Integration unterschiedlicher Planungsprobleme

Agentensystem GNUBrain

René Drießel

Fachgebiet Fabrikbetriebe – Fakultät Maschinenbau
TU-Ilmenau

26. Januar 2006

Motivation

- Integration von Planungsproblemen (z. B. Transport- und Feinplanung)
- Sehr große Modelle

⇒ Laufzeitprobleme mit iterativen Problemlösern

Ausweg: Parallelisierung und Verteilung

- Verteilung des Datenmodells
- Verteilung der Berechnungen
- Steuerung der Verteilung und Parallelisierung
- Sicherstellung der Konsistenz der Ergebnisse

⇒ Verwendung Agententechnologie

Motivation

- Integration von Planungsproblemen (z. B. Transport- und Feinplanung)
- Sehr große Modelle

$\Rightarrow$ Laufzeitprobleme mit iterativen Problemlösern

Ausweg: Parallelisierung und Verteilung

- Verteilung des Datenmodells
- Verteilung der Berechnungen
- Steuerung der Verteilung und Parallelisierung
- Sicherstellung der Konsistenz der Ergebnisse

$\Rightarrow$ Verwendung Agententechnologie

Motivation

Designschwächen bisheriger Agentenframeworks

- Fokussierung auf eine Agentenplattform und nicht auf den einzelnen Agenten
- Komplizierte Agentenkommunikation mittels RMI/CORBA
- Portabilität

Agentensystem GNUBrain

- Fokussierung auf einen Agenten
- Einfache Agentenkommunikation über XML
- Portabilität mittels GNU autoconf und MinGW
- Verwendung von Freier Software

Motivation

Designschwächen bisheriger Agentenframeworks

- Fokussierung auf eine Agentenplattform und nicht auf den einzelnen Agenten
- Komplizierte Agentenkommunikation mittels RMI/CORBA
- Portabilität

Agentensystem GNUBrain

- Fokussierung auf einen Agenten
- Einfache Agentenkommunikation über XML
- Portabilität mittels GNU autoconf und MinGW
- Verwendung von Freier Software

Datenhaltung

Grundidee

- Objektidentifikation durch globale ID (Eindeutigkeit durch Namespaces)
- Serialisierung als XML
- Pro ID mehrere Semantiken möglich (Typen)

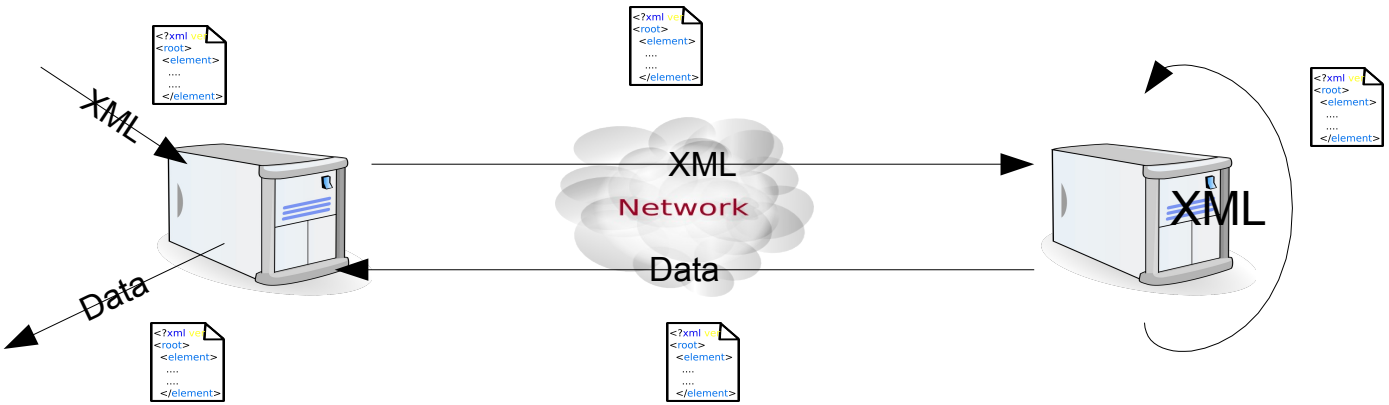
Beispiel

```
<id name = "/land/unternehmen/werk_1/tool_1">
  <transport weight="10" length="20" />
  <machine batch-min="1" batch-max="3" />
</id>
```

Kommunikation / Verarbeitung

Grundidee

- Agentenkommunikation mittels XML
- XML Nachricht → Thread → XML Nachricht
- Terminierung, wenn Antwortdokument vom Typ `http://gnubrain.org/types:data`



Plugin “process”

Stellt grundlegende Verarbeitungsdirektiven zur Verfügung.

Methoden

- process-iterative
- process-parallel
- process-single
- process-background

Beispiel

```
<process-parallel>
  <print>Hallo Dresden!</print>
  <print>Hallo Welt!</print>
</process-parallel>
```

Plugin “process”

Stellt grundlegende Verarbeitungsdirektiven zur Verfügung.

Methoden

- process-iterative
- process-parallel
- process-single
- process-background

Beispiel

```
<process-parallel>
  <print>Hallo Dresden!</print>
  <print>Hallo Welt!</print>
</process-parallel>
```

Plugin “share”

Verwaltung der verteilten Daten in Anlehnung der ”Theory of Patches”:

- Objekt besitzt verantwortlichen Agenten
- Agent kann Daten vom Masteragent bei Bedarf beziehen
- Kommunikation der Änderungen über Patches
- Werden bei Bedarf synchronisiert (Publish/Subscribe)

Methoden

- get-id - Liefert Objekt zurück
- subscribe - Registrierung für Änderungsbenachrichtigungen
- update-data - Änderung einer Semantik eines Objekts
- patch - Aggregation mehrerer Änderungen

Plugin "share"

Verwaltung der verteilten Daten in Anlehnung der "Theory of Patches":

- Objekt besitzt verantwortlichen Agenten
- Agent kann Daten vom Masteragent bei Bedarf beziehen
- Kommunikation der Änderungen über Patches
- Werden bei Bedarf synchronisiert (Publish/Subscribe)

Methoden

- get-id - Liefert Objekt zurück
- subscribe - Registrierung für Änderungsbenachrichtigungen
- update-data - Änderung einer Semantik eines Objekts
- patch - Aggregation mehrerer Änderungen

Plugin “simulation”

Implementiert einen diskreten ereignisgesteuerten Simulator:

- Ereignisse führen zu Methodenaufrufen
- Ereignisse zum gleichen virtuellen Zeitpunkt werden parallel verarbeitet (process-parallel)

Beispiel

```
<process-iterative model="sim">
  <register-events time="0">
    <print>Hallo Dresden!</print>
    <print>Hallo Welt!</print>
  </register-events>

  <simulate />
</process-iterative>
```

Plugin “simulation”

Implementiert einen diskreten ereignisgesteuerten Simulator:

- Ereignisse führen zu Methodenaufrufen
- Ereignisse zum gleichen virtuellen Zeitpunkt werden parallel verarbeitet (process-parallel)

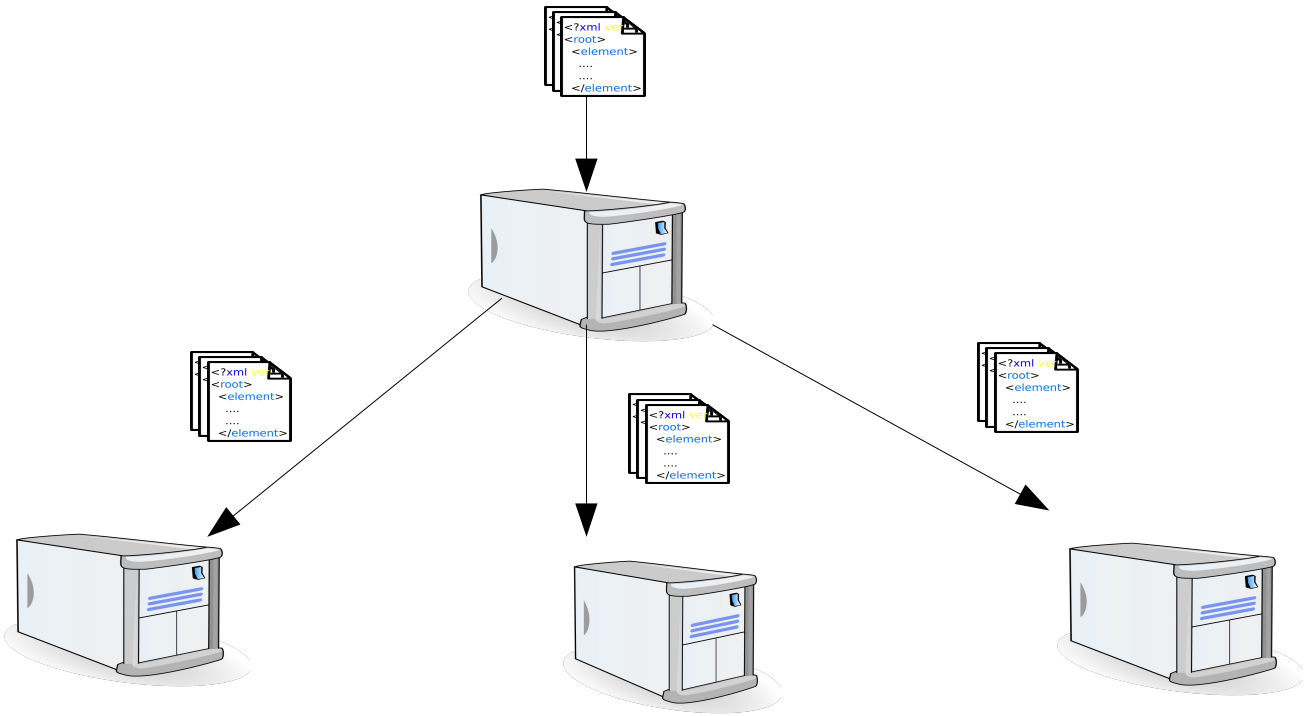
Beispiel

```
<process-iterative model="sim">
  <register-events time="0">
    <print>Hallo Dresden!</print>
    <print>Hallo Welt!</print>
  </register-events>

  <simulate />
</process-iterative>
```

Paralleles Lösen des Gesamtproblems

Einzelne Agenten berechnen das Gesamtproblem mit verschiedenen Algorithmen / Parametern (z.B. Simulation mit verschiedenen Dispatchregeln). Die beste Lösung wird genommen.



Beispiel - Paralleles Lösen des Gesamtproblems

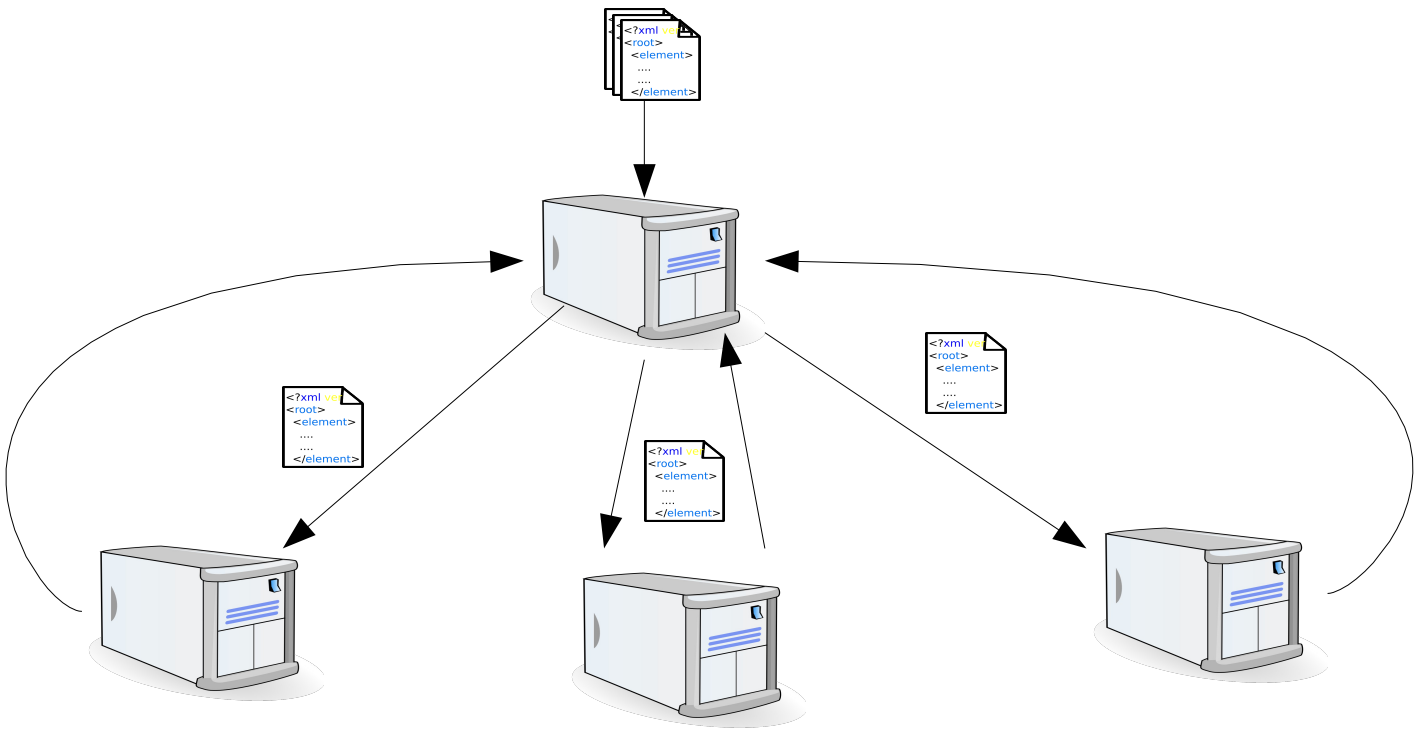
```
<process-iterative>
  <load-file file="model.xml" />

  <process-parallel>
    <schedule model= "/fab" receiver= "agent-1" />
    <schedule model= "/fab" receiver= "agent-2" />
  </process-parallel>

  <select-solution>
    <agent name= "agent-1" />
    <agent name= "agent-2" />
  </select-solution>
</process-iterative>
```

Paralleles Lösen von Teilproblemen

Das Problem wird in Teilprobleme zerlegt. Die einzelnen Agenten lösen diese Teilprobleme. Die Gesamtlösung wird aus den Lösungen der Teilprobleme komponiert.



Beispiel - Paralleles Lösen von Teilproblemen

```
<process-iterative>
  <simulate model="/fab"/>
  ...
  <process-parallel>
    <schedule model="/fab/t-1" receiver="agent-1" />
    <schedule model="/fab/t-2" receiver="agent-2" />
  </process-parallel>
  ...
  <implement-best-solution>
    <agent name="agent-1" />
    <agent name="agent-2" />
  </implement-best-solution>
</process-iterative>
```

Zusammenfassung

Anforderungen durch die Integration von Planungsproblemen

- Flexible Abbildung von Problemen notwendig
- Komplexitätsbewältigung durch Verteilung und Parallelisierung

Umsetzung mittels GNUBrain

- Datenintegration durch verschiedene Semantiken pro ID
- Steuerung der Berechnungen (process-parallel etc.)
- Integration und Verabeitung beliebiger Probleme möglich (Voraussetzung: Methoden, Typen)

Zusammenfassung

Anforderungen durch die Integration von Planungsproblemen

- Flexible Abbildung von Problemen notwendig
- Komplexitätsbewältigung durch Verteilung und Parallelisierung

Umsetzung mittels GNUBrain

- Datenintegration durch verschiedene Semantiken pro ID
- Steuerung der Berechnungen (process-parallel etc.)
- Integration und Verabeitung beliebiger Probleme möglich (Voraussetzung: Methoden, Typen)

Weiterführende Informationen

GNUBrain

http://gnubrain.org/

Darcs Repository

darcs get http://gnubrain.org/repo/gnubrain/

“Theory of Patches”

http://www.abridgegame.org/darcs/manual/

Agententechnologie

http://fipa.org/
http://agentlink.org/