Entwurf und Evaluation eines verteilten Laufzeitsystems zur Ausführung kontextbewusster Anwendungen basierend auf Agententechnologie

Alexander Merkle, Daniel Graff

Fachgebiet Kommunikations- und Betriebssysteme
Technische Universität Berlin
Einsteinufer 17 / EN 6
10587 Berlin
merkrmid@mailbox.tu-berlin.de, daniel.graff@tu-berlin.de

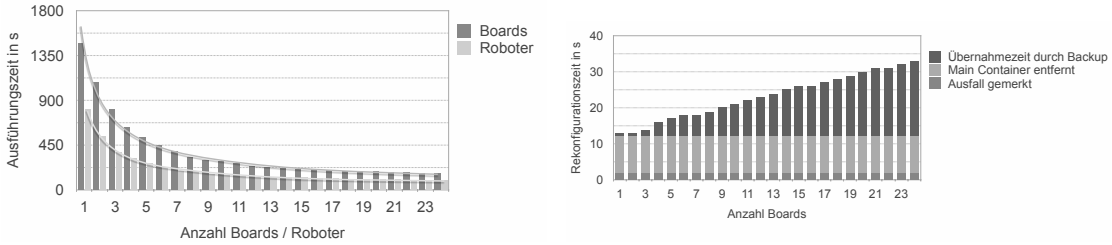
Bereits heutzutage existiert eine Vielzahl diverser mobiler Geräte, angefangen von Smartphones über Wearable Devices hin zu vollkommen autonomen Robotern. Unser Ansatz besteht darin, für die Vielzahl dieser heterogenen Geräte ein verteiltes Laufzeitsystem zu konzipieren, welches als Ausführungsplattform für verteilte, kontextbewusste Anwendungen dient. In [GRW13] wurde eine Referenzarchitektur und ein Programmiermodell für verteilte kontextbewusste Anwendungen vorgestellt. In dieser Arbeit präsentieren wir ein auf Agententechnologie basierendes Laufzeitsystem, welches die Referenzarchitektur implementiert, sodass das bestehende systemische Programmiermodell unterstützt wird. Dazu erweitern wir JADE [BCG07] um weitere Verwaltungsdienste und nutzen das Kommunikationssystem sowie die Fehlertoleranzmechanismen des Frameworks.

Listing 1 zeigt ein Programmausschnitt zur zeit- und ortsbezogenen Ausführung von Komponenten eines Programms für bspw. Überwachungsaufgaben. Das Programm wurde nach dem Programmiermodell entwickelt und ermöglicht das Binden von Raum-Zeit-Bedingungen an Programmteile. Durch Code-Synthese wird daraus ein verteiltes Programm generiert, welches räumlich, zeitlich und im Kontrollfluss entkoppelt ist. Aspekte der Synchronisation, Nebenläufigkeit und Verteiltheit werden vor dem Programmierer verborgen und sind Aufgabe des Laufzeitsystems. In dem vorliegenden Beispiel sind zwei Methoden (`first()`, `second()`) gezeigt, die durch die *Time*- und *Geo*-Annotationen räumlich und zeitlich entkoppelt sind. Erstere spezifiziert dabei das Zeit-Fenster während letztere das Raum-Fenster beschreibt innerhalb welcher eine einmalige Ausführung stattfinden soll. Der Programmierer hat durch `init` und `finish` die Möglichkeit, vor bzw. nach dem Ausführen in den Programmfluss einzugreifen, um Variablen zu initialisieren bzw. auszulesen. Während der Abarbeitung des verteilten Programms ist es dem Programmierer möglich, in jedem verteilt ausgeführten Programmsegment auf eine globale Datenstruktur, die durch die *Global*-Annotation global gemacht wird, zuzugreifen. Somit kann eine Kommunikation zwischen den Programmsegmenten erreicht werden. Zur Ausführung des beschriebenen Programms wird ein Code-Generator sowie ein verteiltes Laufzeitsystem benötigt. Wir benutzen im Kern das auf Agententechnologie basierte Framework JADE und haben dieses um zwei weitere Schichten erweitert.

Auf der untersten Schicht verwenden wir die von JADE bereit gestellten Agenten AMS (Agent Management System), DF (Directory Facilitator) und RMA (Remote Monitoring Agent). Letztgenannter ist dabei optional und dient der grafischen Administration des Systems. Der AMS Agent verwaltet alle im System vorhandenen Agenten und der DF stellt einen Yellow Pages Service zur Verfügung, bei dem sich Agenten registrieren können, um von anderen bezüglich eines bestimmten Service gefunden werden zu können. Darauf aufbauend haben wir eine Systemschicht etabliert, die für die Code-Synthese, das verteilte Ausführen der Programme sowie für die Ressourcen-Verwaltung des Systems zuständig ist. Da mehrere unabhängig voneinander

```
@Global
List list;

init() { list = new List() } // Initialisierung

@Time("00:00:00;23:59:59")
@Geo("2,2;4,4")
first() { /*read sensor and put into list*/ }

@Time("00:00:00;23:59:59")
@Geo("10,10;12,12")
@DependsOn("first")
second() { /*read sensor and put into list*/ }

finish(result) { print(list) } // Ende
```

Listing 1: Programmausschnitt



(a) Laufzeit für das Lösen von Primzahlen in Abhängigkeit von dem Grad der Verteiltheit

(b) Rekonfigurationszeit für den Ausfall des Main-Containers in Abhängigkeit von der Anzahl angebundener Peers

Abbildung 1: Evaluation des Laufzeitsystems

ander entwickelte Programme ausgeführt werden sollen, werden diese von einem Scheduler in Raum und Zeit eingeplant. Diese Komponente ist nicht Teil dieser Arbeit und wird daher nicht weiter erwähnt. Auf der obersten Schicht befinden sich die Anwendungsprogramme. Bevor ein neu gestartetes Programm ausgeführt wird, erfolgt zunächst eine Analyse und durch Code-Synthese wird dies in ein verteiltes Programm überführt. Da JADE agentenorientiert arbeitet, werden die annotierten Programmteile als ein oder mehrere eigenständige Agenten dargestellt. Das generierte Programm besteht also aus einer Menge von Agenten, die innerhalb der JADE Plattform migrieren können. Es gibt zwei Arten von Bewegungen: logische und physische. Das Migrieren von Agenten stellt eine logische Bewegung dar, da hier Programmcode über Rechnergrenzen hinweg ausgetauscht wird. Hat ein Programm aber dagegen die Anforderung, an einer speziellen Koordinate ausgeführt zu werden, an welcher sich aktuell kein Roboter befindet, so muss durch physisches Bewegen ein Roboter entsprechend positioniert werden. Das Berechnen der Roboter-Trajektorie ist ebenfalls Aufgabe des Schedulers. Anschließend wird ein Bewegungsagent beauftragt, die berechnete Trajektorie durch präzise Motorregelung abzufahren.

Als Proof-of-Concept wird in Abbildung 1(a) die Laufzeit für das verteilte Suchen von Primzahlen in dem Intervall von 1 bis 200.000 evaluiert. Der Test wurde dabei so durchgeführt, dass sukzessive die Anzahl der Agenten, die jeweils auf einem separaten Knoten ausgeführt wurden, erhöht wurde. In diesem Fall wurde für jedes Programmteil eine andere Koordinate in der Geo-Annotation verwendet. Das Testbed bestand aus 24 mobilen Robotern (400 MHz ARM9 CPU) und 24 stationären Boards (180 MHz ARM9 CPU), die alle über 64 MB SDRAM verfügen. In Abbildung 1(b) haben wir die Rekonfigurationszeit gemessen, die benötigt wird, um beim Ausfall des Main-Containers, welches die Hauptverwaltungskomponente des Systems darstellt, wieder in einen korrekten Zustand zu gelangen. Der Main-Container kann (auch zur Laufzeit) beliebig oft repliziert werden, um bei einem Ausfall von einem der Backups ersetzt zu werden. Insgesamt können n-1 Ausfälle toleriert werden. Wir haben den Ausfall des Main-Containern mit bis zu 24 Robotern evaluiert. Unabhängig von der Anzahl der Geräte war die Zeit, bis der Ausfall detektiert wurde, sowie die Zeit, bis der Main-Container aus der Liste der aktiven Container und somit aus dem System entfernt wurde, konstant. Ein lineares Anwachsen der Übernahmezeit durch das Backup konnte dabei in Abhängigkeit der Anzahl der verwendeten Geräte festgestellt werden. Dies ist dadurch zu erklären, dass sich die Knoten nach der Übernahme der Verwaltungsaufgaben durch einen der Backup Main-Container neu mit diesem verbinden müssen. Zusätzlich müssen alle Agenten die Registrierung ihrer Services beim DF wiederherstellen. Auch das dauert länger bei einer höheren Anzahl an Geräten, deren Agenten je zwei Services (Ortung und Bewegung) bereitstellen. In der Summe erhöht sich daher die Rekonfigurationszeit des Gesamtsystems mit jedem zusätzlichen Knoten.

Literatur

[BCG07] Fabio Bellifemine, Giovanni Caire und Dominic Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons, 7. Auflage, 2007. 1

[GRW13] Daniel Graff, Jan Richling und Matthias Werner. Programming and Managing the Swarm – An Operating System for an Emerging System of Mobile Devices. In Kai Lin, Heng Qi, Keqiu Li, Ivan Stojmenovic, Albert Zomaya, Hongyi Wu, Song Guo und Symeon Papavassiliou, Hrsg., *9th IEEE International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2013)*, Seiten 9–16. IEEE Computer Society, December 2013. 1