

IBMs Java-Implementierung für OS/390

Bohnen für den Mainframe



Uwe Zanker

Die OS/390-Java-Implementierung erlaubt objektorientierte Softwareentwicklung für Mainframes. Durch Konnektoren ist auch der Zugriff auf existierende Geschäftsprozesse und Unternehmensdaten möglich.

IMS, DB2, MQSeries, TSO und Batch unterstützt. Um einen mit Cobol oder PL1 vergleichbaren Transaktionsdurchsatz zu erreichen, ist geplant, die Performance der OS/390-JVM durch ein internes Redesign erheblich zu steigern.

‘Write-once, run anywhere’ entpuppt sich gerade im Hinblick auf teure Maschinen wie Mainframes als doppelter Vorteil. Zum einen kann man auf preisgünstigen PCs Code programmieren, der auch auf Großrechnern läuft. Zum anderen können Java-Programmierer Serveranwendungen entwickeln, ohne spezielle OS/390-Kenntnisse haben zu müssen.

Java-Komponenten komplett

Seit einigen Monaten ist die Java 2 Standard Edition (J2SE) Version 1.3 für OS/390 verfügbar. J2SE ist eine 100 % kompatible Implementierung der Java Virtual Machine (JVM) und beinhaltet unter anderem ein vollständiges Java Development Kit mit den bekannten Java-Tools:

- *javac*: Java-Compiler,
- *java*: Java-Interpreter (JVM),
- *jdb*: Java-Debugger,
- *javap*: Java-Disassembler,
- *javah*: Java-Headerfile-Generator (JNI),
- *javadoc*: Java Program Documentation Tools,
- *appletviewer*: Testtool für Java Applets,
- *jar*: Tool zum Erstellen, von Java-Archive-Dateien (Jar-Files),
- *javakey*: Tool zum Erzeugen digitaler Signaturen,
- *rmiregistry*: RMI-Registrierung.

Die OS/390-Java-Implementierung basiert auf den OS/390 Unix System Services. Sämtliche Java-Tools sind daher sowohl in der Unix-Shell als auch in der klassischen OS/390-Umgebung (TSO, Batch) einsetzbar.

OS/390, bekannt als klassisches Mainframe-Betriebssystem, entwickelt sich zunehmend zu einer leistungsfähigen E-Business-Plattform. Die Stärken von OS/390, wie Stabilität, Zuverlässigkeit, Skalierbarkeit und

Sicherheit, sind speziell in diesem Bereich, wo es auf die Rund-um-die-Uhr-Verfügbarkeit ankommt, von immenser Wichtigkeit. Mit Java für OS/390 steht auch die ‘Internet-Sprache’ für den IBM-Klassiker zur Verfügung.

Java soll als Standardprogrammiersprache für OS/390 eingeführt werden und mittelfristig die klassischen Programmiersprachen wie Cobol und PL1 ersetzen. Zukünftig wird Java von sämtlichen OS/390-Subsystemen wie CICS,

- Java für OS/390 ist IBMs Basis-Software-Technologie für zukünftige E-Business-Anwendungen.
- Servlets, Java Server Pages und Konnektoren ermöglichen den Zugriff auf existierende Geschäftsprozesse und Unternehmensdaten über WWW-Technik.
- Entscheidend für den Durchbruch von Java auf OS/390 ist die durchgängige Unterstützung der Java 2 Enterprise Edition (J2EE).

Eine Besonderheit von OS/390 ist, dass textbasierte Daten im EBCDIC-Format gespeichert sind. Java dagegen nutzt Unicode für die interne Darstellung von String- und Character-Daten (Dual-Bytecode). Bei der Verwendung von Javas Reader- und Writer-Klassen findet darum standardmäßig eine Datenkonvertierung von der Standard-Codepage des jeweiligen Betriebssystemes (IBM-1047 für OS/390 oder ISO8859-1 für Unix- und Intel-basierende Systeme) nach Unicode statt. Will man eine Datei im ASCII-Format verarbeiten, ist die Standard-Codepage bei den entsprechenden Reader- und Writer-Klassen zu überschreiben. Dasselbe gilt für eine TCP/IP-Socket-Kommunikation zu einer Java-Anwendung, die auf einem ASCII-basierenden Betriebssystem läuft.

Alternativ zur Softwareentwicklung auf dem PC kann ein Java-Programm komplett auf dem Mainframe entwickelt, übersetzt und natürlich ausgeführt werden; Letzteres in der OS/390-Unix-Shell oder als Batch-Job (siehe Abbildung 1).

Neben eigenständigen Anwendungen laufen auch Applets, Servlets und Java Server Pages (JSPs) unter OS/390; Applets nur mit dem Appletviewer, da für OS/390 kein Webbrowser verfügbar ist. Servlets sind ebenso wie JSPs standardisierte Webserverweiterungen und bieten die Möglichkeit, dynamisch Webinhalte zu erzeugen.

Servlets und JSPs benötigen zur Ausführung den WebSphere Application Server

(WAS), einen HTTP-Server kombiniert mit dem Application-Server, der eigentlichen Servlet Engine. Diese dient als Runtime-Umgebung für Servlets sowie JSPs und stellt verschiedene Basisdienste (Session Management et cetera) bereit. Um eine Lastverteilung und hohe Verfügbarkeit zu erzielen, kann WebSphere im 'Scalable Mode' betrieben werden. Unsere Erfahrungen haben gezeigt, dass ein unter Windows NT entwickeltes Servlet ohne Änderung auf dem OS/390 WebSphere Application Server ablauffähig ist.

Java-Programme mit grafischer Benutzeroberfläche benötigen für die Ausgabe einen über das Netz erreichbaren X11-Server, also ein X11-Terminal, eine Unix-Workstation oder einen Windows-PC mit PC-X11-Emulation. Die Adressierung erfolgt auf Unix-übliche Weise durch die Environment-Variable *DISPLAY*. Da die gesamte Dialogsteuerung (selbst das Bewegen des Mauszeigers) auf dem Mainframe erfolgt und dadurch ein ständiger Datenaustausch zwischen Mainframe und X-Server stattfindet, ist ein solches Anwendungsmodell aber nicht zu empfehlen.

Empfohlenes Anwendungsmodell

Als tragfähiges Anwendungsmodell für OS/390 eignet sich eine 'Multi-Tier'-Anwendungsarchitektur, wie sie der Standard Java 2 Enterprise Edition (J2EE) definiert (siehe Abbildung 2). Dieses Schichtenmodell untergliedert

sich in drei beziehungsweise vier Schichten (Präsentation, Verarbeitung, Integration, Datenzugriff).

Die Präsentation der Daten (Tier 1) erfolgt bei diesem Anwendungsmodell ausschließlich auf dem Client-Tier mittels Webbrowser (Applet, HTML) oder als Java-Anwendung. Im letzteren Falle erfolgt die gesamte Aufbereitung der grafischen Daten, die Dialogsteuerung und die Darstellung der Daten auf dem Client-Rechner und entlastet dadurch den Mainframe. Bei einer reinen HTML-Lösung hingegen werden die HTML-Daten auf dem Server mittels Servlets beziehungsweise JSPs aufbereitet. Der Webbrowser übernimmt lediglich die grafische Darstellung der Daten.

Das Middle-Tier (2 und 3) ist zuständig für die Verarbeitungs- beziehungsweise Integrationslogik in Form von Servlets, JSP oder Enterprise JavaBeans (EJB). Mit der

derzeitigen Version 3.0.2 von WebSphere werden jedoch lediglich Servlets (Spec. 2.1) und JSPs (Spec. 1.0) unterstützt. EJBs sind frühestens in einer nachfolgenden Version der WebSphere Enterprise Edition zu erwarten, vermutlich im ersten Halbjahr 2001. Datenbank und Transaktionsmonitor (CICS, IMS) befinden sich auf dem Backend-Tier (Tier 4).

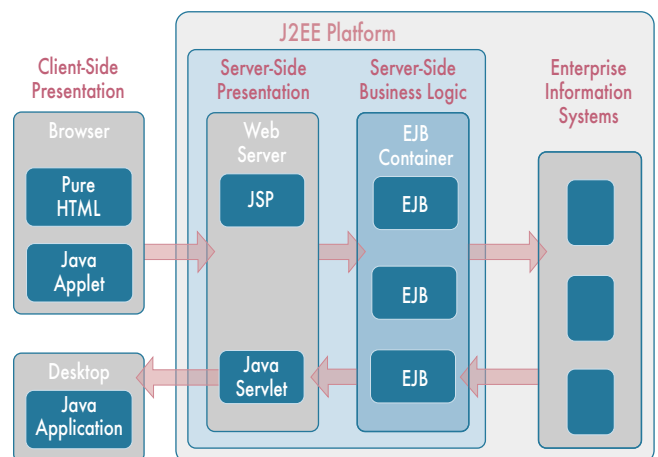
OS/390 ist dafür ausgelegt, unterschiedliche Workloads wie Batch, TSO, Webserver, Transaction Server, Application Server und Database Server konkurrierend zu betreiben, und bietet dadurch die einzigartige Möglichkeit, eine 'Multi-Tier Software Architecture' in einer '2-Tier Hardware Architecture' abzubilden. Es besteht keine Notwendigkeit, für jedes Software-Tier (Middle-/Backend-Tier) auch ein eigenes Hardware-Tier (Hardware-Box mit physischem Netzwerkanschluss) zu implementieren. Die Vor-

```

Tera Term - debis_Systemhaus_SYX1_OS/390 VT
File Edit Setup Control Window Help
</u/saz11/saz1109/java/hello>cat Hello.java
import java.io.*;

public class Hello
{
    public static void main(String args[])
    {
        System.out.println("Hello World");
    }
}
</u/saz11/saz1109/java/hello>javac Hello.java
</u/saz11/saz1109/java/hello>java Hello
Hello World
</u/saz11/saz1109/java/hello>
    
```

Kompilieren und Ausführen eines Java-Programmes unter OS/390 (Abb. 1)



Typische 'Multi-Tier'-Anwendungsarchitektur (Abb. 2)



OS/390-JAVA-KONNEKTOREN

Subsystem	Konnektor
DB2	JDBC, SQLJ
CICS	CICS Transaction Gateway for Java
IMS	IMS Connect for Java
MQSeries	MQSeries Java Client, MQSeries Bindings for Java
VSAM	Java Record I/O
OS/390	Data Sets, Java Record I/O
RACF	Java RACF classes
C/C++	Java Native Interface (JNI)
TCP/IP	Network Computing Interface (debris NCI)

teile einer solchen Lösung sind im Wesentlichen:

- einfacher und gesicherter Betrieb,
- durchgängiges Sicherheitskonzept vom Client bis zur Datenbank,
- bessere Skalierbarkeit,
- hohe Verfügbarkeit,
- bessere Performance (für die Kommunikation zwischen Software-Tiers ist kein physisches Netzwerk erforderlich).

Konnektoren: Brücke zu Altdaten

Java-Konnektoren dienen dem Zugriff auf existierende Geschäftsprozesse und Unternehmensdaten. Der Textkasten 'Verfügbare OS/390-Java-Konnektoren' listet die derzeit zur Verfügung stehenden auf: JDBC (dynamisches SQL) beziehungsweise SQLJ (statisches SQL) ermöglichen den Zugriff auf DB2-Daten.

Bei JDBC handelt es sich um einen von Sun spezifizierten offenen Standard für den Zugriff auf Datenbanksysteme (vergleichbar mit ODBC für C/C++). DB2 für OS/390 unterstützt JDBC-Driver vom Typ 1 (JDBC/ODBC Bridge) und Typ 2 (Native Driver). Da derzeit von IBM noch keine Netzwerktreiber vom Typ 3 oder Typ 4 verfügbar sind, kann ein JDBC-Zugriff über das Netzwerk momentan nur mittels DB2 Connect oder Lösungen von Drittanbietern wie Merant erfolgen. JDBC hat gegenüber SQLJ den Vor-

teil, dass keine Programmvorbereitung in Form einer Vorkompilierung erforderlich ist. Das Interpretieren der SQL-Anweisungen zur Programmaufzeit kann sich jedoch negativ auf die Gesamt-Performance auswirken.

SQLJ stellt sich aus Sicht der Programmierung einfacher dar, erfordert jedoch gegenüber JDBC eine entsprechende Programmvorbereitung in Form einer Vorkompilierung (ähnlich wie bei Cobol). Währenddessen erfolgt eine Syntax-, Typen- und Gültigkeitsprüfung von Schemata. SQLJ bietet den Vorteil, dass Benutzer zur Ausführung von bestimmten Programmen berechtigt werden können und nicht zwangsläufig eine Berechtigung für die gesamte DB2-Tabelle benötigen. Die Sicherheitsmechanismen lassen sich bei statischem SQL wesentlich granularer als bei dynamischem SQL anwenden.

Gateway für CICS und IMS

Eine typische Anwendungsarchitektur für den Zugriff auf OS/390-DB2-Daten kann wie folgt aussehen: Der Client (Webbrowser) sendet einen HTTP-Request, daraufhin startet der OS/390-Webserver (WebSphere) ein Servlet beziehungsweise JSP. Der Zugriff auf DB2 erfolgt auf Basis von JDBC oder SQLJ, das Ergebnis der Datenbankabfrage senden Serv-

let/JSP direkt an den Webbrowser zurück (als HTML, XML oder Text).

Das CICS Transaction Gateway for Java (CTG) bietet die Möglichkeit, aus einer Java-Umgebung direkt mit einer CICS-Transaktion (zum Beispiel Cobol) zu kommunizieren. Der Datenaustausch läuft entweder auf 3270-Basis (External Presentation Interface, EPI) oder besser mittels CICS-Commarea (External Call Interface, ECI). Bei der Verwendung von EPI können bestehende 3270-CICS-Transaktionen unverändert aufgerufen werden, allerdings mit starken Einschränkungen, da der gesamte Datenaustausch auf 3270-Bildschirmmasken basiert. EPI ermöglicht lediglich die Umsetzung von 3270-Screens in ein 'Java Graphical User Interface'. Der gravierende Nachteil dabei ist, dass nach jeder Änderung der 3270-Maske die Java-EPI-Anwendung entsprechend angepasst werden muss. Wesentlich mehr Flexibilität bietet ECI, weil hier der Datenaus-

tausch über CICS-Commarea erfolgt und man dadurch die Java-Anwendung unabhängig vom 3270-Datenformat gestalten kann.

CTG besteht im Wesentlichen aus einem Set von Java-Klassen und einem Daemon-Prozess. Unter Verwendung der CTG-Java-Klassen (ECI-Request/EPIRequest) kommuniziert die Java-Anwendung mit dem CTG-Daemon auf Basis von TCP/IP. Der CTG-Daemon-Prozess, der als TCP/IP-Listener fungiert, übernimmt die klassische Rolle eines Gateways und leitet die Anforderung an CICS weiter. Er kann auf einem Windows- oder Unix-System (als Middle-Tier) oder direkt auf dem OS/390-Mainframe betrieben werden. CTG für OS/390 unterscheidet sich dadurch, dass ECI nicht unterstützt ist und für den internen Zugriff auf die CICS-Commarea kein ECI, sondern das OS/390 spezifische 'External CICS Interface' (EXCI) zum Einsatz kommt. Die CICS-Transaktion liest die Eingabedaten aus

LISTING 1

```
public class ReadFile {
public static void main(String[] args)
{
String fileName = args[0];
int recordLen = Integer.parseInt(args[1]);
byte[] buffer = new byte[recordLen];
int bytesRead;

IRecordFile rf = RecordFile.getInstanceOf(fileName, recordLen,
Constants.JR10_FIXED_MODE);
IFileInputRecordStream firs = FileInputRecordStream.getInstanceOf(rf);

String encoding = System.getProperty("file.encoding");

while((bytesRead = firs.read(buffer)) != -1)
{
System.out.println(new String(buffer, 0, bytesRead, encoding));
}
}
}
```

Windows NT: Lesen der Datei *config.sys* mit einer maximalen Satzlänge von 500

```
java ReadFile c:\config.sys 500
```

Unix: Lesen der Datei: */etc/profile* mit einer maximalen Satzlänge von 500

```
java ReadFile /etc/profile 500
```

OS/390: Lesen des Members *IEASYS00* aus dem Partitioned Data Set *SYS1.PARMLIB* mit einer maximalen Satzlänge von 80

```
java ReadFile m/'SYS1.PARMLIB(IEASYS00)' 80
```

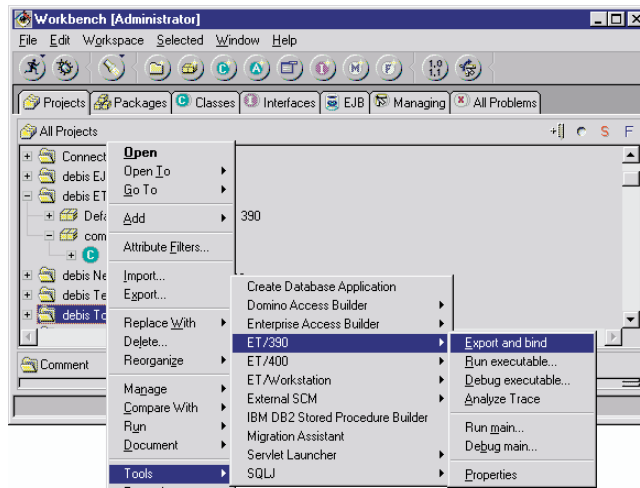
Ein Beispiel für Java-Record-I/O-Code (oben) und die entsprechenden Befehle auf den unterschiedlichen Betriebssystemen.



der Commarea. Nach erfolgter Verarbeitung stehen die Ergebnissdaten über die CICS-Commarea der Client-Anwendung zur Verfügung.

Ähnlich wie für CICS gibt es auch für das hierarchische Datenbanksystem IMS ein entsprechendes TCP/IP-Gateway. IMS TCP/IP OTMA Connection (ITOC, ab IMS Version 7: IMS Connect) bietet die Möglichkeit, aus einer Java-Umgebung direkt mit einer IMS-Transaktion zu kommunizieren. Der Datenaustausch erfolgt über die IMS Message Queue. ITOC besteht ebenfalls wie CTG aus einem Set von Java-Klassen und einem Daemon-Prozess. Der ITOC-Daemon-Prozess fungiert als TCP/IP-Listener und leitet die Anforderung an das entsprechende IMS auf Basis der 'Open Transaction Manager Architecture' (OTMA) weiter. Im Gegensatz zum CICS Transaction Gateway gibt es den ITOC-Daemon lediglich für die OS/390-Plattform.

Ein typisches Anwendungsszenario für TCP/IP OTMA Connection kann wie folgt aussehen: Eine Java-Anwendung verwendet die ITOC-Java-Klassen, um über TCP/IP mit dem ITOC-Daemon-Prozess zu kommunizieren.



Mainframe-Programmierung auf dem PC: Visual Age for Java Enterprise Toolkit 390 (Abb. 3).

ren. Der leitet die Anfrage mittels OTMA an das entsprechende IMS weiter. IMS startet daraufhin eine Transaktion, die die Eingabedaten aus der IMS Message Queue liest (GU IOPCB) und die Ergebnissdaten wiederum in der IMS Message Queue bereitstellt (ISRT IOPCB). Die Ergebnissdaten aus der IMS Message Queue stehen anschließend der Client-Anwendung zur Verfügung.

MQSeries ist eine Middleware für den asynchronen Austausch von Nachrichten. Die Nachrichten (Messages) werden in Warteschlangen (Queues) abgelegt und von

einem Queue-Manager verwaltet. Das Übertragen von Nachrichten findet zwischen den beteiligten Queue-Managern und nicht über eine Programm-zu-Programm-Kommunikation statt.

Die Java-Klassen von MQSeries unterstützen in der Regel zwei unterschiedliche Zugriffsarten: Binding Mode und Client Mode. Der lokale Zugriff (Binding Mode) auf einen Queue-Manager erfolgt mittels Java Native Interface (JNI). Lokal bedeutet in diesem Fall, dass die Java-Anwendung auf demselben Rechner wie der Queue-Manager laufen muss.

TCP/IP-basiert ist dagegen der Zugriff auf einen entfernten Queue-Manager (Client Mode). Die Java-Klassen für OS/390, die als so genanntes Support Pack (MA1G) frei erhältlich sind, unterstützen aber nur den lokalen Zugriff. Versuche bei debis haben jedoch gezeigt, dass auch das Support Pack für Windows NT (MA88) auf OS/390 genutzt werden kann, um über eine Client-Verbindung mit einem entfernten Queue-Manager zu kommunizieren. Warum der Client Mode derzeit nicht offiziell für OS/390 unterstützt wird, ist daher unverständlich.

Zugriff auf OS/390-Dateien

Der Zugriff auf das Unix-Dateisystem von OS/390 (Hierarchical File System) ist über das *java.io*-Package möglich, für den Zugriff auf klassische OS/390-Dateisysteme wie Sequential Data Sets, Partitioned Data Sets sowie VSAM sind Javas Record-I/O-Klassen (JRIO) erforderlich. JRIO ist Bestandteil von J2SE für OS/390.

Record-I/O-Klassen sind auch für Unix und Windows unter dem Namen Rioja erhältlich. Damit können selbst



Java-Anwendungen, die auf klassische OS/390-Dateisysteme zugreifen, unter Windows- oder Unix entwickelt und getestet werden.

Java Record I/O unterstützt folgende Funktionen:

- Zugriff auf Virtual Sequential Access Method (VSAM) Data Sets (nur KSDS),
- Zugriff auf Sequential (PS) und Partitioned Data Set (PDS),
- Anzeigen von Member eines Partitioned Data Set (PDS directory),
- Anzeigen von Data Set High Level Qualifiers (HLQ) aus dem OS/390-Systemkatalog.

Das nachfolgende Beispielprogramm zeigt, wie man mit Javas Record-I/O-Klassen sowohl OS/390-, Unix- als auch Windows-Dateien lesen kann. Der Inhalt einer Datei wird jeweils nach *stdout* ausgegeben.

Serversicherheit

Serveranwendungen sind im Sicherheitskonzept von Java erst einmal nicht vorgesehen. Funktionen wie Benutzerauthentifizierung und Prüfung von Zugriffsrechten fehlen. Die Java-Klassen für RACF beinhalten darum folgende Erweiterungen:

- Extrahieren der effektiven User-ID (RACF User-ID, unter dem die aktuelle Java-Anwendung läuft),
- Benutzerauthentifizierung (User-ID/Passwortprüfung),
- Ändern des aktuellen Passworts,
- Prüfung, ob eine User-ID einer bestimmten RACF-Gruppe zugehört,
- Prüfung, ob die effektive User-ID Zugriff auf bestimmte RACF-Ressourcen hat (Access Control),
- Ändern der effektiven User-ID (geplant).

Java Native Interface (JNI) beschreibt eine standardisierte Schnittstelle für die Interaktion von Java- und C/C++-Programmen [5, 6]. Dadurch können Java-Programme bestehende Anwen-

GROSSRECHNER- UND ANDERE AKRONYME			
CICS	Customer Information Control System	JNDI	Java Naming and Directory Services
CTG	CICS Transaction Gateway for Java	JNI	Java Native Interface
ECI	External Call Interface	JRIO	Java Record I/O
EPI	External Presentation Interface	KSDS	Keyed Sequential Data Set
ET/390	Visual Age Enterprise Toolkit 390	NCI	Network Computing Interface
EXCI	External CICS Interface	OTMA	Open Transaction Manager Architecture
HFS	Hierarchical File System	PDS	Partitioned Data Set
HLQ	High Level Qualifier	PS	Physical Sequential Data Set
HPJ	High Performance Java	RACF	Resource Access Control Facility
IIOB	Internet Inter ORB Protocol	RIOJA	Record I/O Java
IMS	Information Management System	RMI	Remote Method Invocation
ITOC	IMS TCP/IP OTMA Connection	SMB	Server Message Block
J2EE	Java 2 Enterprise Edition	SQJ	Structured Query Language for Java
J2SE	Java 2 Standard Edition	TSO	Time Sharing Option
JCICS	Java Transactions for CICS	VSAM	Virtual Sequential Access Method
JDBC	Java Database Connectivity	WAS	WebSphere Application Server
JIMS	Java Transactions for IMS	WLM	Workload Manager
JMS	Java Messaging Service		

dungslogik in Form von C/C++ wiederverwenden. JNI unterstützt zwar keine direkte Interaktion zwischen Java und klassischen Programmiersprachen (Cobol, PL1, Fortran, Assembler), über den Umweg C/C++ können jedoch auch diese Programme aufgerufen werden.

Da sich die Verwendung von JNI als äußerst umständlich und fehleranfällig gestaltet und zudem die Java-Portabilität verloren geht, sollte man dieses Interface nur nutzen, wenn sonst keine anderen Alternativen zur Verfügung stehen.

Eine weitere Möglichkeit, Java mit existierenden Anwendungen zu verbinden, bietet TCP/IP. Der Vorteil einer TCP/IP-Verbindung besteht darin, dass die Kommunikation auch über Rechengrenzen funktioniert und die Java-Portabilität nicht verloren geht. Dazu existieren Produkte von Drittanbietern.

Als optionales Produkt für OS/390 bietet der High Performance Java (HPJ) Compiler die Möglichkeit, aus Java-Programmen (*.class), die ja bekanntlich im Bytecode und damit in einem plattformneutralen Format vorliegen, OS/390-Binärcode zu erzeugen. Binärcode hat gegenüber Bytecode den Vorteil, dass

ein Programm direkt (ohne Java-Interpreter) ausgeführt werden kann. Nachteilig wirkt sich der Verlust der Plattformunabhängigkeit aus, des Weiteren müssen sämtliche referenzierten Java-Klassen als OS/390-Binärcode vorliegen. Laut IBM ist bei der Ausführung von OS/390-Binärcode im Vergleich zu Java-Bytecode nur ein geringfügiger Performance-Unterschied zu erkennen.

Binäres Java

Für die Programmierung von IMS-Transaktionen (JIMS) oder DB2 Stored Procedures in Java ist derzeit noch HPJ als Voraussetzung erforderlich. Diese Einschränkung, die bis vor kurzem auch für CICS-Transaktionen (JCICS) galt, wird jedoch in Zukunft aufgehoben. Da WebSphere HPJ ohnehin nicht unterstützt, dürfte dieses Produkt zukünftig an Bedeutung verlieren.

Die Enterprise Edition von Visual Age für Java (für Windows) beinhaltet das Enterprise Toolkit 390 (ET/390). Es ermöglicht, aus der Visual Age Workbench Java-Programme auf OS/390 auszuführen und bei Bedarf remote zu debuggen. Dazu muss der

Java-Code aus dem Visual Age Repository exportiert werden. Für den Export des Java-Codes in das OS/390-Filesystem (HFS) ist eine NFS- oder SMB-Verbindung erforderlich. Befindet sich der Java-Code im OS/390-HFS, können folgende ET/390-Funktionen genutzt werden:

- ‘Export and bind’ übersetzt den Java-Bytecode nach OS/390-Binärcode. ET/390 erzeugt dabei ein Unix-Shell-Skript, überträgt dies via FTP ins OS/390-HFS und führt es anschließend mittels ‘Remote Exec’ (REXEC) aus. Im Unix-Shell-Skript wird der High Performance Java (HPJ) Compiler aufgerufen, um den Java-Bytecode in OS/390-Binärcode zu wandeln.
- ‘Run main’ führt Java-Bytecode ohne HPJ aus.
- ‘Run executable’ führt Java-Binärcode aus.
- ‘Debug executable’ startet eine ‘Remote Debug Session’ zum Testen des Java-Binärcodes.
- ‘Analyze Trace’ dient der grafischen Auswertung der Trace-Daten. Tracing muss dazu aktiviert sein.
- ‘Debug main’ startet eine ‘Remote Debug Session’ zum Testen des Java-Bytecodes.

Da sich das gesamte Handling (Konfiguration, Export von Java-Code, Starten des Java-Programmes beziehungsweise der Debug-Session) als umständlich erweist und das Debuggen von Java Servlets und JSPs nicht unterstützt wird, ist es ratsam, den Code soweit möglich lokal unter Visual Age zu testen.

Ausblick

Für das erste Quartal 2001 ist eine überarbeitete JVM (1.3+) mit erheblichen Performance-Verbesserungen angekündigt. In der derzeitigen Version 1.3 verursacht das Auf- und Abbauen der JVM einen großen Overhead, der künftig durch ein so genanntes Master/Slave-Konzept nahezu eliminiert werden soll. Ist eine Master-JVM erstmals aufgebaut, lassen sich so genannte Slave-JVMs mit wenigen Instruktionen starten. Eine Slave-JVM kann auf bereits von der Master-JVM geladene Java-Klassen zugreifen.

Ebenfalls angekündigt ist die Unterstützung von Java für DB2 Stored Procedures und IMS (JIMS). Java kann derzeit für die Programmierung in einer CICS-, MQSeries-, TSO-, Batch- sowie OS/390-

Unix-Shell eingesetzt werden.

Entscheidend für den Durchbruch für Java auf OS/390 ist die durchgängige Unterstützung des Java-2-Enterprise-Edition-Standards (J2EE). Derzeit sind nur Teile davon wie Servlets, JSPs, JDBC, JNDI und RMI/IIOP implementiert. Die Basistechnik für verteilte Objekte, Enterprise JavaBeans (EJB), ist nur rudimentär vorhanden. Daher ist der Einsatz von EJBs derzeit nicht zu empfehlen. Da EJB genau die Themen adressiert, für die OS/390 seit Jahren bekannt ist (Transaktionschutz, Sicherheit, Skalierbarkeit, Verfügbarkeit), bietet sich dieses Betriebssystem als nahezu optimale J2EE-Plattform an. Die WebSphere Enterprise Edition sowie der CICS Transaction Server werden zukünftig eine Laufzeitumgebung für

Enterprise JavaBeans, so genannte 'EJB Container', bereitstellen.

Derzeit nicht beantworten läßt sich die Frage, ob IBMs Vision vom Java-Guru-Entwickler Realität wird. Immerhin soll dieser unter Windows NT EJBs entwickeln, seine Anwendung später unter OS/390 produktiv einsetzen, dabei aber keinerlei OS/390-Kenntnisse benötigen und dennoch die klassischen Stärken von OS/390 im vollem Maße ausnutzen können. (JS)

UWE ZANKER

arbeitet als Systemanalytiker und Verantwortlicher für das 'Competence Center Middleware and Application Integration' im Bereich 'Information System Management' von T-Systems.

Literatur

- [1] Robert Orfail, Dan Harkney; Client/Server Programming with Java and CORBA; John Wiley & Sons 1998
- [2] Joseph L. Weber, Joe Weber; Using Java 1.2; Que 1999
- [3] Ed Roman; Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition; John Wiley & Sons 1999
- [4] Empfehlenswert sind auch diverse IBM Redbooks, alle zu finden unter www.redbooks.ibm.com
- [5] Hans Wegener; Grenzkontrolle; Definierte Schnittstellen mit JNI, iX 7/98; S. 111
- [6] Hans Wegener; Pendlar; Die Arbeit mit JNI; iX 11/98; S. 192 

ONLINE-INFOS

Java für OS/390:	http://www.s390.ibm.com/java
IBM Solution Developer Program für Java:	www.developer.ibm.com/java/index.html
IBMs Java-Technologien:	www.alphaWorks.ibm.com/tech
IBM Java Strategy for Enterprise Success:	www-1.ibm.com/servers/eserver/zseries/library/whitepapers/java_success
WebSphere Application Server:	www.software.ibm.com/webservers
VisualAge for Java:	www.software.ibm.com/ad/vajava
CICS Gateway for Java:	www.software.ibm.com/ts/cics/platforms/internet/cicsgw4j
MQSeries:	www.software.ibm.com/ts/mqseries
IBM Support Pack:	www4.ibm.com/software/ts/mqseries/txppacs/ma1g.html
IMS TOC:	www.software.ibm.com/data/ims/imstoc.html