zum Arbeitsspeicher und den Netzwerkanschlüssen. Je nach Modell können ein bis vier dieser Bücher enthalten sein. Zum einfachen Aufrüsten oder zu Reparaturarbeiten kann das System so eingerichtet werden, daß die Bücher im laufenden Betrieb physisch hinzugefügt oder weggenommen werden können, ohne das System zu anzuhalten.

Prozessormodule

Jedes der Bücher verfügt über vier dedizierte Level-2-Speicherchips mit einer Gesamtkapazität von 40 MByte pro Buch, der über vier Hochgeschwindigkeitsbusse an die Prozessoren angeschlossen ist. Da alle Bücher über einen doppelten Speicherring miteinander verbunden sind, verfügt das Gesamtsystem im Maximalausbau über 160 MByte Level 2 Cache.
Pro Buch können in acht Steckkarten bis zu 128 GByte Arbeitsspeicher enthalten sein, so daß das System maximal 512 GByte Gesamtspeicher bei einem theoretischen Durchsatz von 172 GByte pro Sekunde erreicht. Der Arbeitsspeicher verfügt über einen Chipkill Error Correction Code zur Fehlerbehebung von Multi-Bit-Fehlern, sowie über sechzehn Ersatzchips pro Buch, die transparent vom System aktiviert werden können.
Das Herzstück des System z bildet eine hochintegrierte Platine, das Multi Chip Module (MCM), auf der sich auf einer Fläche von gerade einmal 94 mal 94 Millimeter hundertzwei Schichten, zirka vier Milliarden Transistoren und 0,545 Kilometer interne Verbindungen befinden.
Diese teilen sich in vier Cache-Speicherchips, zwei System-Assist-Prozessoren (zum Abarbeiten von I/O-Anfragen, während die Standardprozessoren weiterarbeiten) und in bis zu zwei Ersatzprozessoren sowie bis zu zehn Prozessoren auf, die für Benutzeraufgaben zur Verfügung stehen. Die Prozessoren sind CMOS-10-basierend und bestehen aus jeweils zehn Lagen Kupfer. Zusätzlich befindet sich eine interne Uhr zur Synchronisation der Prozessoren auf jedem Multi-Chip-Modul.

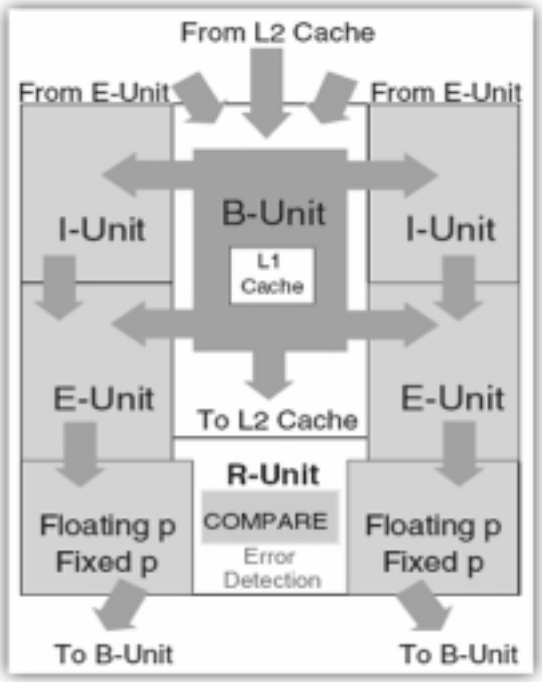


Bild 5: Das doppelte Rechenwerk eines System z9

Verschlüsselung

Die Prozessoren verfügen über eine Reihe an Besonderheiten. Sie arbeiten mit einer Dual-Core-Technologie, um die enorme Packungsdichte zu erreichen. Zusätzlich verfügt jeder Kern noch einmal über ein doppelt ausgelegtes Rechenwerk. Diese zusätzliche Logik steht dem Chip jedoch nicht zur Steigerung seiner Rechenleistung zur Verfügung, sondern dient der Fehlervermeidung. Alle zur Ausführung anstehenden Arbeitsaufträge werden parallel auf beiden Rechenwerken ausgeführt und das Ergebnis verglichen. Nur wenn beide denselben Wert haben, wird das Ergebnis in den Speicher zurückgeschrieben. Stimmt ein Ergebnis nicht, wird die Berechnung erneut durchgeführt. Sollten nach einem dritten Durchgang immer noch Unterschiede vorhanden sein, deaktiviert der Microcode des Systems den entsprechenden Prozessor, leitet die Speicherinhalte des Level-1-Cache in einen Ersatzprozessor und führt auf diesem die Berechnung erneut durch. Dieser Wechsel ist für laufende Anwendungen transparent, das heißt, sie laufen normal weiter, ohne Daten zu verlieren!
Dies könnte man mit einer Operation am offenen Herzen vergleichen, während man nebenher am Arbeiten ist.
Die superskalaren Prozessoren verfügen zusätzlich über Out-of-order Operand Fetching, um jeden Prozessorzyklus auszunut-

zen, sowie eine Branch-Prediction History Table und eine eingebaute Datenkompression.
Insbesondere Umgebungen mit entsprechenden Sicherheitsanforderungen nutzen die Kryptographiefunktionen, die jeder Chip implementiert hat. Sie führen gängige Operationen wie DES, T-DES, MAC und SHA-1 Hashing durch.
Die maximal vierundfünfzig Prozessoren können zusätzlich zu den aufgeführten Besonderheiten in ihrer Rolle speziell logisch definiert werden. Hier besteht die Möglichkeit, sie entweder dediziert für das Betriebssystem Linux (Integrated Facility for Linux, IFL), als Cluster-Verwalter (Internal Coupling Facility, ICF), als I/O-Chip, Datenbank Offload Chip (System z Integrated Information Processor, zIIP) oder als Java Offload Prozessor (System z Application Assist Processor) einsetzen. Technisch gesehen handelt es sich hierbei um die gleiche Physik, jedoch hat eine solche Dedizierung Auswirkung auf die Softwarekosten.

Netzwerk

Der Host unterstützt alle wesentlichen Protokolle wie Gigabit Ethernet und Fibre Channel. Zusätzlich bietet er auch eigene Protokolle mit jeweils eigenen Netzwerkkarten, die im Mainframe-Umfeld auch als Channels bezeichnet werden: ESCON (Enterprise Systems Connection), FICON (Fibre Connection), ein Fibre-Channel-ähnliches Protokoll mit bis zu zwei GByte pro Sekunde, sowie verschiedene mainframe-proprietäre Clusterprotokolle.

Cluster

Das Gehäuse verfügt über drei Bereiche (*Cages*), in die Netzwerkkarten aufgenommen werden können. Jeder Cage bietet Platz für 28 Steckkarten (Adapter), die über mehrere Kanäle verfügen. Jeweils vier Steckplätze sind eine Domäne, die logisch zwei Büchern zugeordnet ist. Die Anbindung an eine Domäne wird mit dem Memory Bus Adapter (MBA) auf Seiten der Bücher, in den Speicher und über Self Timed Interfaces (STI) auf Seiten der I/O-Systeme zu den Netzwerkkarten hin realisiert. Jede dieser Strecken zwischen Memory Bus Adapter und Self Timed Interface verfügt über eine Bandbreite von 2,7 GByte pro Sekunde, die per Multiplexing an die angeschlossenen Karten weitergegeben wird. Um einen optimalen Durchsatz zu erreichen, sollten aus diesem Grund auch viel genutzte Adapter in verschiedenen Domänen liegen. In einem Cluster kann auch der Speicher von mehreren System-z-Rechnern über den Integrated Cluster Bus (ICB) oder die InterSystem Channels (ISC) miteinander verbunden werden, wodurch eine performante Kommunikation von bis zu 2 GByte mit nur geringem Protokolloverhead möglich ist.
Zusätzlich können Kryptographiekarten in die Steckplätze der Netzwerkkarten eingesetzt werden, wenn eine FIPS-140-2-Level-4-zertifizierte Umgebung benötigt wird. Diese Krypto-Express-Karten sind physisch vor Manipulation durch ein Gehäuse geschützt, das dafür Sorge trägt, daß bei einem Öffnen die Inhalte der Karte, das heißt, gespeicherte Schlüssel, unwiderruflich gelöscht werden. Der Funktionsumfang dieser Karten erstreckt sich von Schlüsselmanagement, RSA, Unterstützung für Europay Mastercard VISA (EMV), Public Key Decrypt (PKD) bis zu Public Key Encrypt (PKE).
Man erkennt, daß keine einzelne Komponente das System z maßgeblich charakterisiert, sondern erst die Zusammenarbeit aller Bauteile die Maschine für die besonderen Anforderungen rüstet, wodurch sich auch die Betonung auf »System« im Namen erklärt. ◆

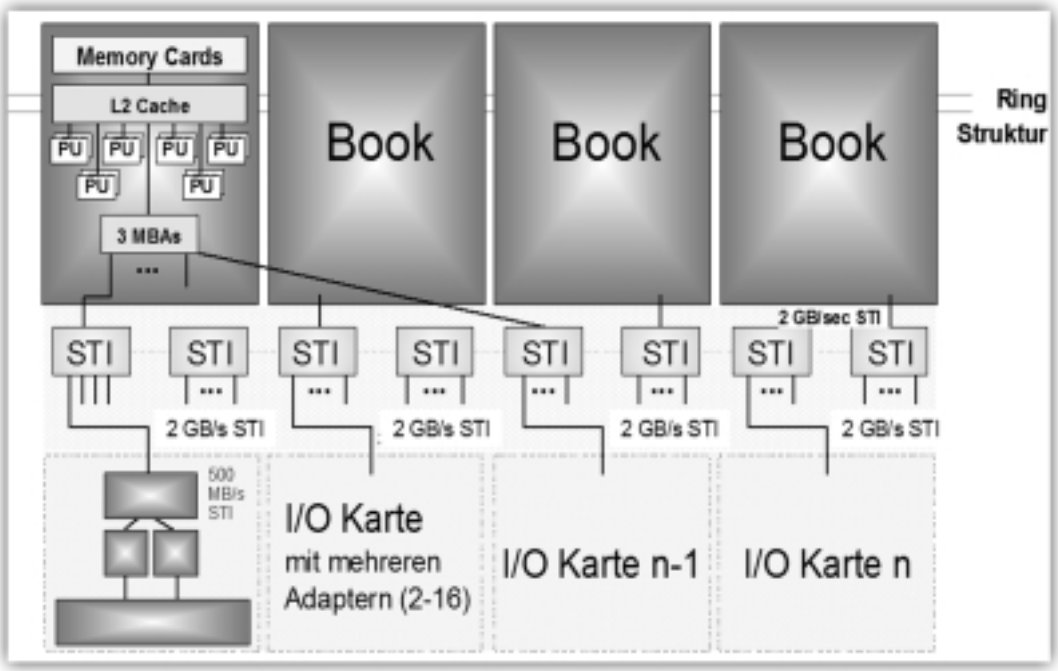


Bild 6:
Aufbau der Netzwerkanbindung in einem System z9

Partitionen auf dem Mainframe

von Martina Schmidt, Thomas Schulze und Michael Störchle

Der Begriff »Virtualisierung« etabliert sich momentan als äußerst populäres Schlagwort in der IT-Szene, dessen Konzept die Struktur der heutigen IT-Landschaft nachhaltig mitgeprägt hat. Nicht zuletzt deswegen wird fälschlicherweise oft angenommen, daß Virtualisierung eine technologische Entwicklung der jüngsten Rechnergeneration ist. Richtig ist, daß der Mainframe durch Technologien wie das Betriebssystem VM oder die PR/SM-Funktion zur logischen Partitionierung als Innovator heutiger Virtualisierungstechnologien gilt.

Grundsätzlich umfaßt Virtualisierung zwei mögliche Richtungen: Zum einen die Integration vieler einzelner physischer Rechner zu einem großen virtuellen System, zum anderen die Segmentierung eines großen physischen Rechners zu vielen einzelnen virtuellen Systemen. Die hier vorgestellten Virtualisierungstechnologien des Mainframes konzentrieren sich auf letztere Richtung. Dementsprechend ermöglicht die Zergliederung der Mainframe-Ressourcen in viele unabhängige virtuelle Systeme die gleichzeitige Ausführung mehrerer voneinander unabhängiger Betriebssysteme in verschiedenen Instanzen auf einem einzigen physischen Rechner.

Derartige virtuelle Systeme beziehungsweise Instanzen werden als »virtuelle Maschinen« bezeichnet. Eine virtuelle Maschine ist eine vollständig geschützte und isolierte virtuelle Kopie der zugrundegelegten Hardware. Somit erhält jedes in einer virtuellen Maschine laufende Betriebssystem, ein sogenannter Gast, die Illusion, über ein dediziertes physisches System zu verfügen. Auf dem Mainframe nimmt sich als Softwarelösung das Betriebssystem VM dieser Aufgabe an, indem es die virtualisierte Hardware für seine Gäste bereitstellt.

Das Betriebssystem VM

Die Wurzeln von VM gehen bis in die sechziger Jahre zurück. Damals wurde an der Universität Cambridge das sogenannte CTSS (Conversional Time Sharing System) entwickelt, ein Betriebssystem zur Aufteilung eines Zentralrechners in mehrere virtuelle Rechner. 1965 wurde das CTSS erstmals für einen IBM Mainframe, das IBM/360 Model 40, umgeschrieben. Dieses Projekt beruhte immer noch auf dem Engagement der Universität Cambridge. Erst ab 1967 unterstützte IBM die Weiterentwicklung dieses Betriebssystems. Als erstes kommerzielles Produkt entstand das Betriebssystem CP-67, der direkte Vorfahre des 1970 auf den Markt eingeführten VM/370. Bis zur heute aktuellen Version z/VM 5.2 hat sich das Betriebssystem ständig weiterentwickelt, die Abwärtskompatibilität jedoch bewahrt.

Grundsätzlich besteht VM aus einem Hypervisor, der CP (Control Programm) genannt wird, sowie aus einem interaktiven Betriebssystem, das besonders auf das Arbeiten mit virtuellen Maschinen und deren Administration ausgelegt ist, dem CMS (Conversional Monitoring System). Die Schnittstelle zwischen dem VM CP und seinen Gästen ist eine simulierte Mainframe-Hardware. Praktisch bedeutet dies, daß der VM-CP-Kernel die realen Systemressourcen auf virtuelle abbildet. Ein Gast ist sich somit der Existenz eines Hypervisors nicht bewußt und arbeitet so, als sei er alleiniger Benutzer eines Mainframes. Die Zuordnung von realen Systemressourcen zu einzelnen Gästen erfolgt nach dem Zeitscheibenverfahren. Das bedeutet, der VM-CP-Kernel als Verwalter der Systemressourcen übergibt einem Gast für

eine gewisse Zeitspanne die Kontrolle über die zugeordneten Systemressourcen und nimmt sie anschließend wieder weg.

Eine virtuelle Maschine wird in VM durch eine Datenstruktur dargestellt, den sogenannten Kontrollblock. Diese Datenstruktur enthält Angaben über den zugeordneten Hauptspeicher, Plattenspeicherplatz, I/O-Konfigurationen sowie andere Ressourcen. Die Umsetzung dieser Datenstruktur erfolgt durch den Prozeßleitblock des CP-Kernels sowie darin enthaltene Zeiger auf weitere zugehörige Kontrollblöcke.

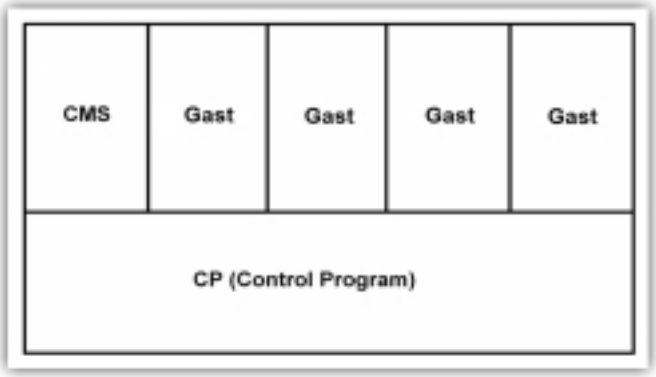


Bild 1: Grundsätzlicher Aufbau von VM

Gast-Betriebssysteme werden vom VM-CP wie Prozesse behandelt, die in unabhängigen virtuellen Adreßräumen laufen. Die Funktion des VM-CP-Kernels als Verantwortlicher für die semantische Konsistenz der zugrundegelegten Systemressourcen macht es notwendig, daß der Kernel eines Gast-Betriebssystems in seiner Funktionalität eingeschränkt wird. Daher arbeitet der VM-CP-Kernel im Kernel-Modus, der Gast-Kernel dagegen lediglich im eingeschränkten Benutzer-Modus. Diese Zuweisung von Modi ist wichtig, da es eine Reihe von Instruktionen gibt, die nur im Kernel-Modus ausgeführt werden können. Diese Instruktionen werden als »privilegiert« bezeichnet. Unabhängig von dieser Klassifikation können Instruktionen auch »sensitiv« sein. Derartige Instruktionen greifen direkt auf kritische Hardwarebereiche zu und würden, wenn die Ausführung im Benutzer-Modus erlaubt wäre, andere virtuelle Maschinen und deren Gäste beeinflussen. Ein Beispiel dafür sind Zugriffe auf Kontrollregister.

Virtuelle Ressourcen

Im Instruktionssatz des Mainframes sind sensitive Instruktionen gleichzeitig privilegiert. Das macht die Interpretation der sensitiven Instruktionen besonders einfach, da ihre Ausführung durch den Gast automatisch eine Unterbrechung auslöst und einen Aufruf des VM-CP-Kernels verursacht. Die weitere Ausführung der Instruktion obliegt dem VM-CP-Kernel. Somit ist sichergestellt, daß die Semantik der Systemressourcen für alle Gäste erhalten bleibt. Aufgrund der Gleichstellung von sensitiven mit privilegierten Instruktionen wird der Mainframe auch als virtualisierbare Rechnerarchitektur bezeichnet.

Die Verwaltung realer Systemressourcen und deren virtuelle Abbildung sowie Bereitstellung für die Gäste ist Hauptaufgabe des VM-CP. Um sie ausführen zu können, muß das VM-CP sämtliche zu virtualisierenden Systemressourcen für sich strukturieren.

Eine virtuellen Zentraleinheit wird durch Zeitscheiben der realen Zentraleinheit abgebildet. Die Länge dieser Zeitscheiben bestimmt die Zeitdauer, die ein Gast die CPU in Anspruch nehmen darf. Sie ist variabel und hängt von verschiedenen Faktoren ab, beispielsweise der Anzahl der aktiven Gäste, der Anzahl zugewiesener realer Zentraleinheiten sowie der Priorität eines Gastes.

Der zugewiesene Hauptspeicher eines Gasts unterliegt dem Paging-Mechanismus des VM-CP-Kernels. Nachdem heutzutage alle für VM in Frage kommenden Gast-Betriebssysteme ihrerseits fähig sind, virtuellen Speicher und Paging zu unterstützen, ist beim Speicherzugriff eine zwei- oder mehrstufige Adreßumsetzung notwendig. Dabei wird unter Verwendung der Segment- und Seitentabellen des Gastes die virtuelle Adresse des Gast-Speichers in

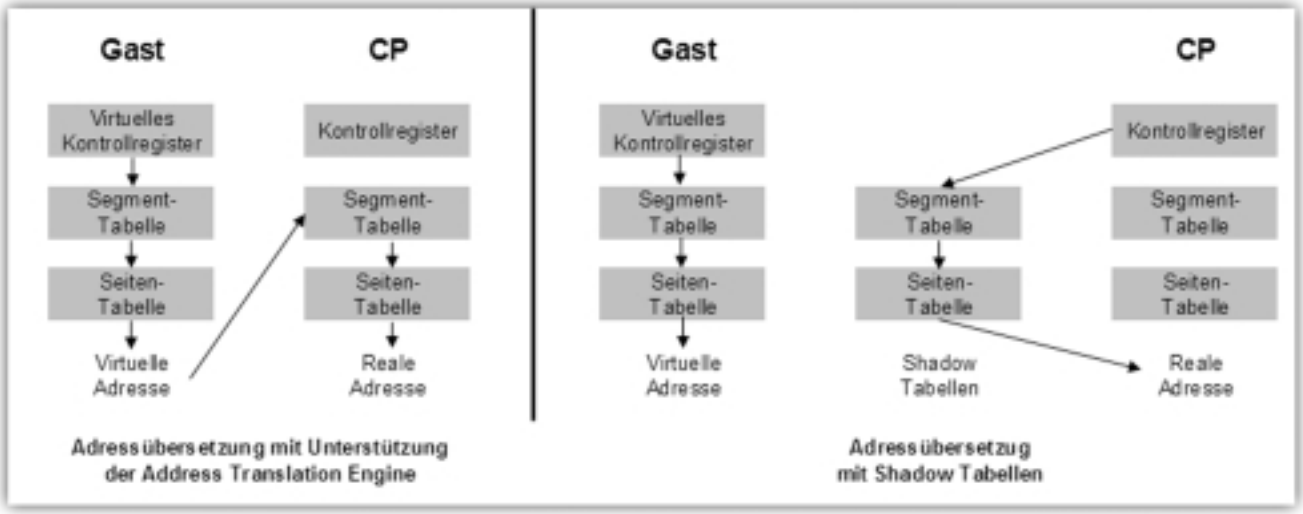


Bild 2: Vergleich der Adreßumsetzungen zwischen S/390 und Vorgängermodellen

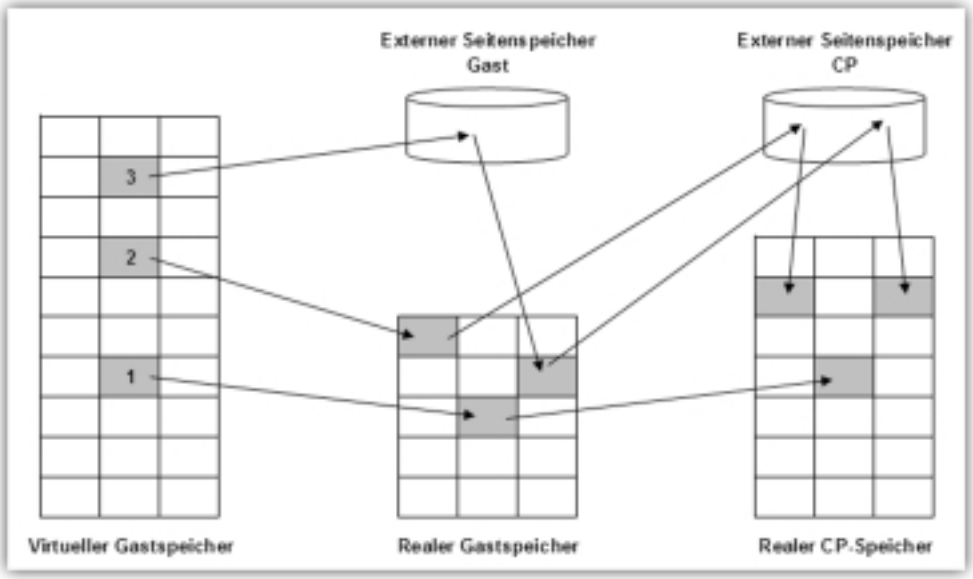


Bild 3: Adreßumsetzung bei VM

eine vom Gast als real angenommene Adresse umgesetzt.
Diese ist aus Sicht des VM-CP-Kernels wiederum eine virtuelle Adresse, die anhand der Segment- und Seitentabelle des VM-CP-Kernels auf eine reale Hauptspeicheradresse umgesetzt wird. Seit der Rechnerarchitektur S/390 wird jeder Vorgang einer Adreßumsetzung dabei vollständig durch die in der Hardware integrierten Address Translation Engine ausgeführt. Diese hardwareseitige Implementierung bewirkt, daß zwischen dem real zugewiesenen Speicher des Gasts und dem realen Speicher des VM-CP-Kernels keine Shadow-Tabellen zur Synchronisation mehr erforderlich sind. Derartige Shadow-Tabellen waren vor Einführung der Rechnerarchitektur der S/390 notwendig. Sie wurden durch den VM-CP-Kernel verwaltet und bildeten die virtuellen Adressen des Gastes direkt auf die realen Adressen des Hosts ab.
Das nachfolgende Beispiel verdeutlicht den Paging-Mechanismus ab der Generation S/390. Hierbei greift das System auf drei Seiten zu. Seite 1 befindet sich im realen Gastspeicher sowie im realen Hauptspeicher, weshalb kein Paging erforderlich ist. Seite 2 befindet sich im realen Gastspeicher (der aus Sicht des VM-CP virtuell ist), aber nicht im realen Speicher des VM-CP. Das VM-CP muß daher eine Seitenersetzungsoperation durchführen. Seite 3 befindet sich weder im realen Gastspeicher noch im realen VM-CP-Speicher. Der Gast-Kernel muß in diesem Fall die Seitenersetzung abarbeiten. Dazu liest der Gast-Kernel diese Seite von der Platte und übergibt sie dem externen Seitenspeicher des VM-CP. Der VM-CP-Kernel holt daraufhin diese Seite in seinen realen Speicher.

Interpretive Execution Facility

Moderne Plattenspeichersysteme werden durch VM auf der realen Seite unterstützt, müssen auf der virtuellen Seite jedoch, bedingt durch die jeweiligen Gäste, als anderer Plattentyp dargestellt werden. Ein angesprochenes Plattenspeichersystem wird durch VM daher ebenso virtualisiert.
In der Version z/VM 5.2 können die Plattenspeicher eines Subsystems beispielsweise als Platten des Typs 3380, 3390, 9345 oder FBA emuliert werden. Die virtuellen Plattenspeicher werden durch VM als sogenannte Minidisks (was nichts mit dem magnetisch-optischen Speichermedium der Firma Sony zu tun hat) verwaltet. Eine Minidisk besteht aus zusammenhängenden Datenbereichen auf einer Platte, dessen Anfangsadresse, die Nummer des Startzylinders, dem VM-CP-Kernel bekannt ist. Die Größe einer Minidisk wird durch die Anzahl der zugewiesenen Blöcke definiert. Ein Gastbetriebssystem adressiert die Blökke der Minidisk beginnend bei 0. Allen I/O-Operationen ist gemein, daß der VM-CP-Kernel die Anfangsadresse der Minidisk aufaddiert. Dadurch entfällt der Zugriff mittels File-Directory.
Nur durch kontinuierliche Weiterentwicklung der Hardwarearchitektur und von dem maßgeschneidertem Betriebssystem VM wurde eine derart leistungsstarke Virtualisierung, wie sie der Mainframe bietet, möglich. Nachfolgend nur eine kleine Facette dieser vielseitigen technologischen Evolution, die aufzeigt, daß reine Softwarelösungen nur ein schwacher Ansatz für die komplexe Problematik der Virtualisierung sind.
Mit Einführung der Mainframegeneration S/390 wurden die erläuterten Virtualisierungsmechanismen in ihrem Leistungsverhalten verbessert. Diese Leistungssteigerung wurde unter anderem durch eine Erweiterung der Hardwarearchitektur, der sogenannten Interpretive Execution Facility (IEF), erreicht. Sie ermöglicht es, eine Untermenge der privilegierten Maschineninstruktionen eines Gasts direkt durch den Gast-Kernel ohne Inanspruchnahme des CP-Kernels abzuarbeiten.
Die IEF erweitert die oben erwähnte Funktionsweise von Kernel- und Benutzermodus um einen Host- und Gastmodus. Durch Aufruf der Instruktionen *Start Interpretive Execution* (SIE) startet der VM-CP-Kernel einen Gast. Dabei übergibt er dem Gast die Kontrolle über die zugeordneten Systemressourcen. Der Gast seinerseits kennt Kernel- und Benutzer-Modus.
Durch die Architekturerweiterungen der IEF wurde ein zweiter Satz Kontrollregister implementiert. Dadurch können die meisten privile-

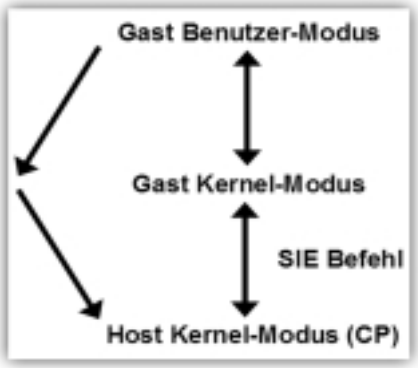


Bild 4: Modi-Übergänge

gierten Maschineninstruktionen des Gast-Kernels im Gast-Modus selbst ausgeführt werden. Privilegierte Instruktionen, die weiterhin den VM-CP-Kernel erfordern, führen zu einer Unterbrechung des Gast-Modus (SIE Intercept). Dies beendet die SIE-Instruktionen und übergibt die Kontrolle dem VM CP.

PR/SM und logische Partitionierung

Die SIE-Instruktion ist sehr komplex und wurde sowohl über Firmware (sogenannten Mikrocode) als auch durch Hardware-Erweiterungen in der CPU implementiert. Ihr Operand zeigt auf einen State Descriptor, den der VM-CP-Kernel für jeden Gast im Hauptspeicher anlegt. Ein State Descriptor speichert den Zustand eines konkreten Gasts, wie beispielsweise Hauptspeicher und Zeitgeber. Zudem sind alle Kontrollregister sowie Teile der Mehrzweckregister abgebildet. Diese Register werden beim Aufruf der SIE-Instruktion geladen und bei Beenden zurückgeschrieben. Im State Descriptor wird darüber hinaus festgehalten, für welche privilegierten Maschineninstruktionen SIE aufgerufen werden soll.

Neben den erläuterten Virtualisierungsmechanismen auf Betriebssystemebene besitzt der Mainframe auch die Möglichkeit zur Virtualisierung und logischen Partitionierung auf Mikrocode-Ebene, der sogenannten PR/SM-Funktion.

Der *Processor Resource/System Manager* (PR/SM) ist ein Teil des Mikrocodes, der historisch aus VM hervorgegangen ist. PR/SM ist ein Hypervisor, der es erlaubt, einen physischen Rechner in mehrere virtuelle Maschinen zu partitionieren. Dabei läuft jede virtuelle Maschine in ihrer eigenen logischen Partition (LPAR).

Jede virtuelle Maschine verfügt über ein eigenes Betriebssystem, einen eigenen unabhängigen realen Hauptspeicher, eigene Kanäle und eigene Ein-/Ausgabegeräte. Eine logische Partition kann aber nur über die Ressourcen verfügen, die ihr zugeteilt wurden. Diese Ressourcenvergabe kann auch im laufenden Betrieb erfolgen. Durch entsprechende Konfiguration lassen sich Ressourcen auch von mehreren LPARs gemeinsam nutzen.

Eine LPAR kann folgendermaßen definiert werden: Sie verfügt über CPU-Ressourcen (von Bruchteilen bis zu 32 CPUs), Hauptspeicher und Kanäle. Pro Mainframe können maximal sechzig LPARs definiert werden. Normalerweise laufen virtuelle Maschinen in einem virtuellen Adreßraum, der von einem Hypervisor überwacht wird. Im Gegensatz dazu laufen PR/SM-Gastmaschinen in einem realen Adreßraum, so daß eine doppelte Adreßübersetzung entfällt. Bei der logischen Partitionierung durch PR/SM wird der physische Speicher in reale Adreßräume (Zonen) aufgeteilt. Jede Zone hat einen zusammenhängenden Namensraum. Die Gastmaschine spricht diese realen Adressen direkt an.

Gemeinsam genutzte Prozessoren

Die Größe einer Zone läßt sich auch dynamisch verändern, wofür 64-MByte-Blöcke auf einem nicht zusammenhängenden Teil des physischen Speichers zur Verfügung stehen.

Kanäle werden in der Regel einer LPAR fest zugeordnet.

Es ist möglich, in einer LPAR z/VM zu installieren und über z/VM wiederum Gastbetriebssysteme. In diesem Fall spricht man von *Second Level Guests*.

Da es sich bei LPARs um virtuelle Konstrukte handelt, werden die Prozessorressourcen als logische Prozessoren bezeichnet. Eine LPAR kann sich hierbei entweder aus einem gemeinsamen Pool an realen Prozessoren bedienen oder sie erhält dedizierte Prozessorleitung. Bild 7 zeigt

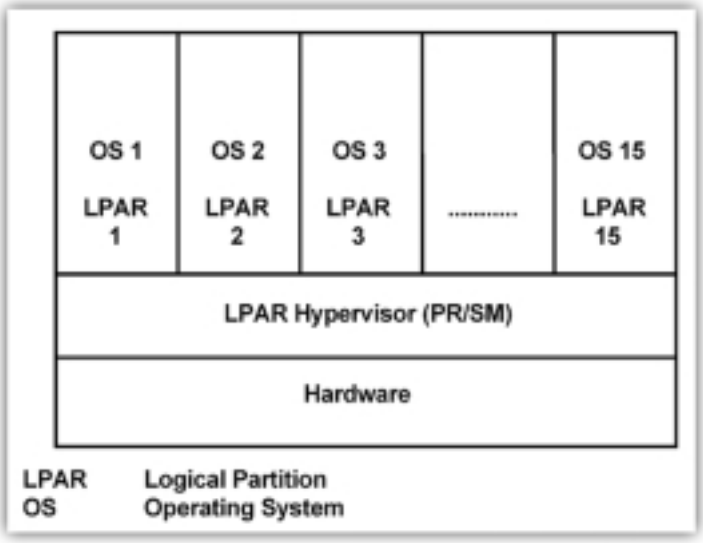


Bild 5: IBM PR/SM und LPAR

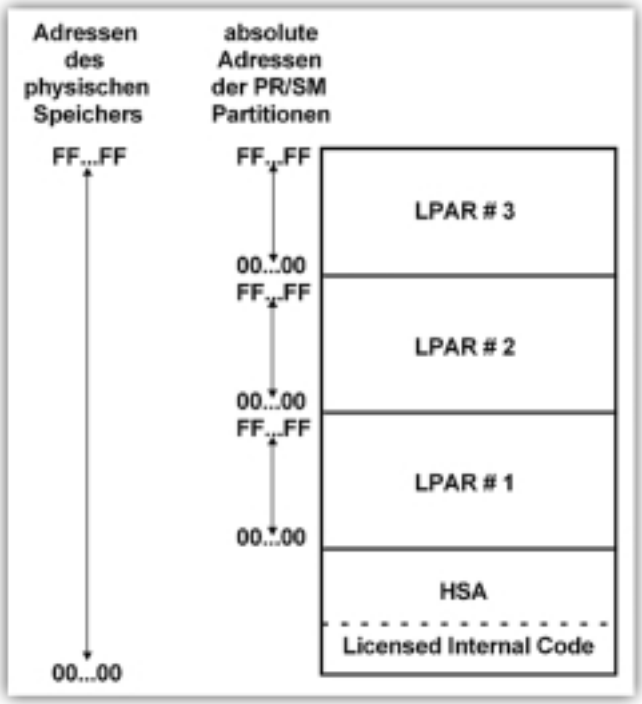


Bild 6: PR/SM und LPAR-Speicheraufteilung des physischen Speichers

einen Mainframe mit zehn physischen Prozessoren. Wenn diese den Status *dedicated* haben, werden sie permanent einem logischen Prozessor einer LPAR zugeordnet. Diese Konfiguration bedeutet weniger LPAR-Overhead. Der Performanceverlust gegenüber dem gleichen Server, der im Basic-Modus betrieben wird, ist vergleichsweise marginal.

Wenn Prozessoren von mehreren LPARs gemeinsam genutzt werden, ist der Overhead größer. Er steigt linear mit dem Verhältnis von logischen Prozessoren, die in allen aktiven LPARs definiert sind, zu der Anzahl gemeinsam genutzter physischer Prozessoren.

Wenn ein Betriebssystem, das *shared* Prozessoren nutzt, in einen Wartezustand fällt, gibt es den physischen Prozessor im Normalfall frei, so daß er dann von anderen LPARs genutzt werden kann.

LPAR-Gewichtungen

LPAR-Gewichtungen (engl. *weights*) kontrollieren die Verteilung der gemeinsam benutzten Prozessoren zwischen den LPARs. Die Gewichtung einer LPAR bestimmt im Verhältnis die garantierte (minimale) Prozessorleistung, die sie bei Bedarf erhält, selbst wenn das System physisch voll ausgelastet ist. Das stellt sicher, daß kritische Systeme zu jedem Zeitpunkt mit einer garantierten Antwortzeit arbeiten, selbst wenn die Prozessorauslastung bei 99 Prozent liegt.

Eine LPAR kann weniger als die ihr garantierte Anzahl von Ressourcen nutzen, wenn sie nicht viel zu tun hat. Genauso kann sie auch mehr Leistung nutzen, als ihr zustehen, wenn andere LPARs ihre Ressourcen nicht komplett ausnutzen, sofern man dies nicht mit einem sogenannten Capping verhindert, das eine Maximalleistung festlegt. Meist werden Testsysteme mit einem solchem Capping versehen, damit sie die Produktion nicht behindern, oder für den Fall, daß die Service Level Agreements keine höheren Anforderungen haben.

Intelligent Resource Director

Der *Intelligent Resource Director* (IRD) ist eine Erweiterung des PR/SM- und LPAR-Konzeptes. Die Aufgabe des IRD ist es, Prozessor- und Kanalressourcen über die verschiedenen LPARs zu optimieren. Die wichtigsten Funktionen sind LPAR CPU-Management, Dynamic Channel Path-Management und Channel Subsystem Priority Queuing.

Die IRD-Funktionen werden vor allem von der z/OS-Komponente *Work Load Manager* (WLM) genutzt. Das Ziel von IRD und WLM ist es, die Nutzung von Systemressourcen zu optimieren und somit den gesamten Systemdurchsatz zu steigern.

Das LPAR-CPU-Management wird von WLM genutzt. Darüber können die Gewichte und die Anzahl der zugeordneten CPUs von z/OS LPARs im *shared* Modus dynamisch und transparent geändert werden, ohne daß Anwendungen angehalten werden müssen oder manuelle Interaktion durch einen Administrator notwendig sind.

Kanäle gehören, wie CPU-Zeit und Hauptspeicherplatz, zu den kritischen Systemressourcen, die entweder fest zugeordnet sind oder mehreren

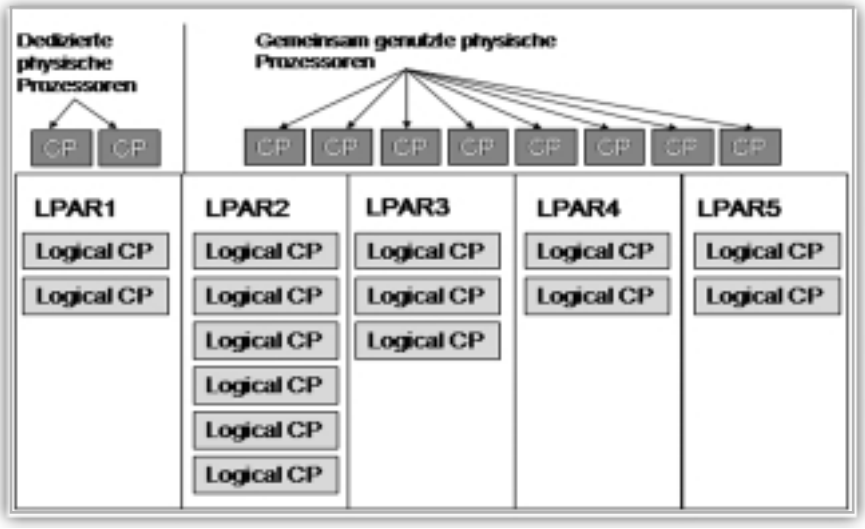


Bild 7: Beispiel für gemeinsam genutzte Prozessoren

LPARs zur Verfügung stehen. Das dynamische Kanalmanagement ermöglicht die Optimierung der Zuordnungen, beispielsweise durch kurzzeitige, exklusive Verwendung durch eine LPAR. Ein- und Ausgabeanforderungen werden in Warteschlangen (*Queues*) gestellt, bevor sie abgearbeitet werden können. Dies geschieht im Normalfall nach dem FIFO-Prinzip (first in, first out). Die dritte wichtige Funktion des IRD, das Channel Subsystem Priority Queuing, ermöglicht dem WLM, diese Reihenfolge zu verbessern.

Sicherheit

Die Sicherheitseigenschaften der PR/SM-Architektur gehen über die anderer virtueller Maschinenkonzepte hinaus. Die wichtigsten Unterschiede sind wohl, daß logische Partitionen durch den Hypervisor voneinander getrennt sind und somit nicht miteinander kommunizieren können; Hauptspeicherbereiche verschiedener LPARs können sich, durch den Mikrocode kontrolliert, nicht überlappen. Ein Kanal, der einer LPAR fest zugeordnet ist, kann von einer anderen LPAR nicht angesprochen werden. Ist der Kanal so konfiguriert, daß er von mehreren LPARs genutzt werden kann, sorgt er selbst für die Trennung der verschiedenen Auftraggeber. Laut einer US-Zertifizierung können PR/SM-LPARs so eingerichtet werden, daß sie die gleichen Sicherheitseigenschaften wie räumlich voneinander getrennte Rechner aufweisen.

Erst durch die Möglichkeiten der Virtualisierung durch PR/SM und z/VM kann die Hardware der System z9 optimal – üblicherweise nahe der 100 Prozent – ausgelastet werden. Nach näherer Kenntnis der Virtualisierungsmöglichkeiten und Hardware-Eigenschaften kann man den in den TV-Werbespots in Szene gesetzten Anspruch, System z könne ganze Rechenzentren voll von x86-Servern ablösen, für einen recht pragmatischen Ansatz halten. ◆

Literatur

Joachim von Buttlar und Wilhelm G. Spruth: Virtuelle Maschinen: zSeries und S/390 Partitionierung. IFE-Informatik Forschung und Entwicklung, Heft 1'2004.

ABCs of z/OS System Programming Volume 10, IBM Redbook, SG24-6990-01, *http://www.redbooks.ibm.com/abstracts/sg246990.html?Open*

Short History of IBM's Virtual Machines: *http://www.cap-lore.com/Software/CP.html*

IBM: z/VM Literature: *http://www.vm.ibm.com/library/*

Universitätsvorlesungen: *http://www-ti.informatik.uni-tuebingen.de/~spruth/*

IBM System z9 Enterprise Class Technical Guide, IBM Redbook, SG24-7124-00, *http://www.redbooks.ibm.com/abstracts/sg247124.html?Open*

Impressum

freeX

Das Magazin für freie Unix-Systeme
Eine Publikation des Computer- und Literaturverlag GmbH
Herausgeberin: Rosa Riebl

Verlagsanschrift:
C&L Computer- und Literaturverlag GmbH
Zavelsteinerstr. 20, D-71034 Böblingen
E-Mail: freeX@cul.de
Redaktion: *Jörg Braun (jb), Rosa Riebl (rr)*
Projekt-Koordination: *Herwig Diessner*
Bildredaktion: *Mohamed Hawa (mh)*
Titelmotiv: *Abbildung IBM*

Autoren dieser Ausgabe:
Isabel Arnold, Christian Daser, Selita Faller, Andreas Gröschl, Paul Herrmann, Manuel Müller, Martina Schmidt, Thomas Schulze, Fred Stefan, Michael Störchle, Benjamin Storz, Christian Strauer, Christine Urban, Dirk Ziesemann

Redaktionsanschrift:
freeX, Büro Luvigny, 10 Rue des Hauts Champs,
F-88110 Luvigny
Tel.: (0033) 3.29.42.40.02, Fax: (0033) 3.29.42.40.03
E-Mail: freex@cul.de

Anzeigenverkauf:
Jörg Kalies
Dorfstr. 60
85235 Unterumbach
Tel.: 0 81 34 / 93 10-0
Fax: 0 81 34 / 93 10-30
Es gilt die Anzeigenpreisliste vom 15.03.2005

Layout/DTP/Grafik:
Hawa & Nöh Grafikdesign, Neu-Eichenberg, www.hn-grafik.de

Druck/Herstellung:
Kössinger AG, Schierling
Printed in Bavaria

Die größte Kaffeedose der Welt

von Selita Faller, Christian Strauer, Thomas Schulze und Dirk Ziesemann

Java ist überall, vom Handheld über das Telefon bis hin zu Serversystemen. Natürlich auch auf dem Mainframe!

Als IBM von Sun Microsystems 1997 die erste Referenzimplementierung der Java Virtual Machine für Unix-Systeme erhielt und auf z/OS anpaßte, fielen massive Performanceprobleme im Bereich des Sechzig- bis Hundertfachen gegenüber der Standardedition auf.
Da der Aufwand zum Lösen dieser Probleme vertretbar war, wurde die Virtuelle Maschine nicht komplett neu implementiert, sondern man löste sukzessiv die Performanceprobleme, unter anderem im Bereich der Codepage-Konvertierung zwischen ASCII und EBCDIC sowie beim Locking, Threading, Serialisierung, Nutzen von Systemdiensten und der Speicherverwaltung.
Hinzu kam die Entwicklung des Just-in-Time-Compilers JIT, der häufig durchlaufene Codeteile als ausführbaren Bytecode direkt im Speicher ablegt. Bei einem erneuten Aufruf wird der Bytecode dann direkt ausgeführt und nicht erneut interpretiert.
Außerdem wurde der sogenannte Shared Class Cache entwickelt, der dafür sorgt, daß bestimmte Java- oder auch vom User angegebene Klassen von mehreren JVMs gelesen werden können, was die Ladezeit einer JVM erheblich verbessert.
Im Laufe der Zeit entstand durch diese und weitere Anpassungen ein leistungsfähiges Java-Paket auf dem Mainframe, das in vielen Bereichen Technologievorreiter war.
Mit der Version 5.0 der JVM wurde deswegen beschlossen, eine gemeinsame Codebasis der JVM über alle Plattformen (Windows, Linux, AIX, z/OS) hinweg zu haben, die unter anderem auch den JIT und das Konzept der Shared Classes implementiert.

Integrative Funktionen

Neben den Grundfunktionen der IBM-JVM auf dem Großrechner und auf verteilten Plattformen unterscheidet sich Java auf dem Mainframe nach wie vor durch plattformspezifische Erweiterungen gegenüber dem »normalen« PC-Java. Zu den interessantesten Features zählt der JDBC-Treiber des Typs II, den es nur für DB2 unter z/OS gibt. Er ermöglicht es Java-Programmen, sich durch Speicher-zu-Speicher-Kommunikation direkt mit der Datenbank zu verbinden, falls sich DB2 und das Java-Programm in der gleichen LPAR befinden. Gegenüber dem normalen Type-IV-Treiber hat das den Vorteil, daß der TCP/IP- und physische Netzwerk-Overhead bei einem Call zur Datenbank übergangen werden und so Geschwindigkeitssteigerungen im zweistelligen Prozentbereich erzielt werden. Für den Java-Programmierer ist dieser Unterschied nur minimal, weil er lediglich die Server- und Port-Adresse des Datenbankservers beim Aufruf wegläßt:

```
...
Connection con;
...
Class.forName(
  "com.ibm.db2.jcc.DB2Driver");
# Bei der Nutzung von JDBC Type IV:
con = DriverManager.getConnection(
      jdbc:db2://SERVER:PORT/DATABASE);
# Alternativ bei der Nutzung von JDBC-
# Type-II-Änderung auf:
con = DriverManager.getConnection(
      jdbc:db2://DATABASE);
...
```

Neben diesem besonderen JDBC-Treiber verfügt Java auf z/OS über Erweiterungen für den Zugriff auf

spezielle z/OS-Dateiressourcen wie Datasets oder VSAM und die Integration von Java-Programmen in die ausgeklügelte z/OS-Systemautomation oder die Nutzung spezieller Cryptokarten bei der Verschlüsselung.
Java kann auf dem Mainframe – genauso wie auf verteilten Plattformen – für die verschiedensten Zwecke eingesetzt werden.

Von der Lochkarte zum Objekt

Als der Mainframe geboren wurde, basierte er durch seine Lochkarteneingabegeräte hauptsächlich auf der Stapelverarbeitung. Diese Batchverarbeitung hat bis heute eine kontinuierliche Evolution erlebt: Im Laufe der Zeit wurden die Batchmöglichkeiten des Großrechners sukzessive weiter ausgebaut, so daß der Mainframe auf dem Servermarkt aufgrund der langen Erfahrungen nach wie als unangefochtene Batchmaschine gilt. Hunderte bis Tausende Batchjobs können neben dem Onlinebetrieb problemlos parallel laufen!
Java wurde komplett in dieses Umfeld integriert, das heißt, Java-Programme können wie klassische z/OS-Batchjobs abgeschickt werden. Das geschieht über die sogenannte Job Control Language (JCL). Sie erlaubt das Definieren und Parametrisieren von Batchjobs, um sie nach dem Abschicken durch das z/OS Job Entry Subsystem (JES) verwalten zu lassen.

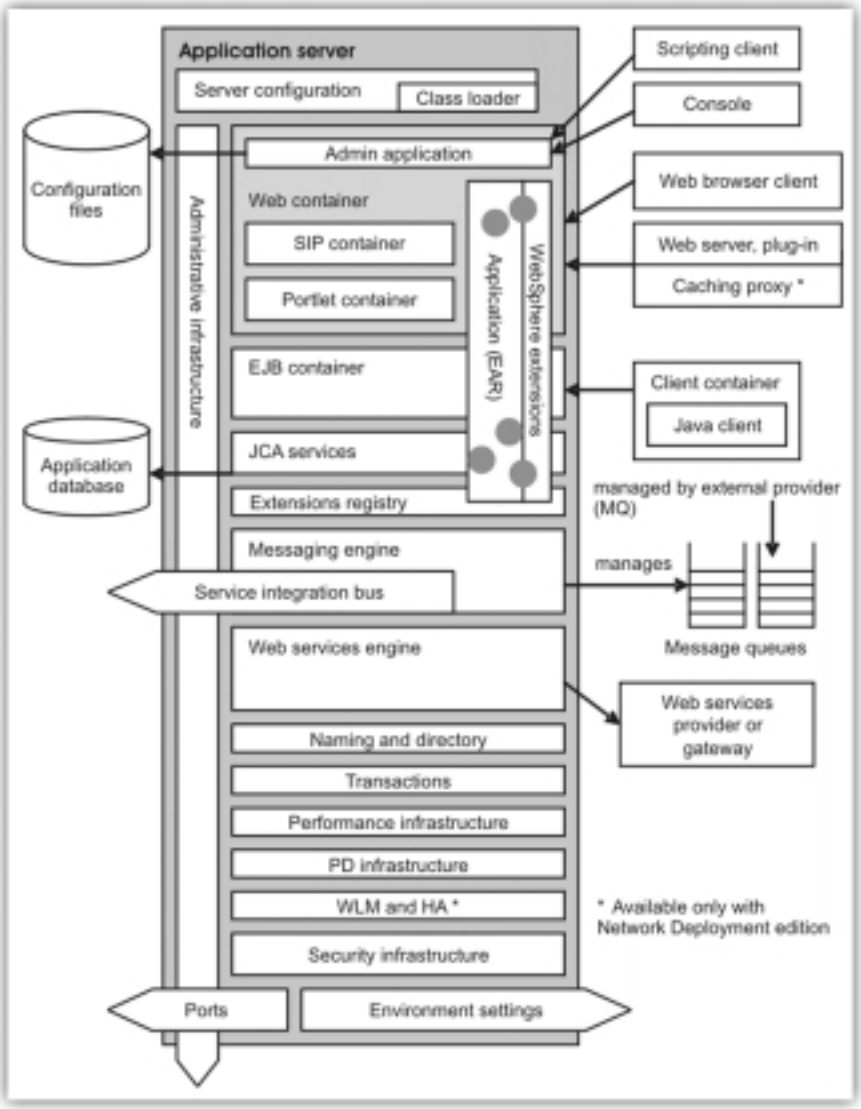


Bild1: Aufbau eines WebSphere Application Servers

Für ein Hello-World-Programm in Java könnte die JCL zum Beispiel folgendermaßen aussehen:

```
//ZUSER30A JOB
//PROCLIB JCLLIB ORDER=SYS1.PROCLIB
//JAVA EXEC PROC=JVMPRC14,
// JAVACLS='com.ibm.bank.sample.Main'
//OUTPUT DD SYSOUT=*
//STDENV DD *
# This is a shell script which
# configures any environment variables
#  for the Java JVM. Variables must be
# exported to be seen by the launcher.

. /etc/profile
export APPL_HOME=/u/zuser30/myjava/\
ibmbank
export JAVA_HOME=/usr/lpp/java/J1.4
export PATH="$PATH":"${JAVA_HOME}"/bin:

LIBPATH="$LIBPATH":"${JAVA_HOME}"/bin
LIBPATH="$LIBPATH":"${JAVA_HOME}"\
/bin/classic
LIBPATH="$LIBPATH":"${JZOS_HOME}"
export LIBPATH="$LIBPATH":

# Customize your CLASSPATH here
# Add application home directory
# and jars to CLASSPATH
for i in "${APPL_HOME}"/*.jar; do
    CLASSPATH="$CLASSPATH":"$i"
done
export CLASSPATH="$CLASSPATH":
# Set JZOS specific options
# Use this variable to specify encoding
# for DD STDOUT and STDERR
#export JZOS_OUTPUT_ENCODING=Cp1047
# Use this variable to prevent JZOS
# from handling MVS operator commands
#export JZOS_ENABLE_MVS_COMMANDS=false
# Use this variable to supply additional
# arguments to main
#export JZOS_MAIN_ARGS=""

# Configure JVM options
IJO="-Xms16m -Xmx128m"
# Uncomment the following line if you
# want to debug the application
#IJO="$IJO -Xdebug -Xrunjdwp:\
#transport=dt_socket,server=\
#y,address=8000"
IJO="$IJO -Djzos.home=${JZOS_HOME}"
# Uncomment the following if you want
# to run without JIT
#IJO="$IJO -Djava.compiler=NONE"
# Uncomment the following if you want
# to run with Ascii file encoding..
#IJO="$IJO -Dfile.encoding=ISO8859-1"
export IBM_JAVA_OPTIONS="$IJO "

export JAVA_DUMP_HEAP=false
export JAVA_PROPAGATE=NO
export IBM_JAVA_ZOS_TDUMP=NO
```

Die JCL beschreibt in diesem Beispiel neben einigen z/OS-spezifischen Parametern die klassischen Java-Parameter wie *MAINCLASS*, *CLASS-*

PATH, *PATH* und spezifische JVM-Parameter wie das Encoding. Klassische z/OS-Batchjobs laufen häufig über Nacht, während die Maschine weniger ausgelastet ist. In diesen Jobs werden dann beispielsweise massenhaft Rechnungen erstellt, die am nächsten Tag per Post verschickt werden. Aufgrund der diversen technischen Möglichkeiten von Java kommt durch Java-Batch erweiterte Funktionalität hinzu. Beispielsweise ist es nun möglich, neben den klassischen Ausdrucken auch PDFs dieser Rechnungen per Java zu erstellen und danach über E-Mail zu verschicken.
Für den Programmier ist die Entwicklung der Java-Programme sehr einfach, er kann beispielsweise die Eclipse-IDE für die Batchentwicklung mit einem besonderen Plug-in nutzen.

WebSphere Application Server

Die Java Standard Edition ist Grundlage für alle Java-Anwendungen. Für Unternehmen von enormer Bedeutung ist vor allem die Java Enterprise Edition JEE, früher als J2EE bekannt. Anders als der Name vermuten läßt, handelt es sich bei J2EE nicht um eine eigene Sprache, sondern um eine Menge von Schnittstellen und Standards. Typischerweise werden Implementierungen dieser Schnittstellen und Standards »Application Server« genannt.
Ein solcher besteht im wesentlichen aus zwei Containern: dem Web-Container für Servlets und JSPs und dem EJB-Container für Enterprise Java Beans. Daneben gibt es Serviceklassen für die Kommunikation und das interne Management.
Diese Eigenschaften bedingen, daß fast alle Application Server im Grunde gleich aufgebaut sind. Der WebSphere Application Server auf z/OS ist eine Ausnahme: Nach außen hin präsentiert er sich identisch mit seinen Verwandten der Linux-/Unix-Derivate sowie der Windows-Variante. Unter der Haube sieht die Implementierung aber anders aus, um dem hohen Qualities-of-Service-Anspruch auf dem Mainframe gerecht zu werden. Über die Container für Servlets und Enterprise Java Beans, die plattformübergreifend implementiert sind, integriert sich der Application Server auf dem Mainframe besonders tief in das Betriebssystem z/OS. Das bedeutet, daß hier bestimmte Funktionen wie beispielsweise Transaktionalität, die auf dezentralen Plattformen extra interne oder externe Software voraussetzen, vom Betriebssystem zur Verfügung gestellt werden.
Eine Reihe von Features eines WebSphere auf z/OS sind nicht auf anderen Plattformen implementiert, unter anderem:

- Das Multi-Region-Konzept baut bereits für eine Single-Server-Instanz einen Cluster von Transaktionsbearbeitern (Servanten) mit eigener JVM auf, die WebSphere intern alle Vorteile eines Clusters liefern. Neben einem automatischen Nachstarten im Fehlerfalle können auch beim Nichteinhalten der definierten Antwortzeiten zusätzliche Servanten automatisch aktiviert werden.
- Bei Verschlüsselungsalgorithmen kann auf eine aufwendige und ressourcenintensive Softwareverschlüsselung verzichtet werden, da die Krypto-Hardwarekarten im Mainframe verwendet werden können, die keine normalen Prozessorressourcen beanspruchen. Dies trifft für Standardverschlüsselung wie SSL, aber auch für komplexere Algorithmen zu.
- Aufbau eines echten Clusters über mehrere Systeme hinweg, wobei bei eingehenden Transaktionen von der Mainframe-Infrastruktur das beste Hardwaresystem zum Abarbeiten ermittelt wird.
- Lokale Anbindung von Datenbanken spart durch schlankere Protokolle Prozessor-Ressourcen und verringert die Laufzeit der Transaktionen.

Während alle Application Server auf anderen Plattformen im Grunde auf einem einzelnen Prozeß (einer JVM) aufbauen, besteht eine Application-Server-Instanz unter z/OS aus insgesamt mindestens zwei Adreßräumen, also mehreren Prozessen: In der *Control Region* findet die Kommunikation zwischen dem Application Server und der Außenwelt statt, in der *Servant Region* wird die eigentliche Anwendung in einer oder mehreren JVMs ausgeführt. Müssen in der Anwendung auch Java-Messaging-Anfragen verarbeitet werden,

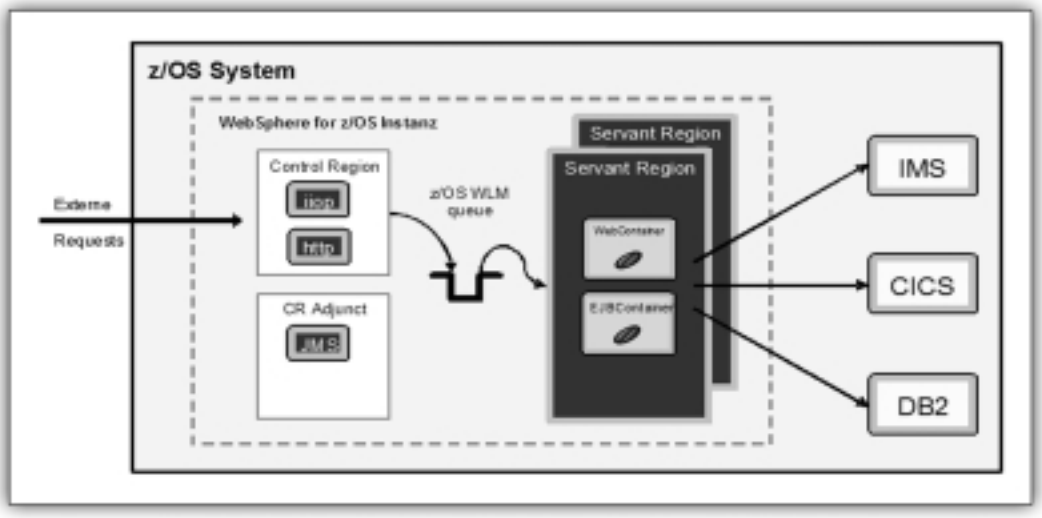


Bild 2: Adreßräume eines WebSphere Application Servers für z/OS

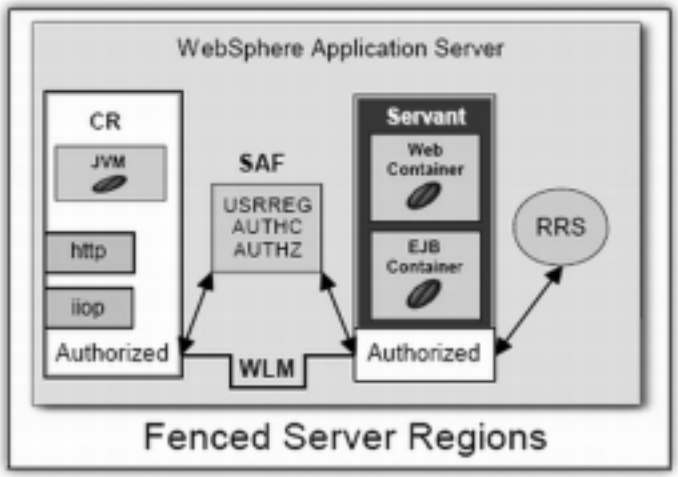


Bild 3: Fenced Server Regions

kommt es zum Start eines dritten Adreßraums, dem *Control-Region Adjunct*. Er repräsentiert die Implementierung der Messaging Engine im Application Server.

Die Aufteilung in einen Controller und mehrere »Arbeiter« gewährleistet, daß selbst bei einem Fehler während der transaktionellen Bearbeitung oder dem Hacken einer Applikation der gesamte Server weiterlaufen kann, da in der Regel nur einer der Servanten betroffen ist und die anderen weiterarbeiten.

Für effiziente Skalierung, Workload-Balancing, Verfügbarkeit und Failover nutzt der WebSphere Application Server auf z/OS eine Kombination aus Mainframe-Hard- und -software. Neben verschiedenen mainframe-proprietären Techniken zum Verwalten von Partitionen und Clustern ist hier insbesondere der z/OS Workload Manager zu erwähnen. Seine Kern-Funktionen sind Routing, Queuing, Workload-Priorisierung und das Einhalten von vorgegebenen Performancezielen.

WebSphere generell unterstützt das Two-Phase-Commit. Auf z/OS wurde diese Technik erweitert, so daß die Kommunikation nicht über den gesamten TCP/IP-Stack erfolgen muß, sondern nur über den Hauptspeicher. Dabei werden alle auf dem Mainframe gängigen DBMS wie IMS, CICS und DB2 unterstützt.

Dieses ganz wesentliche Unterscheidungsmerkmal zwischen den Application Servern wird besonders bei extrem hoher Transaktionslast beim Zugriff auf Daten, die im DB2-Datenbank-Subsystem auf z/OS gespeichert sind, deutlich, zum Beispiel bei Zahlungsverkehr-Anwendungen im Bankenumfeld. Dies kann eine Ersparnis der CPU-Zeit pro Transaktion von zirka siebenundsiebzig Prozent und eine Reduktion der Datenmenge von zirka neunundneunzig Prozent bedeuten. Bei einer hypothetischen Last von geforderten hundert Transaktionen pro Sekunde über einen längeren Zeitraum, was bei Großbanken häufig vorkommt, stehen also 4223 CPU-Sekunden den 950 CPU-Sekunden oder 19584 MByte nur 180 MByte transferierten Daten gegenüber.

WebSphere integriert sich tief in die Sicherheitsmechanismen der Mainframe-Plattform. Der z/OS Application Server kann die z/OS System Authorization Facility (SAF) als Registry für Authentifizierung und Authorisierung nutzen, für den Schutz der Runtime ist es sogar unumgänglich. Dabei müssen die Server autorisiert werden, um auf bestimmte z/OS-Systemressourcen zugreifen zu können.

Java macht also auch auf dem Mainframe eine gute Figur. Die Geschwindigkeitsprobleme aus den Anfangszeiten sind beseitigt, neue Techniken des Mainframe-Javas werden auch in andere Versionen der JVM eingebaut. Java auf dem Mainframe, ob nativ oder in einem Application Server Environment, ist auch in kritischen Umgebungen produktionsreif. ◆

Literatur

IBM z/OS Java Homepage: *http://www-03.ibm.com/servers/eserver/zseries/software/java/*.

Redbooks: Java Stand-alone Applications on z/OS, Volume I: *http://www.redbooks.ibm.com/abstracts/sg247177.html*

Java Stand-alone Applications on z/OS, Volume II: *http://www.redbooks.ibm.com/abstracts/sg247291.html?Open*

WebSphere for z/OS V6 Connectivity Handbook: http://www.redbooks.ibm.com/abstracts/sg247064.html?Open

Architecting High Availability Using WebSphere V6 on z/OS: *http://www.redbooks.ibm.com/abstracts/sg246850.html?Open*

WebSphere Application Server on z/OS and Security Integration: *http://www.redbooks.ibm.com/abstracts/redp4161.html?Open*

IBM WebSphere Application Server z/OS Homepage: *http://www-306.ibm.com/software/webservers/appserv/zos_os390/*

White Papers: Building a Quick and Simple WebSphere for z/OS V6 Runtime Environment: *http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100570*

WebSphere z/OS V6 – WSC Sample ND Configuration: *http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100653*

WebSphere Application Server for z/OS – Planning for Test, Production and Maintenance: *http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100396*

Der Weg in die Moderne

VON FRED STEFAN, BENJAMIN STORZ UND PAUL HERRMANN

In vielen Firmen sind Anwendungen im täglichen Einsatz, die bereits vor vielen Jahren entwickelt wurden und auf großen Transaktionsservern laufen. Um den modernen Geschäftsanforderungen schnell und flexibel gerecht werden zu können, müssen diese Anwendungen erweitert und angepaßt werden.

Das Ziel der meisten Firmen ist bei der Modernisierung der bestehenden Prozesse nicht die Neuentwicklung oder Umgestaltung, sondern deren Anpassung. Dafür werden Werkzeuge benötigt. Sie sollten dabei helfen, die vorhandenen Anwendungen weiterzuverwenden und mit anderen Funktionen zu einem Prozeß zu integrieren, der später auch Bestandteil einer service-orientierten Architektur (SOA) werden kann. Die unter großem Aufwand in der Vergangenheit entwickelten qualitativ hochwertigen Funktionen sind oft unflexibel und unter Mühen in eine SOA-Architektur zu integrieren. Sie wurden meist mit viel Redundanz rein nach praktischen Gesichtpunkten implementiert.

Auf eine saubere Trennung von Präsentations-, Geschäfts- und Datenbank-Logik, die die modernen Paradigmen der Informatik fordern, wurde kein Wert gelegt.

Hinzu kommt, daß solche Anwendungen aufgrund der 3270-Darstellung keine beziehungsweise nur eingeschränkte GUI-Funktionalitäten aufweisen. Sie bieten darüber hinaus keine Möglichkeit, die verschiedenen CICS-Transaktionen automatisch und parallel aufzurufen und zu nutzen. Das Problem ist, daß nie ein Datenaustausch der Transaktionen vorgesehen wurde.

Modernisierung mit dem SFF

Für die Lösung dieser Probleme sind drei Ansätze denkbar:

- *Modernisierung der Endbenutzer-Oberfläche*: Anhand neuer Technologien wie Java Server Pages oder HATS wird die eingeschränkte BMS-Präsentationslogik ersetzt und mit neuen Funktionalitäten versehen. Auf diese Weise können hostterminal-basierende Anwendungen ohne Änderungen browserfähig gemacht werden.
- *Modernisierung der Anwendungsarchitektur*: Mit Java und CICS lassen sich bestehende Anwendungen schrittweise modernisieren. Bei diesem Prozeß werden Anwendungskomponenten (wie beispielsweise CICS-COBOL-Anwendungen) sukzessive durch Java-Komponenten ersetzt.
- *Modernisierung der Anwendungskommunikation*: Eine solche Vorgehensweise hat den Vorteil, daß bereits vorhandene Anwendungskomponenten weiterverwendet werden können, während gleichzeitig neue Funktionalitäten integriert werden. Eine Realisierung dieses Ansatzes ist das CICS Service Flow Feature (SFF) der eclipse-basierten Entwicklungsumgebung WebSphere Developer for System z V7.0 (kurz: WDz). Es ermöglicht die Aggregation mehrerer CICS-Anwendungen zu einem Geschäftsprozeß, der dann hochgradig für den CICS Transaction Server optimiert ist. Das Service Flow Feature bietet dabei die Möglichkeit, eine Reihe von Interaktionen zwischen den CICS-Anwendungen zu automatisieren.

Letztere Anwendungsmodernisierung wird nachfolgend näher beschrieben. Das CICS Service Flow Feature ist ein zusätzliches Feature

von CICS TS seit Version 3 und in zwei Teile gegliedert: den Service Flow Modeler (SFM) und die Service Flow Runtime (SFR).
Die Service Flow Runtime stellt diverse Adapter als Erweiterungen der CICS-TS-Umgebung bereit, um die generierten Geschäftsprozesse später ausführen zu können, wobei der Service Flow Modeler den Websphere Developer for System z (der im WDz Studio Environment enthalten ist) mit Entwicklungswerkzeugen ergänzt, die für die Geschäftsprozeßentwicklung benötigt werden.
Das CICS Service Flow Feature bietet eine integrierte grafische Entwicklungsumgebung an, um aus verschiedensten CICS-Anwendungen einen Flow zu erstellen, dazu einen Generator, der eine CICS-TS-optimierte Laufzeitkomponente aus den aggregierten CICS-Anwendungen erstellt, und einen Adapter, der die CICS-TS-Umgebung so erweitert, daß die erzeugte Laufzeitkomponente ausgeführt werden kann.
Während der gesamten Entwicklung mit dem Service Flow Modeler benötigt der Entwickler kein zusätzliches Wissen über die Implementation der CICS-Anwendungen. Um die Entwicklungsarbeit zu erleichtern, können Commarea-Interfaces und 3270-Bildschirmdefinitionen direkt in den Service Flow Modeler importiert werden und als Erweiterung dienen. Gleichzeitig stellen SFR-Adapter den Zugriff auf die CICS-Anwendungen sicher. Aufgrund dieser nicht-invasi-ven Methoden sind keine Änderungen der CICS-Applikationen nötig.
Nun zu den nötigen Schritten, um aus Legacy-CICS-Anwendungen einen SOA-fähigen Geschäftsprozeß zu modellieren. Der Service Flow Modeler ist eine Zusammenstellung

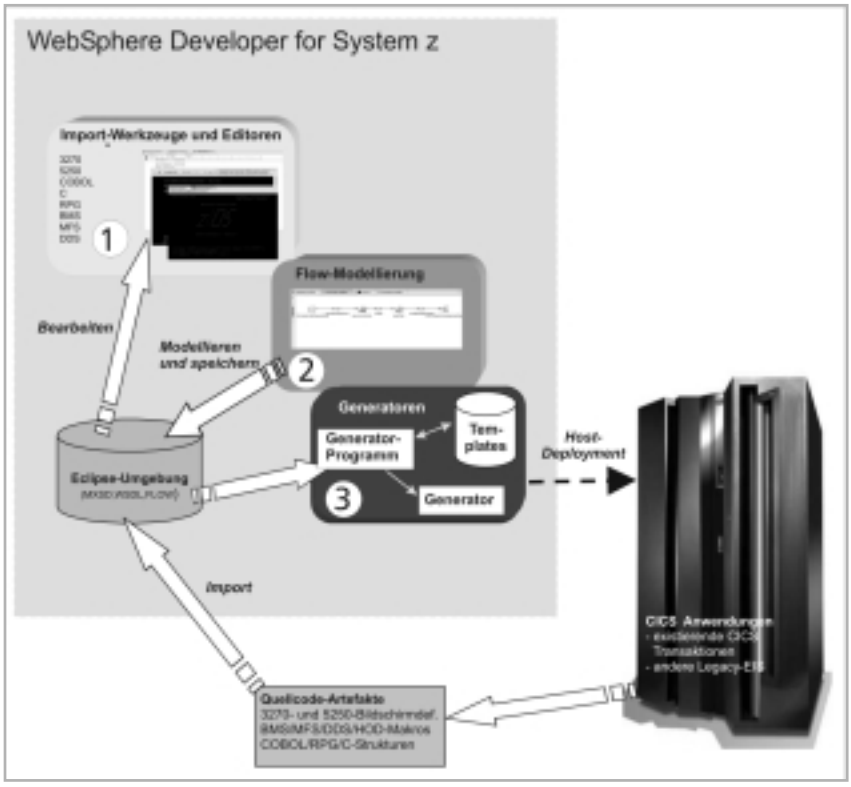


Bild 1: Schritte bei der Entwicklung eines Flows

von Werkzeugen in der Entwicklungsumgebung WDz für die Aggregation neuer Geschäftsprozesse und Flüsse aus bestehenden Anwendungen und deren Interfaces. Sie unterstützen die Programmierung von Bildschirmmasken und stellen SOA-konforme Schnittstellen bereit. WDz ermöglicht die Daten-Transformation zwischen den einzelnen CICS-Anwendungen, wobei die neuen Geschäftsprozesse zum Schluß als Webservice bereitgestellt werden.
Am Ende des Modellierungsprozesses wird anhand von SFF-Generatoren der Cobol-Quelltext erzeugt, der das Verhalten des modellierten Flows abbildet. Abschließend werden die Laufzeit-Eigenschaften für die jeweilige CICS-Konfiguration definiert, alle JCLs für das Deployment erzeugt und eine ausführbare WSDL-Datei des Geschäftsprozesses generiert. Für den Entwicklungsprozeß des Flows im Service Flow Modeler gibt es verschiedene Werkzeuge zum Importieren und Editieren von CICS-Anwendungen und für das Erzeugen der Laufzeitkomponenten. Der Ablauf ist folgendermaßen:

- Importieren und Editieren vorhandener Transaktionen anhand der integrierten Import-Werkzeuge und Editoren.
- Modellierung des Service Flows und weiterer Änderungen (z.B. Variablen-Mappings).
- Anhand von Template-Dateien (bei der SFR-Installation initial angelegt) erstellen Generatoren die für die Laufzeit benötigten Komponenten.

Diesen Ablauf verdeutlicht auch Bild 1.

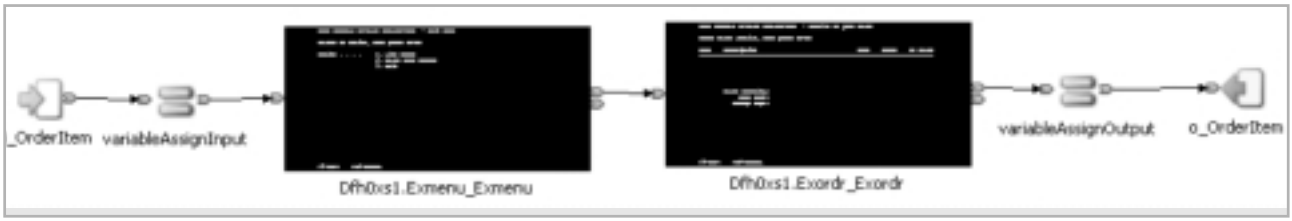


Bild 2: Service Flow, bei dem mehrere CICS-Bildschirme durchlaufen werden

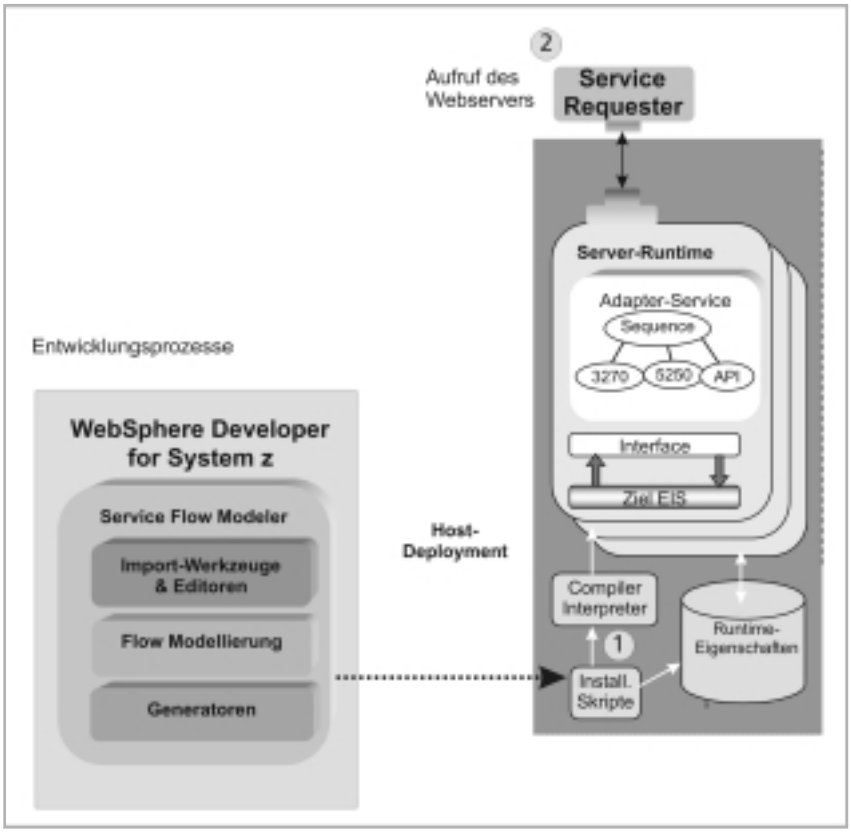


Bild 3: Die Verarbeitung auf dem Host

Ein Beispiel eines Flows zeigt Bild 2. Die Pfeile zwischen den einzelnen Symbolen zeigen die Durchlaufrichtung des Flows. Durch einfaches Hinzufügen oder Entfernen von Komponenten beziehungsweise Pfeilen läßt sich die gesamte Logik des Geschäftsprozesses umgestalten.

Service Flow Runtime

Nach der Modellierung bilden die Generatoren eine Laufzeitkomponente, die auf CICS-spezifische Möglichkeiten wie beispielsweise CICS Business Transaction Services und die Link3270-Bridge zurückgreifen. Erstere sorgen für eine hochoptimierte Implementierung in der CICS-Umgebung, die die inhärente QoS, die von der integrierten CICS-Anwendung bereitgestellt wurden, sicherstellt. Sobald die Laufzeitkomponente erzeugt wurde, wird sie vom SFM zum Host übertragen und kann von nun an verwendet werden.
Der CICS TS wird mit der SFR-Komponente des CICS SFF erweitert, die die für die Orchestrierung von Service Flows benötigen Komponenten und Konfigurationen bereitstellt. Weiterhin stellt die SFR die Adapter für den Zugriff auf CICS-Transaktionen

und Schnittstellen von Legacy-Anwendungen zur Verfügung. Da bei der Geschäftsprozeßmodellierung nicht-invasive Techniken zum Einsatz kommen, muß der Quelltext der zugrundeliegenden Anwendungen nicht verändert werden, womit eine schnelle und effiziente Wiederverwendung gewährleistet ist. In Bild 3 ist der Ablauf in der SFR nach dem Deployment eines Webservices dargestellt. Dabei installieren die vom SFM generierten JCLs den neuen Webservice im CICS. Nach erfolgreicher Installation kann der Webservice über einen Service Requester benutzt werden.
Jedesmal, wenn ein Webservice veröffentlicht wird, müssen dem CICS einige Parameter über die zu installierende Anwendung mitgeteilt werden. Dies sind beispielsweise die benötigten Ressourcen oder eventuelle Interaktionen zwischen verschiedenen Ressourcen. Aus diesem Grund muß der Webservice in einer Reihe von Schritten lauffähig gemacht werden. Allerdings wird dem Anwender diese Arbeit von den vom SFM automatisch erzeugten JCLs abgenommen. Wurde ein Webservice erfolgreich erzeugt und auf den Host exportiert, muß er noch manuell kompiliert werden, wozu nur ein einziger Befehl nötig ist. Anschließend ist der Webservice einsatzfähig.

Zusammenfassung

Der Service Flow bietet einen interessanten neuen Ansatz zur Modernisierung von Legacy CICS-Anwendungen, weil bereits laufende Anwendungen weiterverwendet und in vorhandene Architekturen integriert werden können. Zudem werden bessere Modernisierungsmöglichkeiten bereitgestellt als lediglich die Anpassung der Oberfläche. Hinzu kommt, daß mehrere CICS-Anwendungen aggregiert werden können und die damit erzeugten Geschäftsprozesse über offene Schnittstellen erreichbar sind.
Gleichzeitig findet eine Entkopplung der Präsentationslogik von der Anwendungslogik statt. ◆

Schon gelesen?

Testen Sie die freeX mit dem Miniabo!

- Kompetenz in Linux, BSD und Solaris
- Administration, Anwendung und Programmierung
- Systemsicherheit, Netzwerk und Datenbanken
- Hardware und Peripherie

Drei aktuelle Ausgaben freeX zum Preis von zwei: nur 23,80 Euro!

Ab jetzt immer mit DVD!

freeX ist eine Publikation des Computer- und Literaturverlag **www.cul.de**

Per Fax an: 0 89 / 20 02 81-15

oder per Post an unten stehende Adresse,

oder formlos per E-Mail auf www.cul.de

Antwort

Computer & Literaturverlag
freeX Abo-Service CSJ
Postfach 140 220

D-80452 München

Alle 2 Monate neu im Bahnhofsbuchhandel und am Kiosk mit diesem Zeichen:

PRESSE

Ja, ich bestelle 3 Ausgaben der Zeitschrift **freeX** im Mini-Abo zum Sonderpreis von nur 23,80 € frei Haus.
Ich war in den letzten 12 Monaten kein freeX-Abonnent.

Name, Vorname: _____

Straße, Nr.: _____

PLZ, Ort: _____

Datum, Unterschrift: X _____

Sollte sich meine Adresse ändern, erlaube ich der Deutschen Bundespost, meine neue Adresse dem Verlag mitzuteilen.

Ich wünsche folgende Zahlungsweise (wie angekreuzt):

❑ Bequem und bargeldlos durch Bankabbuchung:

Kontonummer: _____ Bankleitzahl: _____

Geldinstitut: _____

❑ Durch Überweisung nach Erhalt der Rechnung.

Die Datenbank des Host

Isabel Arnold,
Manuel Müller,
Christian Daser und
Andreas Gröschl

DB2 für z/OS feiert in diesem Jahr den 25. Geburtstag, IMS kann bereits auf 39 Jahre zurückblicken. Native XML-Unterstützung, Webservice-Funktionalitäten und Datawarehousing sind nur einige hochaktuelle Features der neuesten Releases. Die Entwicklungsgeschichte der Datenbanksysteme auf dem Host ist eng mit Transaktionssystemen wie den CICS Transaction Server verknüpft.

Seit dem 16. März 2007 ist DB2 für z/OS in der Version 9 mit vielen innovativen Features verfügbar. Die Entwicklung findet hauptsächlich im Silicon Valley in Kalifornien statt. Seit zirka zweieinhalb Jahren sind zudem Teile der Entwicklung nach Deutschland in das Böblinger IBM-Labor gewandert.

Auch wenn man sich dessen wahrscheinlich gar nicht bewußt ist, wikkelt man im täglichen Leben zahlreiche Transaktionen ab, zum Beispiel den Kauf eines Tickets oder die Bezahlung eines Medikamentes. Handelt es sich um abgeschlossene Geschäftsvorgänge, spricht man auch von Geschäftstransaktionen, die sich durch folgende Merkmale auszeichnen: viele Benutzer, Wiederholungsvorgänge, kurze Interaktionszeiten, gemeinsame Daten und niedrige Kosten pro Transaktion.

Systeme, die solche Geschäftstransaktionen implementieren, arbeiten alle nach dem sogenannten ACID-Prinzip, dazu später mehr.

Um zu verstehen, weshalb Transaktionssysteme entstanden sind, muß man in die Vergangenheit blicken. Bis Anfang der siebziger Jahre wurden Transaktionen ohne Benutzerinteraktion in Stapelverarbeitung (dem Batch-Modus) auf dem Großrechner durchgeführt. Vor allem wurden auf diese Weise größere Datenmengen verarbeitet wie beispielsweise eine wöchentliche Datensicherung oder der Monatsabschluß. Diese Batch-Jobs wurden in der Job Control Language (JCL) geschrieben und vom Job Entry System (JES) des Betriebssystems interpretiert (vergleichbar mit Unix-Shellskripten).

Online-Transaktionsverarbeitung

Diese Jobs wurden meist nach Zeitplan ausgeführt, wobei sie typischerweise eine Eingabedatei einlasen, Daten lasen und bearbeiteten und Berichte erstellten. Genau an dieser Stelle zeigten sich die Schwächen des Systems, die manchmal zu stundenlangen Wartezeiten auf die Ergebnisse führten. Im Extremfall mußte beispielsweise eine Bestellung verzögert werden, da zum Zeitpunkt der Eingabe das gewünschte Produkt bereits nicht mehr auf Lager war. Als Konsequenz forderten die Benutzer aktuelle Informationen, auf die man in Sekunden zugreifen konnte: die Online- oder auch Echtzeitverarbeitung.

Da Batch-Jobs in einem geplanten Zeitfenster ausgeführt werden, haben sie normalerweise exklusiven Zugriff auf alle Dateien, die sie verändern müssen. Online-Transaktionen müssen dagegen Ressourcen sperren und freigeben können, um konkurrierende Änderungen zu verhindern. Weitere Anforderungen sind zum Beispiel Benutzerberechtigungen, Behandlung von Ausnahmebedingungen, parallele Verarbeitung von Transaktionen oder auch das Anlegen einer Präsentationsschicht. Jede Transaktion hätte das alles selbst implementieren müssen, was die Programmierung komplex und das Programm schwer wartbar gemacht hätte. Um den Entwickler von Systemfunktionen zu entlasten, wurden Transaktionssysteme aus der Taufe geho-

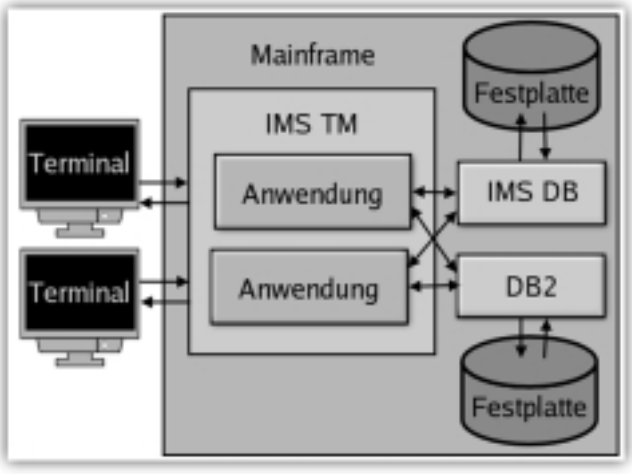


Bild 1: Client/ Server-Architektur in IMS

ben. Zu den bekanntesten gehören CICS (Customer Information Control System) und IMS (Information Management System).
Während CICS ursprünglich wegen seiner Flexibilität und Anpassungsfähigkeit eher für kleinere Maschinen entwickelt wurde, fand IMS dank des Fokus auf Geschwindigkeit und Zuverlässigkeit viele große Kunden, meist Banken.
Online-Verarbeitung erfordert die direkte Interaktion mit dem Benutzer. Aus diesem Grund wurde Anfang der 70er Jahre das Schreibmaschinenterminal mit Endlospapier als Ausgabemedium vom Bildschirmterminal Typ 3270 abgelöst. Dieser Bildschirm ließ sich programmieren und mit Ein- und Ausgabefeldern belegen. Erleichterungen waren bei IMS der Message Format Service, bei CICS der Basic Mapping Support (BMS) und bei TSO/ISPF die Panel-Definitionen.

Information Management System – IMS

Hinter dem biederen Namen IMS (Information Management System) steht eine der ersten, auf hierarchischen Strukturen basierenden Datenbanken zur extrem schnellen und sicheren Speicherung von Daten. Die Entwicklung einer hierarchischen Datenbank begann 1965 im Zusammenhang mit dem US-amerikanischen Apollo-Raumfahrtprogramm, 1968 wurde mit IMS/360 die erste Version ausgeliefert. Bis in die neunziger Jahre war IMS das wohl bedeutendste Datenbanksystem. Obwohl heute relationale oder objektrelationale Datenbanken der Standard für neue Anwendungen sind, bleibt IMS in der Verwaltung von hierarchischen Daten die wichtigste Größe. Neben den relationalen Systemen hat IMS als hierarchisches System einen festen Platz in der Verwaltung von geschäftskritischen Daten in Banken, der öffentlichen Verwaltung, der Industrie und im Einzelhandel.
Die zwei Komponenten von IMS, IMS-Datenbank (IMS-DB) und IMS-Transaktions-Manager (IMS-TM), folgen dem Konzept der Client/Server-Architektur: Ein Client schickt Anfragen an IMS-TM, der als Server fungiert. Dort läuft eine Anwendung, die auf die in IMS-DB gespeicherten Daten zugreift und eine Antwort erstellt.

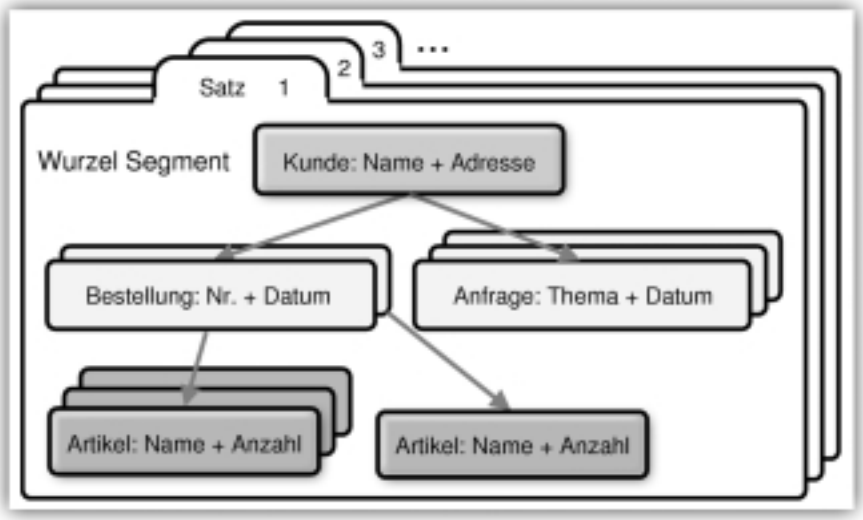


Bild 2: Struktur eines IMS-Datensatzes

Der IMS-Transaktions-Manager erhält Nachrichten beispielsweise von einem Webservice-Client, einem Webclient (über die Zwischenstation eines Webservers) oder etwa direkt von einem Geldautomaten.
Transaktionen – etwa eine Barentnahme an einem Geldautomaten – werden von in IMS-TM laufenden Anwendungen bearbeitet, die typischerweise Abfragen auf die IMS-DB durchführen.
IMS-TM kommuniziert asynchron. Die eingehenden Nachrichten werden in eine Queue (Warteschlange) gestellt und der nächsten verfügbaren Anwendung, die den jeweiligen Nachrichtentyp verarbeitet kann, zur Bearbeitung übergeben. Die dabei entstehende Verzögerung liegt typischerweise zwischen einigen Dutzend und einigen Hundert Millisekunden. Die Antwortnachricht wird wiederum in eine Queue gestellt. Im Beispiel des Geldautomaten wartet der Client aber in einem blockierenden Zustand. Deshalb wird die Antwortnachricht sofort vom Client abgeholt, wodurch sich die Kommunikation quasi-synchron darstellt.
Ist der Client ein Bildschirm-Terminal, kommt zur Formatierung der eingehenden und ausgehenden Nachrichten oft der Message Formating Service (MFS) zum Einsatz, der in die Nachrichten Formatattribute wie Farbe und Position auf dem Bildschirm einfügt.
Geschrieben sind IMS-Anwendungen oft in COBOL oder, vor allem in Europa, in PL/I. Weitaus seltener sind Programme in Assembler und C. Jüngste Sprache im Bunde ist Java, die durch eine spezielle Kontrolle der JVMs durch IMS zu vergleichbarer Performance wie die übrigen Sprachen fähig ist.

Hierarchische Struktur der IMS-DB

Eine IMS-Datenbank wird in Datensätze unterteilt, die im Gegensatz zu relationalen Datenbanken eine Baumstruktur (B-Tree) aufweisen. Für bestimmte Anwendungsfälle bietet diese Datenhaltung die Möglichkeit, von Wurzel- oder Muttersegmenten sehr schnell zu den abhängigen Kindsegmenten zu navigieren. Diesen strukturellen Vorteil weist auch XML auf, die Strukturen von IMS-Datenbanken und XML-Dokumenten sind in weiten Bereichen identisch. Spezielle Anwendungen erlauben es, XML in neuen oder bereits bestehenden IMS-Datenbanken effizient zu speichern und mit der gewohnten Geschwindigkeit und Zuverlässigkeit von IMS wieder abzufragen.
Anfragen an die Datenbank erfolgen in der Sprache Data Language

One (DL/I). Wie zur Zeit ihrer Einführung in den sechziger Jahren üblich, hält sich DL/I an strenge Formatvorgaben. Beispielsweise ist vorgegeben, an welcher Spalte ein DLI-Befehl anfängt, wie viele Zeichen der Funktionsname hat, in welchen beiden Spalten der Operator (etwa EQal) zu stehen hat und in welcher Spalte der nächsten Zeile weitergeschrieben wird, falls ein Befehl mehr als achtzig Spalten braucht. Das ist ein Erbe der Programmierung von Rechnern mit Lochkarten.
Das folgende Beispiel zeigt den Datenbankzugriff mit DL/I. Der erste Zugriff einer Anwendung erfolgt meist über der Befehl *Get Unique (GU)*, der ein Segment mit vorgegebenen Eigenschaften zurückliefert. Die Eigenschaften werden durch ein mitgeschicktes Segment-Search-Argument (SSA) definiert.

```
GU   KUNDE    (KDNO  EQ69042)
     BESTELL  (DATE  >=20060901&DATE
<=20070304)
```

Dieser DL/I-Befehl findet die erste Bestellung (Segment *Bestell*) des Kunden mit der Kundennummer 69042, die nach dem 1. September 2006 und vor dem 4. März 2007 eingetragen wurde. Weitere Aufrufe (*Get Next, GN*) finden andere Bestellungen, die diesen Kriterien entsprechen. Der Befehl *Get Next in Parent* (GNP) verhält sich ähnlich, beschränkt sich aber auf die Kinder eines Wurzelelements, sucht also am Ende des Satzes nicht im nächsten Satz weiter.
Eine weitere Besonderheit von IMS ist der *Update Intent*. Wenn mit einer gewissen Wahrscheinlichkeit davon auszugehen ist, daß die Anwendung das nächste gelesene Segment überschreiben will, beantragt sie ein Lock auf dieses Segment. Die Wahrscheinlichkeit sollte bei mindestens zehn Prozent liegen, denn so hoch ist der zusätzliche Aufwand beim Lesen des Segments. Die Befehle dafür lauten *Get Hold Unique* (GHU) und entsprechend *GHN* und *GHNP*.
Mit einer gleich zu Anfang abgesetzten Folge von GN-Befehlen läßt sich eine Datenbank komplett auslesen. Ein solches Vorgehen nimmt allerdings einige Zeit in Anspruch und ist deshalb nicht ratsam. Hier zeigt sich die enge Verzahnung zwischen Anwendung und Datenbank. Ein hierarchisches Modell wird auf den jeweiligen Zweck maßgeschneidert. Je genauer das Datenbankdesign auf den Einsatzzweck abgestimmt ist und je weiter sich ein Anwendungs-Programmierer in dieses Design vertieft, desto mehr Stärken kann IMS ausspielen.
Für die Erhöhung der Geschwindigkeit lassen sich auch Indexe verwenden. Der Primary Index verhält sich identisch zu dem aus relationalen Datenbanken bekannten. Der Secondary Index stellt eine eigene Datenbank dar. Er besteht aus Daten der ursprünglichen Datenbank und bietet über Zeiger Zugriff auf tiefer liegende Segmente. Bei einer Adreßdatenbank könnten zum Beispiel alle Telefonnummern in einem Secondary Index verzeichnet werden. Wenn auf dem klingelnden Telefon eine Nummer erscheint, kann nun über diesen Index das Telefon-Segment des entsprechenden Kunden gefunden und dessen Daten (im Baum weiter oben) gelesen werden.
IMS-TM und IMS-DB sind auch einzeln von anderen Systemen nutzbar. So kann beispielsweise CICS auf die IMS-DB zugreifen. Aber seit mit DB2 die erste relationale Datenbank auf z/OS Einzug hielt, ist auch der Zugriff von IMS-TM-Anwendungen auf IMS-DB möglich und auch inzwischen durchaus üblich. Doch was macht gerade DB2 auf dem Host so besonders, denn DB2 läuft ja auf sehr vielen Plattformen?

DB2 für z/OS

Wer an der Universität eine Datenbankvorlesung genossen hat, den haben sicherlich auch die Codd'schen Regeln aus den 70er Jahren tangiert. Diese im sonnigen Kalifornien entwickelten Regeln stellen bis heute die Grundlage der relationalen Datenbanktechniken und des SQL-Sprachumfangs dar. Begonnen als ein IBM-Forschungsprojekt, führte es letztendlich 1983 zur Datenbank DB2, der zweiten Datenbankgeneration aus dem Hause IBM.
Vor DB2 wurden bereits Daten persistent unter Verwendung prärelationaler Systeme gespeichert. Die Datenbankkomponente von IMS war und ist hier weiterhin ein großer Player, der die ACID-Prinzipien (A: Atomicity, C: Consistency, I: Isolation, D: Durability) umsetzt. Ohne

Kennzahlen von IMS- und DB2-Installationen

Die weltweite Bedeutung von IMS:	
Transaktionen weltweit pro Tag	50 Milliarden
Gespeicherte Produktionsdaten weltweit	15 Petabyte (Millionen GByte)
Anzahl IMS-User pro Tag	200 Millionen
Geldtransfer einer großen Installation pro Tag	2 Billionen US-Dollar
Eine typische DB2-Installation:	
Anzahl Konten	30 Millionen
Transaktionen pro Tag	100 Millionen
SQLs pro Sek. (Spitzenzeit)	20000
Transaktionen < 1 Sek	99%
Anzahl Tabellen	200000

Tabelle 1: System z ist vorrangig in großen Unternehmen zu Hause

deren Einhaltung darf sich eine Datenbank nicht also solche bezeichnen.
Das relationale DB2 ist heute auf diversen Plattformen verfügbar, beispielsweise z/OS, VSE, i5/OS, Linux, Unix, Windows und sogar auf PDAs oder ähnlichen Kleinstgeräten. Die Philosophie der DB2-Familie geht klar dahin, daß sich der Anwendungsentwickler auf allen Plattformen zurechtfindet und auch keinen Unterschied bei der Sprachfunktionalität und beim Umgang spüren sollte. Natürlich gibt es auf den Kleinstgeräten (DB2 Everyplace) nicht die volle Unterstützung wie bei den große Geschwistern, die vorhandenen Funktionen sind jedoch zwischen den Plattformen portabel.
Viele Datenbankfunktionalitäten werden auf z/OS – anders als bei anderen Plattformen – direkt von der Hardware unterstützt. Beispiele dafür sind Unicode-Umsetzung, Datenkomprimierung, Kryptographie oder Plattenzugriffe. Für solche Aufgaben nutzt System z speziellen Mikrocode, Einschubkarten und Hardware-Instruktionen.
Die Hauptklientel von DB2 auf z/OS ist (nicht zuletzt aufgrund der Hardware) vor allem der Banken- und Versicherungssektor sowie der Großunternehmensbereich. Die meisten Bankentransaktionen, sei es nun ATM, Überweisungen oder Börsengeschäfte, laufen an irgend einer Stelle über DB2 auf z/OS.
Einen simplen Webshop alleine auf DB2 z/OS zu betreiben, wäre im ersten Ansatz der Versuch, mit Kanonen auf Spatzen zu schießen. Kommen aber zum initialen Webshop nach und nach weitere Projekte und erhöhen sich die Anforderungen an Verfügbarkeit und Antwortzeiten, beispielsweise durch Service Level Agreements (SLA), macht es Sinn, diesen Webshop auf einer Mainframedatenbank zu betreiben. Vor allem bei Unternehmen, die höchste Anforderungen an Sicherheit, Verfügbarkeit und Skalierbarkeit erheben, ist DB2 nicht wegzudenken.
Typischerweise sind in einer z/OS-Umgebung auch die DB2-Administrationsaufgaben auf unterschiedliche Personen verteilt. Ein Systemprogrammierer übernimmt Aufgaben wie das Konfigurieren der DB2-Systeme, das Einspielen neuer Releases oder das Planen von Backup und Recovery.

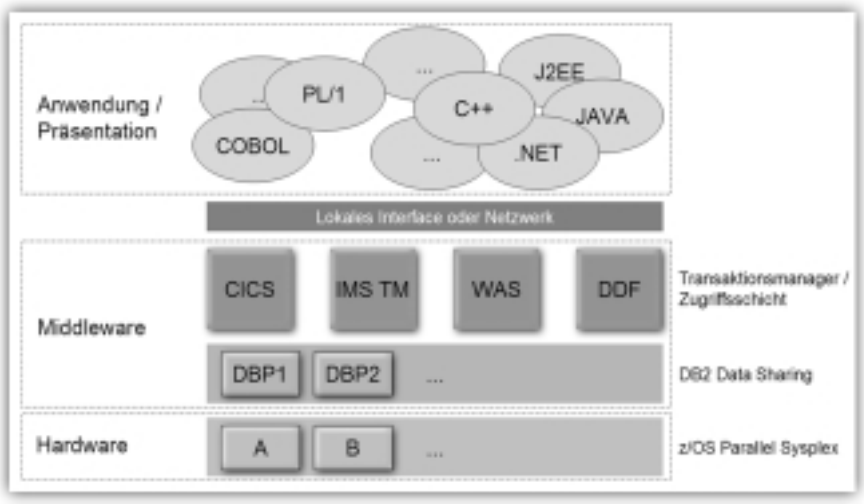


Bild 3: Eingliederung von DB2

Rollen im DB2

Ein Datenbankadministrator beschäftigt sich mit Aufgaben, die mehr in Richtung Anwendungsbetreuung gehen. Das heißt, sich Konzepte zu überlegen, wie eine Anwendung möglichst performant, sicher und nachhaltig auf der Datenbank zum Einsatz kommt. Typischerweise ist er ein Profi in Data Definition Language (DDL) und Data Manipulation Language (DML) als Unterformen von SQL und kennt sich mit Optimierungsmöglichkeiten wie beispielsweise Zugriffspfadanalyse aus.
All diese Tätigkeiten sind im z/OS-Umfeld hochautomatisiert und aufgrund der langjährigen Erfahrung auf dem Mainframe perfekt organisiert. Da sich in einer DB2 für z/OS unternehmenskritischsten Daten mit höchsten Anforderungen an Security und Verfügbarkeit befinden, sind solche Konzepte notwendig und überlebenswichtig für das Unternehmen.
Ein spezielles Feature für DB2 für z/OS ist das DataSharing. Dieses Konzept nutzt die Hardwarefunktionalität des System z aus, um eine optimale Verfügbarkeit zu gewährleisten. Hierbei fungiert ein Parallel Sysplex (Verteilung von Hardware gekoppelt über eine Coupling Facility) als Basis. Unterschiedliche DB2-Subsysteme treten nun als Member einer Data-Sharing-Gruppe auf. Dabei können einzelne Member über mehrere Kilometer verteilt positioniert werden. Anhand solcher Modelle sichern sich Unternehmen gegen katastrophale Szenarien wie beispielsweise den Verlust eines kompletten Rechenzentrums ab. Hierbei kann wieder ein Hardwarekonzept der System-z-Plattform (GDPS, Geographically Dispersed Parallel Sysplex) ausgenutzt werden.
Das Shared-Data-Konzept ist vollkommen transparent für Remote-Clients, die sich an eine DataSharing-Gruppe verbinden. Durch die mitgelieferten DB2-Treiber (JDBC, ODBC, und so weiter) kennt die Anwendung lediglich eine DataSharing-Gruppe. Der Workloadmanager von

Wie spricht man CICS eigentlich aus?

Während man in Deutschland häufig »Zicks« sagt, ist in England, Kanada, Australien oder Frankreich »Kicks« verbreiteter. In den Vereinigten Staaten wird CICS meist buchstabiert (»see eye see as«). Die Italiener nennen es »Chicks«, auf Spanisch heißt es »Thicks« und in Brasilien, Peru und Mexiko sagt man »Sicks«.

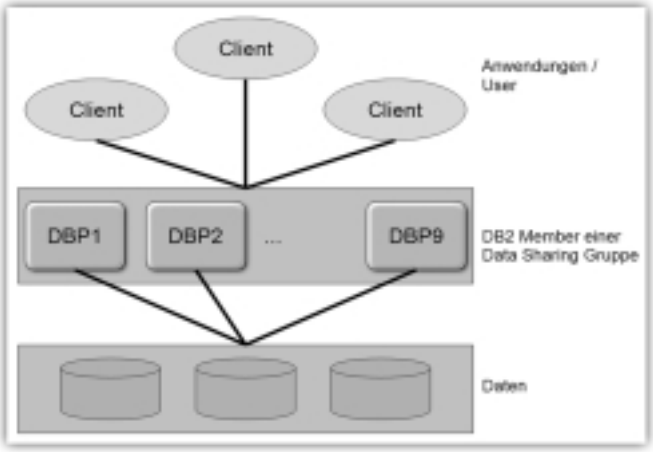


Bild 4: Eine Shared-Data-Umgebung

z/OS verteilt die Last auf unterschiedliche DB2-Member. Die Information, welcher Member wie stark ausgelastet ist, liegt ebenfalls am Client vor, der nun zu Transaktionsgrenzen (Rollback, Commit) die Anfrage auf ein jeweils günstiges Member routet. DataSharing hat weitere Vorteile in der Administration. Beispielsweise kann eine Rolled Migration durchgeführt werden. Dabei läuft ein Versionswechsel komplett unterbrechungsfrei ab. Nacheinander werden einzelne Member auf die neue Version gehoben, die komplette Data-Sharing-Gruppe bleibt jedoch die komplette Zeit online.

CICS

Doch noch einmal zurück in der Zeit. Neben dem monolithischen IMS mit integrierter Datenbank entstand der Transaktionsmonitor CICS.
Die Geschichte von CICS (Customer Information Control System) ist eng mit der System/360-Revolution verknüpft. In den Jahren 1964 bis 1967 verkaufte IBM viele der neuen Systeme. Wegen des Fachkräftemangels bei IBM und den Kunden für die Implementierung entwickelt IBM in dieser Zeit Anwendungsprogramme, die als kostenlose Beispiele weitergegeben werden konnten. Ben Riggins, der auch als der »Vater von CICS« bezeichnet wird, schlug vor, ein umfangreiches Beispielprogramm zu erstellen, das Terminals und Dateien lesen und schreiben und Transaktionen initiieren konnte. Nachdem CICS zunächst als kostenfreies Beispielprogramm verteilt wurde, fand es sich 1969 zusammen mit IMS und GIS (Generalized Information System) auf der Liste der ersten kostenpflichtigen IBM-Softwarelizenzen wieder. Nachdem CICS ursprünglich in den Vereinigten Staaten im IBM Labor Palo Alto aus der Taufe gehoben wurde, zog die Entwicklungsabteilung 1974 nach Hursley, Großbritannien, wo sie auch heute noch zu finden ist. In den 80er Jahren wurde das Command Level Interface für CICS entwickelt, das das Macro Level Interface ablöste. Eingebettete, sogenannte »EXEC CICS«-Kommandos ersetzten die Makros und vereinfachten durch ihre gleiche Syntax in allen Sprachen die Programmierung. Eine Datei wird beispielsweise mit

```
EXEC CICS READ FILE(filename) INTO(area)
...
END-EXEC
```

eingelesen. Um dem Standardcompiler die EXEC-CICS-Befehle verständlich zu machen, übersetzt ein separater Pretranslator diese Befehle zunächst in den äquivalenten Quelltext der jeweiligen Programmiersprache. Die einzige Ausnahme ist Java, das über die JCICS-Bibliothek auf Basis von objektorientierter Programmierung Zugriffe auf CICS-Services erhält und wegen der Java Virtual Machine keinen Pretranslator benötigt. Mittlerweile hat CICS den Beinamen »Transaction Server« erhalten und zahlreiche neue Funktionen hinzubekommen. CICS Transaction Server 3.1 unterstützt neben COBOL, PL/I und Assembler auch C, C++, ReXX und Java. Zahlreiche Wege in das System hinein und heraus wurden geöffnet, beispielsweise implementiert der CICS Transaction Gateway unter anderem die Java-Connector-Architektur, aus CICS-Programmen können Webservices erstellt werden und CICS hat in Version 2 sogar einen EJB-Container für Session-Beans dazubekommen. Neben den traditionellen Plattformen wie z/OS und z/VSE läuft CICS inzwischen auch auf AIX, Windows, Solaris und HP-UX, wo CICS TXSeries bis auf einige Einschränkungen die gleichen CICS-APIs zur Verfügung stellt.
Heute befassen sich geschätzte 950.000 Programmierer mit CICS. Das System hat ungefähr dreißig Millionen Benutzer bei etwa 20.000 Kunden weltweit, die ein tägliches Transaktionsvolumen von zirka dreißig Milliarden Transaktionen pro Tag erzeugen. Auch wenn der Großteil aller geschäftlichen Transaktionen dieser Welt in den vorgestellten Systemen vor sich gehen, hat sich doch in den letzten Jahren der Charakter von Geschäftsanwendungen auch im großen Maßstab gewandelt. Geschäftslogik wird in Frameworks und Container gekapselt und objektorientiert programmiert. Auf dem Host gibt es inzwischen viele Anwendungen in J2EE auf einem WebSphere Application Server. Es laufen Portal-Server, Laufzeitumgebungen für Geschäftsprozesse und Servicebusse für serviceorientierte Architekturen auf z/OS. Auch CICS und IMS bekamen EJB-Container. Java schickt sich an, neben den traditionellen Sprachen Cobol, Assembler und PL/I – die wohl unzweifelhaft nie aussterben werden – zur wichtigsten Sprache für geschäftskritische Anwendung auf z/OS zu werden. ◆

Quellen:

http://www.zois.co.uk/tpm/cics.html
http://www.yelavich.com/history/ev196803.htm
http://www-306.ibm.com/software/data/ims/v9/v9index.html

_INFRASTRUKTUR-PROTOKOLL

_Tag 82: Es gibt so viele Risiken da draußen. Spurrillen, Naturkatastrophen, Firmenfusionen. Wie schützt man sich davor? Jede dritte Firma erholt sich nicht mehr nach einer ungeplanten Ausfallzeit.[1] Wie wäre es bei uns?

_Till hat alles in Blisterfolie gepackt. Nur um auf der sicheren Seite zu sein.

_Tag 83: Ich schütze uns mit den IBM Business Resilience Solutions. IBM Business Continuity Services helfen uns, Risiken zu beurteilen und einen proaktiven Plan zu entwickeln, um mit ihnen umzugehen. IBM Tivoli ermöglicht, Infrastrukturprobleme zu erkennen und zu beheben. Und die Hochverfügbarkeitsfeatures des IBM System p sorgen für maximale Betriebszeit.

_Die Blisterfolie ist alle. Und ich muss ein Paket verschicken. Na klasse.

Tivoli

Nehmen Sie am „Business Continuity Assessment" teil unter:
IBM.COM/**TAKEBACKCONTROL**/READY/DE

[1] Quelle: „Business Continuity Unwrapped", Continuity Central, 2006, www.continuitycentral.com/feature0350.htm

GE&M IBM.07.2502