Moderne Verteilungsplattformen für die Automatisierungstechnik

Stefan Lankes, Andreas Jabs
Lehrstuhl für Betriebssysteme,
RWTH Aachen, Kopernikusstr. 16,
52056 Aachen, Germany
E-Mail: {lankes, jabs}@lfbs.rwth-aachen.de

Verteilungsplattformen bieten die Möglichkeit, Aufgaben und Problemstellungen zu abstrahieren und ungeachtet von Systemgrenzen umzusetzen. Die Automatisierungstechnik könnte von den positiven Eigenschaften der Verteilungsplattformen profitieren, jedoch existieren demgegenüber noch häufig Vorbehalte. In diesem Artikel werden Verteilungsplattformen und deren Erweiterung für die Verwendung in eingebetteten, echtzeitfähigen Systemen vorgestellt und anschließend miteinander verglichen.

1 Einleitung

Lange Zeit standen moderne Verteilungsplattformen wie CORBA[1] oder Javas RMI[2] im Ruf, durch ihre Größe und Komplexität nur für Anwendungen im Bereich von Banken und Versicherungen in Frage zu kommen. Doch bieten sie gerade für die Automatisierungstechnik einige interessante Aspekte, welche die Verbreitung moderner Verteilungsplattformen in diesem Bereich voran treiben.

In der Automatisierungstechnik werden häufig eingebettete Systeme eingesetzt, die sich dadurch auszeichnen, dass sie nur über geringe Ressourcen verfügen. Für solche Systeme existiert eine Vielzahl von Betriebssystemen und Netzarchitekturen. Gerade in diesem Umfeld sind Verteilungsplattformen äußerst interessant, da sie dem Entwickler unnötige Implementierungsdetails, wie zum Beispiel das verwendete Betriebssystem, die Art der Verbindung und das verwendete Protokoll verbergen, so dass der Entwickler sein Augenmerk auf die Lösung des eigentlichen Problems richten kann. Durch solche Mechanismen verringert sich die Gefahr von Fehlern und die Qualität der Software wird gesteigert. Außerdem wird durch die Konzentration des Entwicklers auf den Kern seiner Aufgabe die Entwicklungszeit verkürzt, so dass durch die Verwendung moderner Verteilungsplattformen die Entwicklungskosten reduziert werden. Doch benötigen die meisten modernen Verteilungsplattformen zu viele Ressourcen, so dass sie nicht für eingebettete Systeme in Frage kommen. Aus diesem Grund werden zurzeit Verteilungsplattformen für eingebettete, echtzeitfähige Systeme entwickelt. Dabei wird berücksichtigt, dass die bestehenden Verteilungsplattformen mit denen für eingebettete Systeme ohne Einschränkungen kommunizieren können. So können eingebettete Systeme mit bestehenden Anwendungen in der Außenwelt, die keinem Ressourcenmangel unterliegen, problemlos kommunizieren.

2 CORBA für eingebettete, echtzeitfähige Systeme

Die *Object Management Group*, die für die Standardisierung von CORBA zuständig ist, erkannte die Notwendigkeit einer standardisierten Verteilungsplattform für eingebettete Systeme. Aus diesem Grund entwarf sie einen neuen CORBA-Standard, der speziell den Bedürfnissen der eingebetteten Systeme angepasst wurde und unter dem Namen *minimumCORBA* [OMG98a] bekannt ist.

Für verteilte, echtzeitfähige Anwendungen reicht dieser Standard jedoch nicht aus. Um diese zu ermöglichen, definierte die OMG in [OMG98b] einen weiteren Standard, der CORBA mit Echtzeitfunktionalitäten erweitert. Dieser ist unter dem Namen *Real-Time CORBA* bekannt und wird ausführlicher in [SK00] erläutert. Für ihn existieren mit *ORBexpress* von *Object Interface*, *TAO* von der *Distributed Object Computing Group* und ROFES [Lan03] vom Lehrstuhl für Betriebssysteme erste Implementierungen.

Beispielsweise definiert der Real-Time CORBA-Stan-

[1] *Common Object Request Broker Architecture*
[2] *Remote Method Invocation*

dard, wie eine verteilte Client/Server-Anwendung Prioritäten behandelt. In einem solchen Szenario stellt der Client Anfragen an einen Server, der diese bearbeitet und sein Ergebnis anschließend an den Client zurückliefert. Meistens bestehen verteilte Anwendungen aus mehreren Clients, die Anfragen an den Server stellen. Bei verteilten Echtzeitanwendungen sollten die Anfragen desjenigen Clients zuerst bearbeitet werden, dessen Priorität am höchsten ist. Doch im ursprünglichen CORBA-Standard werden dem Server keinerlei Informationen mitgeliefert, mit welcher Priorität die einzelnen Clients arbeiten, die die Anfrage an den Server gestellt haben. Demzufolge kann der Server nicht entscheiden, welche Anfrage wichtiger ist. Somit wird der Server die Anfragen als gleichberechtigt betrachten und im Round-Robin-Verfahren bearbeiten. Diese Vorgehensweise ist für Echtzeitsysteme vollkommen ungeeignet. Solche und ähnliche Probleme werden mit der Real-Time CORBA-Spezifikation [OMG98b] gelöst. Beim Versenden einer Anfrage wird zusätzlich die Priorität des Clients mit versendet. Auf diese Weise kann der Server entscheiden, welche Anfrage bedeutender ist und zuerst bearbeitet wird.

Leider definiert die OMG in ihren Standards nur ein Protokoll zur Kommunikation zwischen CORBA-Anwendungen, das auf TCP/IP-basiert. Typische Netze für eingebettete Systeme wie das *Controller Area Network* (CAN) werden nicht berücksichtigt. Allerdings bieten die Hersteller von CORBA-Implementierungen Schnittstellen an, mit dessen Hilfe solche Netze einfach zu integrieren sind. Zudem existieren Forschungsprojekte [KJH00, LJB05], die den CAN-Bus integrieren.

3 Java für eingebettete, echtzeitfähige Systeme

Wie bereits erläutert, existiert eine Vielzahl von Betriebssystemen für eingebettete Systeme. Würde der Programmierer Java verwenden, müsste er nicht die speziellen Eigenschaften der einzelnen Betriebssysteme erlernen, da seine Java-Applikationen auf jedem Betriebssystem laufen würden, das eine Virtuelle Maschine besitzt. Dies könnte die Zeit für die Entwicklung eines Programmes verkürzen und die Entwicklungskosten reduzieren. Die Hersteller der Betriebssysteme für eingebettete und echtzeitfähige Systeme haben dies erkannt und bieten für ihre Systeme eine spezielle Version der Java Virtual Machine an. Sun selbst stellte mit ihrer *Java 2 Micro Edition* (J2ME) eine stark reduzierte Version von Java für eingebettete Systeme vor.

Durch die Java 2 Micro Edition wird der Speicherverbrauch stark reduziert, es werden jedoch noch immer mehr Ressourcen als bei Lösungen, die zum Beispiel auf C bzw. C++ basieren, benötigt. Außerdem verwenden Java-Lösungen eine virtuelle Maschine, die den Java-Byte-Code interpretiert und somit mehr Rechenleistung als C/C++-Lösungen benötigt. Um die mangelnde Leistungsfähigkeit der Interpretation von Java-Code zu umgehen, wurden Java-Compiler entwickelt, die Maschinencode erzeugen.

Erste Ansätze für eine Java-Implementierung mit echtzeitfähigen Eigenschaften werden seit einigen Jahren diskutiert und entwickelt. Dies erfordert weitere Änderungen in der Java-Architektur. Hierzu wurde eine Expertengruppe gebildet, die ihre Lösung in der *Real-Time Specification for Java* [Rea01] vorgestellt hat. Leider geht die Spezifikation nicht auf die Probleme verteilter Anwendungen ein. Zurzeit entstehen erste Überlegungen, wie dies gelöst werden kann. Sie sollen in der *Distributed Real-Time Specification for Java* standardisiert werden.

4 Zusammenfassung

Durch die Verwendung von Verteilungsplattformen werden Implementierungsdetails abstrahiert, so dass sich der Entwickler auf die Lösung der eigentlichen Aufgabenstellung konzentrieren kann. Durch solche Mechanismen verringert sich die Gefahr von Fehlern und die Qualität der Software wird gesteigert. Allerdings vergrößert sich der Ressourcenverbrauch bei der Verwendung solcher Verteilungsplattformen.

Dies wurde erkannt, so dass sowohl Java- als auch CORBA-basierte Lösungen bezüglich ihres Ressourcenverbrauchs verbessert wurden. Hierdurch sind beide Verteilungsplattformen für die Verwendung im Bereich der eingebetteten Systeme geeignet. Müssen Echtzeitbedingungen erfüllt werden, so muss auf Real-Time CORBA zurückgegriffen werden, welches die einzige Lösung für verteilte, echtzeitfähige Anwendungen darstellt.

Literatur

[KJH00] KIM, T. ; JEON, G. ; HONG, S.: Seamless Integration of Real-Time Communications into CAN-CORBA with Extended IDL and Fast-Track Messages. In: *IFAC Workshop on Distributed Computer Control Systems (DCCS)*. Sydney, Australia, Dezember 2000

[Lan03] LANKES, S.: *Konzeption und Umsetzung einer echtzeitfähigen Verteilungsplattform für eingebettete Systeme*. Shaker Verlag Aachen, 2003. – ISBN 3–8322–2205–7

[LJB05] LANKES, S. ; JABS, A. ; BEMMERL, T.: Design and Performance of a CAN-based Connection-oriented Protocol for Real-Time CORBA. In: *Journal of Systems and Software* Vol. 77 (2005), Nr. 1, S. 37–45

[OMG98a] OMG Technical Document orbos/98-08-04: *minimumCORBA – Joint Submission*. 1998

[OMG98b] OMG Technical Document orbos/98-10-05: *Realtime CORBA – Joint Submission*. 1998

[Rea01] The Real Time for Java Experts Group: *Real-Time Specification for Java*. 2001. – Version 1.0

[SK00] SCHMIDT, D. C. ; KUHNS, F.: An Overview of the Real-Time CORBA Specification. In: *IEEE Computer* 33 (2000), Nr. 6, S. 56–63