Überlegungen zur parallelen Verarbeitung in Konfigurierungssystemen

Lothar Hotz *

Zusammenfassung

Dieser Beitrag untersucht die Verwendung paralleler Methoden für die Lösung von Konfigurierungsaufgaben. Dabei wird von einer Synthese der Konfiguration und damit einer schrittweisen Entwicklung einer Konfiguration ausgegangen. Um Parallelität zu untersuchen, werden als Abstraktion komplexe Schritte eingeführt, die je nach Verwendung unterschiedliche Granularitäten, d.h. Grade von Parallelverarbeitung, repräsentieren. Weiterhin werden Einflußfaktoren auf die parallele Verarbeitung diskutiert.

1 Einleitung

Gegenwärtiges Ziel in der Konfigurierung ist die Untersuchung der Anwendbarkeit von Konfigurierungsmethoden auf reale Domänen und damit der Transfer von wissenschaftlichen Methoden zu wirtschaftlichen Anwendungen (siehe z.B. [7]). Reale Anwendungen stellen im Gegensatz zu Spielbeispielen und Prototypentwicklungen durch Größe und Komplexität höchste Anforderungen an die schnelle Bearbeitung der gestellten Probleme. In [4] wird die These vertreten, daß aus realen Anwendungen kommende Probleme, mittels KI-Methoden nur mithilfe paralleler Rechner (z.B. massiv-paralleler Architekturen) gelöst werden können. Auch in [6] wird als wichtigster Begrenzungsfaktor wissensbasierter Systeme ihre Verarbeitungsgeschwindigkeit betrachtet. Gerade die Kombination aufwendiger Verfahren, wie eines Assumption Based Truth Maintenance System (ATMS) und eines Constraint-Systems, wird aus Effizienzüberlegungen vermieden. Die Betrachtung der Parallelisierbarkeit von Konfigurierungsmethoden erscheint unter diesem anwendungsbezogenen Aspekt unausweichlich.

Um einen Entwickler von Konfigurierungsmethoden von Problemen paralleler Verarbeitung zu entlasten, entwickeln wir ein Werkzeugsystem, welches parallele Verarbeitung verbirgt. Um Merkmale eines solchen Werkzeugs zu ermitteln, betrachten wir in diesem Beitrag die wichtigsten Einflußfaktoren auf die parallele Verarbeitung – Kontroll- und Datenabhängigkeiten. Vorhandene Abhängigkeiten zwischen Berechnungen implizieren, daß diese nicht parallel ausgeführt werden können.

*Universität Hamburg, Fachbereich Informatik, Vogt-Koelln-Str. 30, 22527 Hamburg, e-mail: hotz@informatik.uni-hamburg.de

Wir fokussieren Abhängigkeiten, die in einem Konfigurierungsvorgang auftreten. Daher steht die Suche nach einer Menge von Konfigurierungsschritten (hier "komplexer Schritt" genannt), die simultan ausgeführt werden können, im Vordergrund. Wir gehen von einem für Konfigurierungssysteme bereits bewährten Vorgehen aus: dem schrittweisen, sequentiellen Entwickeln einer vollspezifizierten Konfiguration aus einer Aufgabenstellung. Die durchgeführten Schritte sind jedoch teilweise unabhängig voneinander, so daß sie zumindest theoretisch parallel ausführbar sind. Dies soll im vorliegenden Beitrag präzisiert werden.

In Abschnitt 2 wird ein Rahmen für ein strukturorientiertes Konfigurierungssystem beschrieben.[1] In Abschnitt 3 werden durch die Einführung eines komplexen Schritts unterschiedliche Parallelisierungsmöglichkeiten diskutiert. In Abschnitt 4 stellen wir Merkmale vor, die für einen Systementwurf eines parallelen Konfigurierungssystems wichtig sind und beschreiben einige mögliche Varianten.

2 Konfigurierung durch schrittweise Entwicklung

Wir gehen hier von einem strukturorientierten Konfigurierungsmodell aus, wie es in Systemen wie IDAX [8] oder KONWERK [2] zur Anwendung kommt. Komponenten werden über Eigenschaften (Relationen) beschrieben, jede Eigenschaft besitzt einen Wertebereich. Beziehungen zwischen mehreren Komponenten und/oder Eigenschaften werden über komponentenübergreifende Relationen beschrieben. Diese können auch funktional repräsentiert sein, eine Einschränkung auf endlich aufgezählte Relationenmengen erscheint uns für viele Anwendungen zu restriktiv. "Konfigurieren" bedeutet nun, ausgehend von einer partiell gegebenen Beschreibung von Komponenten und Eigenschaften ("Aufgabenstellung"), eine vollständige Konfiguration bestehend aus obligatorischen und evtl. optionalen Komponenten mit deren Eigenschaftsbelegungen zu synthetisieren. In einem Konfigurierungssystem mit schrittweiser Entwicklung können aus einer solchen Repräsentation die während des Konfigurierungsvorgangs notwendigen Schritte abgeleitet werden. Ein durchzuführender Schritt (Konstruktionsschritt) besteht aus einer Komponente, einer ihrer Eigenschaften, für die ein Wert festgelegt werden soll, einer Menge von Wertberechnungsverfahren (wie Funktionen, Festlegungen durch den Benutzer, Defaultwerte, etc.) sowie Voraussetzungen des Schritts. Letztere beschreiben, welche Eigenschaften bestimmt sein müssen, um den Konstruktionsschritt durchführen zu können (d.h. den Wert bestimmen zu können). Ein Beispiel ist durch die Argumente einer einen Wert berechnenden Funktion gegeben. Der Konfigurierungsprozess bestimmt, welches Argument einer Relation (Stelle der Relation) von welchen anderen Argumenten abhängt, d.h. welche(s) Argument(e) zuerst berechnet wurde(n). Als "Schritte bezüglich einer Komponente" bezeichnen wir die Konstruktionsschritte, die die Eigenschaften der Komponente bestimmen.

In anderen Konfigurierungsmethoden stehen nicht Konstruktionsschritte als Basisentitäten im Vordergrund, sondern z.B. Ressourcen oder vollständig vorspezifizier-

[1] Die Überlegungen lassen sich jedoch auch auf andere Konfigurierungsmethoden (wie z.B. fall- oder ressourcen-basiert) übertragen, da sie grundsätzlicher Natur sind.

te Komponenten oder Konfigurationen. Wesentlich für die Parallelisierbarkeit dieser Methoden, ist die zum folgenden analoge Berücksichtigung von Voraussetzungen und Abhängigkeiten dieser Basisentitäten.

3 Komplexe Konstruktionsschritte

Wir führen einen komplexen Konstruktionsschritt (KKS) als Abstraktion ein, um die Merkmale paralleler Verarbeitung (wie Abhängigkeiten, Synchronisation, Granularität) diesen Schritten zuzuordnen. Dazu werden sequentielle Schritte zu komplexen Schritten gebündelt. Komplexe Schritte bilden damit Partitionierungen der Menge der Konstruktionsschritte in möglicherweise parallel ausführbare Teilmengen. Jeder komplexe Schritt steht für eine Aufgabe (Task), mit deren Lösung später ein Agent beauftragt wird. Diese Aufgabe wird durch die Größe und Art des komplexen Schritts bestimmt. Wir bezeichnen im folgenden mit "Agent" ein Modul, welches in der Lage ist, einen komplexen Schritt auszuführen. Je nach Art des Schritts ist der Agent mehr oder weniger komplex – ein Agent kann über einen Thread, einen Prozess oder ein komplexes Anwendungssystem (z.B. LISP-Image) realisiert sein.

Ein komplexer Schritte hat folgende Eigenschaften:

a) Ein komplexer Schritt enthält beliebig viele (evtl. auch nur einen) sequentielle Schritte.
b) Die *einem* komplexen Schritt zugeordneten sequentiellen Schritte werden *sequentiell* abgearbeitet.
c) Parallele Verarbeitung wird durch die parallele Ausführung mehrerer komplexer Schritte erzeugt. Jeder komplexe Schritt kann zu jedem anderen komplexen Schritt unter Berücksichtigung von Abhängigkeiten parallel ausgeführt werden.
d) An jedem komplexen Schritt wird vermerkt (manuell oder automatisch), welche komplexen Schritte Voraussetzungen für seine Berechnung bilden (Abhängigkeiten des komplexen Schritts). Die Voraussetzungen eines komplexen Schritts werden durch die Vereinigung der Voraussetzungen der sequentiellen Schritte, für die er steht, festgelegt.

Je nachdem, welche Schritte ein KKS enthält, sind verschiedene Arten von Parallelität und daraus abzuleitende Systeme zu unterscheiden. Durch diese Arten komplexer Schritte wird es möglich, das parallel zu verarbeitende Konfigurierungsproblem zu skalieren, d.h. möglichst leicht auf eine vorhandene Architektur (gegeben durch Anzahl und Leistungsfähigkeit der Prozessoren, Leistungsfähigkeit des Kommunikationsnetzes, Art der Synchronisation) anzupassen.

Enthält jeder KKS mehrere Schritte, sprechen wir von "grober Granularität". Eine Form ist die "Komponentenparallelität", bei der ein KKS Schritte bezüglich einer Komponente enthält. Ein passendes System erlaubt, Komponenten parallel zu konfigurieren.[2] Eine andere Form ("Abhängigkeitsparallelität") faßt untereinander stark abhängige Schritte zu einem komplexen Schritt zusammen.

[2]Konfigurierungssysteme wie REDUX, die es erlauben mehrere Komponenten kooperativ zu konfigurieren, fallen in diese Kategorie.

Enthält jeder KKS genau einen Schritt, sprechen wir von "Schrittparallelität" oder "mittlerer Granularität". Ein passendes System erlaubt es, alle Schritte z.B. einer aktuellen Agenda unter Berücksichtigung der Voraussetzungen parallel zu bearbeiten.

Enthält jeder KKS genau einen Schritt und einen festen Wert (steht damit für genau *einen* Wert einer Eigenschaft, nicht mehr für die *Bestimmung* eines Wertes), sprechen wir von "Wertparallelität" oder "feiner Granularität". Ein passendes System erlaubt die parallele Verarbeitung der (alternativen) Werte von Eigenschaften. Dabei wird während der Konfigurierung für mehrere solcher Schritte parallel entschieden, ob sie noch gelten oder nicht. Aufgrund der meist großen Werteanzahl heißt ein solches System auch "massiv-paralleles" System. Voraussetzung ist die explizite Aufzählbarkeit der Werte.

Enthält jeder KKS Schritte, die mit dem gleichen Verfahren bearbeitet werden, sprechen wir von "Modulparallelität" oder "modularer Granularität". Ein passendes System erlaubt, Module, wie Spezialisierungsmodul, ATMS-Modul, Constraintmodul, parallel zueinander arbeiten zu lassen.

Neben diesen Parallelisierungsmöglichkeiten des Konfigurierungsvorgangs gibt es Varianten, die innerhalb eines Moduls Parallelverarbeitung durchführen. Dazu gehören etwa eine paralleles ATMS, eine parallele Breitensuche, die parallele Bearbeitung eines Constraint-Netzes oder die parallele Bearbeitung einer Beschreibungslogik ([5, 1]), falls eine solche zur Konfiguration herangezogen wird. Die Parallelisierung von Constraint-Netzen hängt von den zu verarbeitenden Wertebereichen der Variablen ab. Werden endliche Bereiche benutzt, kann z.B. ein massiv-paralleles Verfahren, wie in [3] beschrieben, zur Anwendung kommen. Sind solche Bereiche nicht ausreichend (wie in den meisten Anwendungen), ist es nur möglich, voneinander unabhängige Teilnetze eines Constraint-Netzes zu identifizieren (vgl. Abschnitt 4).

4 Merkmale paralleler Verarbeitung

Wovon hängt nun die Entscheidung ab, welche der beschriebenen Möglichkeiten bei einem Systementwurf gewählt wird? Kriterien hierbei sind: der im Problem inhärente Parallelisierungsgrad der Domäne (Granularität und Abhängigkeiten), die Art der Synchronisation (zentrale oder verteilte Synchronisation) und die Rechnerarchitektur, auf der das parallele Konfigurierungssystem ablaufen soll.

Abhängigkeiten Die Parallelisierbarkeit einer Domäne hängt im wesentlichen von den Abhängigkeiten der komplexen Schritte untereinander ab. Durch die Voraussetzungen der Schritte wird ein Abhängigkeitsnetz beschrieben, welches aus den KKS als Knoten besteht und für jede Voraussetzung eines komplexen Schritts (Knotens) eine Kante zu einem anderen Knoten aufweist. Da die Schritte und ihre Voraussetzungen auch die mehrstelligen Relationen beinhalten, werden durch diese Netzbildung unabhängige Teilnetze (z.B. gebildet durch mehrstellige Relationen, die nur einen Teil der Komponenten betreffen) erzeugt. Diese entsprechen den bei Constraint-Netzen zu beobachtenden Teilnetzen (vgl. Abbildung 1). Der Parallelisierungsgrad dieses Netzes (d.h. die Anzahl der Schritte, die parallel ausgeführt

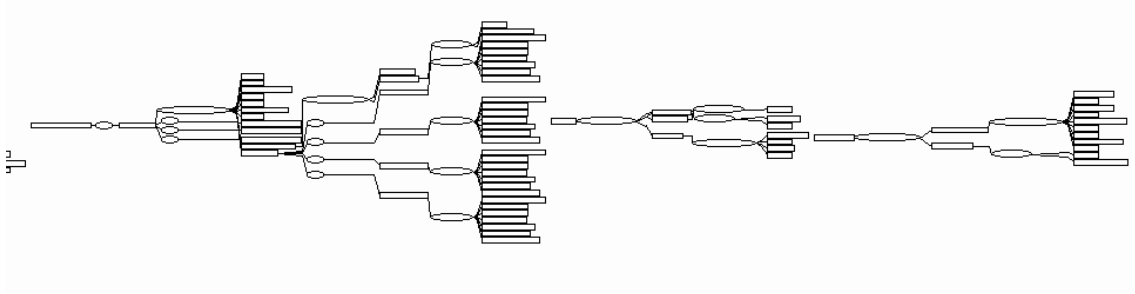

Abbildung 1: Ein Constraint-Netz, welches während einer Konfigurierung entstanden ist. Es sind drei unabhängige Teilnetze zu erkennen. Ellipsen beschreiben Relationen, Rechtecke Variablen.

werden können) hängt von seiner Struktur ab. Wird z.B. im Extremfall durch die Voraussetzungen eine sequentielle Reihenfolge der Schritte bestimmt, kann keinerlei Parallelarbeit erfolgen. Untersuchungen an bestehenden Konfigurierungsanwendungen zeigen jedoch, daß durchaus unabhängige Teile einer Konfiguration existieren.

Der Parallelisierungsgrad hängt weiterhin von den für die Eigenschaftsbestimmungen erforderlichen Berechnungen ab. Die Bearbeitung eines aufwendigen Knotens kann evtl. zu mehreren anderen schnell zu verarbeitenden Knoten parallel ablaufen (vgl. [1]).

Das Abhängigkeitsnetz ist jedoch nicht nur domänenspezifisch, sondern auch aufgabenspezifisch – zur Lösung einer Aufgabe braucht man etwa eine bestimmte Komponente X, zur Lösung einer anderen Aufgabe nicht. Welche Abhängigkeiten entstehen, kann erst vollständig während der Konfigurierung festgestellt werden, speziell die Verwendung der mehrstelligen Relationen bestimmt das Abhängigkeitsnetz. Werden etwa in einem Fall die Argumente a und b einer dreistelligen Relation zuerst berechnet, bedingen diese die Berechnung des dritten Arguments c. In einem anderen Fall werden b und c zuerst berechnet (oder sind durch die Aufgabenstellung gegeben) und bedingen a. Der Parallelisierungsgrad ist damit nicht vollständig aus einer Domäne im Vorwege ableitbar. Zur Lösung dieses Problems gibt es einmal die Möglichkeit, vollständig dynamisch, die entstehenden Aufgaben (d.h. die aus der Domäne abgeleiteten komplexen Schritte) auf Agenten zu verteilen. Dabei ist eine dynamische Abhängigkeitsanalyse notwendig. Dazu muß vor der Erzeugung eines Agenten[3] überprüft werden, ob die Voraussetzungen der Berechnung bereits vorliegen. Bei einer zentralen Synchronisation der komplexen Schritte ist diese Verwaltung einfacher und effizienter zu realisieren, als bei einer verteilten Synchronisation (s.u.).

Eine andere Lösung besteht in der teilweisen Vorausberechnung der obligatorisch entstehenden Abhängigkeiten, z.B. inferiert aus der Zerlegungsstruktur der Domäne. So ist es möglich, zumindest teilweise im Vorwege die notwendigen parallelen Resourcen zu allozieren, die Kommunikationspfade (gegeben durch die Abhängigkeiten) festzulegen und so, Teilaussagen über die parallele Lauffähigkeit der Problemlösung

[3]Wird ein Agent zuerst erzeugt und wartet dann auf die Erfüllung seiner Voraussetzungen, blockiert er die Abarbeitung anderer Agenten.

zu treffen. Untersuchungen aus dem Bereich der semantischen Netze liefern als Ergebnis, daß eine solche Erstellung eines Netzes und seine Abbildung auf Prozessoren effektiv realisiert werden kann (vgl. [1]).

Granularität Der Parallelisierungsgrad hängt u.a. von der gewählten Granularität ab. Wird z.B. die Komponentenparallelität gewählt, ist es nicht möglich unabhängige Eigenschaften einer Komponente parallel zu bestimmen. Die Wertparallelität ist auf der anderen Seite nur für endliche, explizit aufgezählte Wertemengen möglich. Um auch die Bestimmung von Eigenschaften über Funktionen zuzulassen und eine möglichst hohe Parallelität zu erreichen, erscheint die Schrittparallelität als geeignet. Die Wahl der Granularität hängt jedoch auch wesentlich mit der Rechnerarchitektur zusammen.

Verteilte vs. zentrale Synchronisation Ein weiteres Merkmal für die Verarbeitung der komplexer Schritte ist durch die Art der Synchronisation gegeben. Zentrale Synchronisation arbeitet nach dem Master/Slave Prinzip, während verteilte Synchronisation auf miteinander verhandelnden Agenten beruht.

Durch einen Steuerungsagenten (Masterprozess) werden bei zentraler Synchronisation an einer Stelle zentral die Agenten, ihre zugeordneten komplexen Schritte und die Abbildung auf eine Rechnerarchitektur verwaltet. Weiterhin können die Agenten synchron oder asynchron laufen. Bei synchroner Bearbeitung sorgt der Master für die Ausführung einer bestimmten Anzahl (in Abhängigkeit von den vorhandenen Ressourcen) komplexer Schritte und wartet bis alle Agenten ihren Schritt bearbeitet haben. So sind nicht parallelisierte (oder parallelisierbare) sequentielle Module, etwa die Propagation eines Constraintnetzes oder die Verwaltung eines ATMS, in die parallele Verarbeitung einpassbar. Bei asynchroner Bearbeitung der Agenten werden kontinuierlich die notwendigen komplexen Schritte an Agenten verteilt. Durch die zentrale Verwaltung können in jedem Fall die Voraussetzungen für die Berechnung eines komplexen Schritts ohne Kommunikationsaufwand zentral geprüft werden.

Bei verteilter Synchronisation sorgen die Agenten selbst für die Erzeugung weiterer Agenten, d.h. nach der Ausführung eines komplexen Schritts werden die sich daraus ergebenen weiteren Schritte an neue Agenten weitergereicht. Ein Zerlegungsschritt bei dem neue Komponenten erzeugt werden, verursacht z.B. die Notwendigkeit weiterer Eigenschaftsbestimmungen dieser neuen Komponenten. Agenten kommunizieren lediglich mit so erzeugten Nachfolgern oder Nachbaragenten (d.h. lokal), da keinerlei zentrale Information vorliegt.

Varianten paralleler Konfigurierungssysteme Wir schlagen die Verwendung der mittleren Granularität (Schrittparallelität) vor, da so der Parallelitätsgrad nur von den Abhängigkeiten der Schritte untereinander beeinflußt wird. Die Abhängigkeiten bilden ein Netz, welches statisch oder dynamisch auf Agenten verteilt werden kann. Aus obigen Überlegungen ergeben sich folgende mögliche Systemarchitekturen für ein paralleles Konfigurierungssystem in Abhängigkeit von der vorliegenden Rechnerarchitektur. Rechnerarchitekturen mit langsamen Kommunikationsnetzen erfordern eine Systemkonfiguration mit geringstem Kommunikationsaufwand. Für ein solches kann das Netz gemäß Struktur und Berechnungsaufwand der Knoten statisch auf die Agenten verteilt werden. Ein Agent könnte z.B. so alle Schritte

eines stark vernetzten Teilnetzes bearbeiten. Weiterhin ist eine zentrale Verwaltung der Schritte und ihrer Voraussetzungen hier vorzuziehen. Rechnerarchitekturen mit gemeinsamen Speicher bieten eine hohe Bandbreite und damit eine effektive Kommunikation. Für solche kann eine Synchronisation der Agenten über die Abhängigkeiten dynamisch erfolgen, d.h. die Abhängigkeiten synchronisieren den Konfigurierungsprozess. Durch einen mittleren Granularitätsgrad (pro Agent ein Schritt) wird eine hohe Parallelisierung, z.B. realisiert über eine große Prozessanzahl, möglich.

5 Zusammenfassung

Der Konfigurierungsvorgang in einem Konfigurierungssystem mit schrittweiser Entwicklung der Konfiguration kann durch parallele Ausführung einzelner Konfigurierungsschritte beschleunigt werden. Dabei bilden die Granularität (d.h. die Größe der zu parallelisierenden Aufgaben, hier "komplexe Schritte" genannt) und die Abhängigkeiten zwischen diesen Schritten die wesentlichen Faktoren bei der Parallelisierung. Die Abhängigkeiten der Eigenschaften untereinander bilden ein Netz, welches von der Domäne und der aktuellen Aufgabenstellung abhängt. Dieses Netz wird unter Berücksichtigung der Netzstruktur und der Berechnungszeit der einzelnen Knoten auf Prozessoren verteilt. Um einen möglichst hohen Grad an Parallelität zu gewinnen, schlagen wir als komplexen Schritt die Bearbeitung einer Eigenschaft einer Komponente vor. So werden etwaige unabhängige Eigenschaften z.B. innerhalb einer Komponente identifiziert.

Literatur

[1] V. Fischer. *Parallelverarbeitung in einem semantischen Netzwerk für die wissensbasierte Musteranalyse.* Infix, DISKI 95, Sankt Augustin, 1995.

[2] A. Günter (Hrsg.). *Wissensbasiertes Konfigurierung – Ergebnisse aus dem Projekt PROKON.* Infix, Sankt Augustin, 1995.

[3] H. W. Guesgen, K. Ho und P. N. Hilfinger. A Tagging Method for Parallel Constraint Satisfaction. *Journal of Parallel and Distributed Computing*, No. 16, S. 72-75, 1992.

[4] H. Kitano. Challenges of Massive Parallelism. *IJCAI'93*, Chambery, 813-831, 1993.

[5] E. R. Melz und R. M. Macgregor. *Design, Implementation, and Analysis of a Parallele Description Classifier.* Technischer Report, USC Information Sciences Institute, 1995.

[6] D. I. Moldovan. *Parallel Processing. From Applications to Systems.* Morgan Kaufmann, 1993.

[7] B. Neumann. Kommentierung und Wertung der PROKON-Ergebnisse. In: [2].

[8] J. Paulokat. Entscheidungsorientierte Rechtfertigungsverwaltung zur Unterstützung des Konfigurationsprozesses in IDAX. *Expertensystme*, M. Richter und F. Maurer (Hrsg.), infix, Sankt Augustin, 1995.