Ein Raum-Zeit-Scheduler und Trajektorienplaner für ein Schwarmsystem

Gliederung

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

- Schwierigkeit bei der Entwicklung verteilter Raum-Zeit-Anwendungen
- SwarmOS vorstellen
- Notwendigkeit eines Schedulers darlegen

1.2 Zielsetzung und Abgrenzung

- Was muss der Scheduler können? (Einplanen :P)
- Was muss er noch nicht können? (hinsichtlich Fehlermodell u.Ä.)

1.3 Struktur der Arbeit

2 Grundlagen

- Bisher noch knapp. Einige Grundlagen werden sich durch die Inhalte in der Problemanalyse und dem Lösungsansatz noch ergeben.

2.1 Path-Velocity-Decomposition

- Aufs Nötigste beschränkte Erklärung des Verfahrens.

3 Begriffe?

- Noch unsicher wie und wo ich diese Begriffe erläutere. Gehört eigentlich zu den Grundlagen :/

3.1 Job

3.2 Knoten

3.3 Constraint

3.4 Transaktion

4 Entwicklung verteilter Anwendungen [Problemanalyse]

- Erläuterung der Probleme bei der Entwicklung verteilter Raum-Zeit-Anwendungen
- Ableiten der Anforderungen einer Abstraktion
    - Transparenz hinsichtlich: Heterogenität, Koordination, Wegplanung, Wegausführung, …
    - Konzept der Constraint getriebenen Einplanung
- Was muss der Scheduler können? (konkrete Anforderung im Gegensatz zur Zielsetzung und Abgrenzung)

5 Raum- und Echtzeitbedingungen [Problemanalyse]

- Welche Invarianten müssen gelten?

- Rolle der Gegenwart
- Raum-Zeit-Eigenschaften der Knoten

6 Modell des allwissenden Schedulers [Lösungsansatz]

- Vorstellen des Modells
- Warum ist es ungeeignet?
- Warum wird es dennoch verwendet?
- Fehlermodell („keine“ Fehler)

7 Jobplanung [Lösungsansatz]

7.1 Einzelner Job

- Belegung der Variablen Ort, Zeit, Knoten
- Einhaltung der Invarianten (Kollisionsfreiheit, Gegenwart, Spezifikation, …)

7.2 Mehrere Jobs

- Umgang beim Einplanen mehrere Jobs (Alternativen beachten)

7.3 Periodische Jobs

- Reduzierung auf Einplanung mehrerer einzelner Jobs

7.4 Abhängige Jobs

- Reduzierung auf Einplanung mehrerer einzelner Jobs
- Klassisches Modell ohne relative Contraints
- Erweitertes Modell mit relativen Constraints

7.5 Job entfernen

- Optimierung des Weges
- Einhaltung der Invarianten (Kollisionsfreiheit, Gegenwart, Spezifikation, …)

7.6 Job umplanen

- atomares Entfernen und Neuplanen

7.7 Commit/Abort

- Warum commit/abort-Semantik?
- Realisierung durch Locks und Beachtung mehrerer Alternativen.

7.8 Umgang mit Fehlern

- Was kann man bei welchen Fehlersituationen unternehmen?
- Wo fällt das Modell des allwissenden Schedulers auf die Füße?

7.9 Geister

- Was sind Geister?
- Wie kann man sie vermeiden?
- Was macht die Vermeidung so kompliziert?

- Warum werden sie in Kauf genommen?

8 Trajektorienplanung [Lösungsansatz]

8.1 Räumliche Planung

- Generelles Vorgehen (vor allem Unterschiede zum Abschnitt Path-Velocity-Decomposition)

8.2 Temporale Planung

- Berechnung der Verbotenen Regionen (im dazugehörigen Paper nur angerissen)
- Parameter und Optionen

9 Raum-Zeit-Scheduler in Java

9.1 Struktur/Komponenten

- genereller Überblick der Kernkomponenten und ihrer Aufgaben

9.2 Konventionen

- Schnittstellenkonventionen
- Funktionales Design (kein Dienst)

9.3 Abhängigkeiten

- Bibliotheken: Java8, JTS, JGraphT, straightedge

9.4 Schnittstellen

- Überblick der wichtigsten Schnittstellen ((un-/re-)schedule, commit, abort, …)
- Informative Schnittstellen (calcLoad, etc.)

9.5 Abweichungen zum Ansatz

- keine erweiterten Abhängigkeiten (relative Constraints)
- …

10 Komplexität [Evaluation]

- Evaluation der eigenen Kernkomponenten (Zeit, evtl. auch Speicher)
    - Belegung von Variablen für einen einzelnen Job
    - Einplanen mehrerer Jobs (periodisch, abhängig)
    - Trajektorienplanung (Verbotene Regionen, Meshing, Dijkstra)

11 Benchmark [Evaluation]

- Analog zur Komplexität
    - eventuell Verzahne ich Komplexität und Benchmark sodass komponentweise die Ergebnisse vorgestellt werden

12 Erfahrung mit dem Scheduler?

- Vielleicht kann ich mich schon auf einige eurer Erfahrungen berufen, die ihr in der Zeit mit dem Scheduler sammelt.

13 Fazit und Ausblick

- Was wurde erreicht?
- Was wurde nicht umgesetzt?
- Wo liegen die Grenzen?
- Was kommt als nächstes?