Westfälische Wilhelms-Universität Münster
Institut für Informatik
Seminar Softwareagenten
Wintersemester 2001 / 2002
Dozenten: Prof. Lippe, Dr. Lammers

SciAgents – Eine agentenbasierte Umgebung für verteilte wissenschaftliche Berechnungen

Gliederung

Alexander Starke, E-Mail: alstar@uni-muenster.de

Einführung

Wissenschaftliche Berechnungen verwenden numerische Modelle, welche, durch die Fortschritte in der Physik, immer komplexer werden. Ein realistisches Modell beinhaltet eine große Anzahl an kleineren Modellen, die Subsysteme des Gesamtsystems repräsentieren. Das Verhalten des Systems resultiert aus der Interaktion der Subsysteme. Ein Software-System für solche wissenschaftlichen Anwendungen reflektiert diese Komplexität. Da die Designer solcher komplexen Modelle und deren Anwender meistens Fachwissenschaftler sind, wird es für diese, mit zunehmender Verfeinerung der Soft- und Hardware, immer schwerer komplexe Softwaresysteme selbst zu entwickeln.
Es ist heutzutage bekannt, daß keine universellen Problemlöser für solche komplexen, heterogenen Modelle entwickelt werden können, sondern für jedes Subsystem eigene Lösungen entwickelt werden müssen. Wenn ein komplexes Modell in kleinere Modelle unterteilt werden kann und die Interaktionen mathematisch modelliert werden können, dann können einfachere, interagierende Problemlöser das Gesamtproblem lösen. Das im folgenden beschriebene agentenbasierte SciAgents-System verfolgt den oben beschriebenen Ansatz.
In der Softwareentwicklung haben Trends wie Abstraktion, Kapselung und Modularisierung zum Konzept der Agenten geführt. Ein System wird als Agent bezeichnet, wenn es folgende Eigenschaften besitzt:

- Autonomität
- Soziale Fähigkeiten
- Reaktivität
- Proaktivität

Zwei bedeutende Aspekte von Agenten sind ihre Modularität und Flexibilität. Ein agentenbasiertes System bietet natürliche Methoden der Dekomposition oder der Aggregation mehrerer Agenten, zur Lösung der einzelnen Subsysteme. Dieses führt zur verteilten Problemlösung in verteilten Systemen mit lokalen Problemlöser-Agenten. Die Aufgabe der Entspannung der Schnittstellen benachbarter Subsysteme fällt den Mediator-Agenten zu. Dabei wird die Fähigkeit der Agenten, selbstständig ihre Ziele zu verfolgen ausgenutzt, um, ohne Intervention des Anwenders, zu einer globalen Lösung zu kommen.

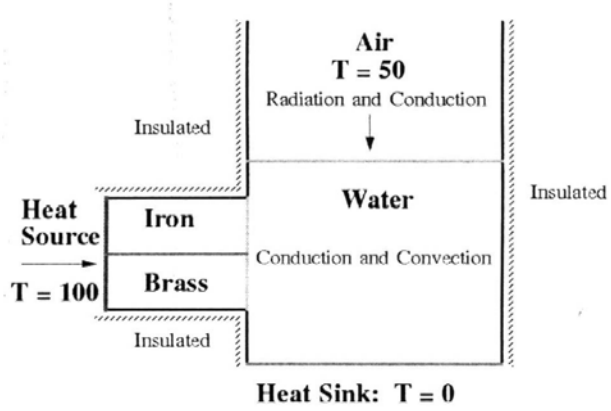
PDE-Modelle

SciAgents wurde entwickelt, um Probleme zu lösen, die auf partiellen Differentialgleichungen (partial differential equation, kurz: PDE) basieren. Viele physische Probleme werden durch solche PDE´s modelliert, wobei einfache Probleme aus einem einzigen Wertebereich und einer einzigen Differentialgleichung bestehen. Für solche „simplen“ Probleme wurden generelle (agentenbasierte) Lösungen entwickelt.
Viele Probleme der realen Welt sind aber zu komplex, um durch solche einfachen PDE´s modelliert zu werden, daher besteht ein Bedarf nach Lösungen für multiple-domain PDE´s. SciAgents wurde für genau solche Probleme entwickelt, allerdings müssen einige Voraussetzungen erfüllt sein:

- Das physische Problem besteht aus einfach verbundenen Teilen

- Jeder dieser Teile (Subsystem)kann durch ein einfaches PDE modelliert werden
- Die einzelnen Subsysteme interagieren dadurch, daß die Interfacebedingungen an den Grenzen der Subsysteme angepaßt werden

Ein einfaches Beispiel für ein solches Problem ist in Abbildung 1 dargestellt; hier wird die Temperaturverteilung in verschiedenen Substanzen (mit unterschiedlichen physikalischen Gesetzen) modelliert.



(Abb. 1 - Temperaturverteilung in verschiedenen Substanzen [1])

Das Problem ist nun, wie man von den lokalen Lösungen zum Gesamtergebnis kommt. Hiefür wird die Interface-Relaxation-Technik eingesetzt, welche die physikalischen Verbindungen der Subsysteme verwendet. An jeder Schnittstelle müssen somit bestimmte Bedingungen erfüllt sein, die durch die Interface-Relaxation-Technik eingehalten werden. Diese Technik sieht folgendermaßen aus:

1) Initialisiere jedes Subsystem mit den entsprechenden Startwerten
2) Löse das Gleichungssystem der Subdomäne und erhalte eine lokale Lösung
3) Untersuche, wie gut die jeweiligen Interface-Bedingungen erfüllt sind und berechne neue Werte für die Schnittstellen
4) Iteriere 1) bis 3) bis Konvergenz erreicht ist

SciAgents aus der Sicht des Benutzers

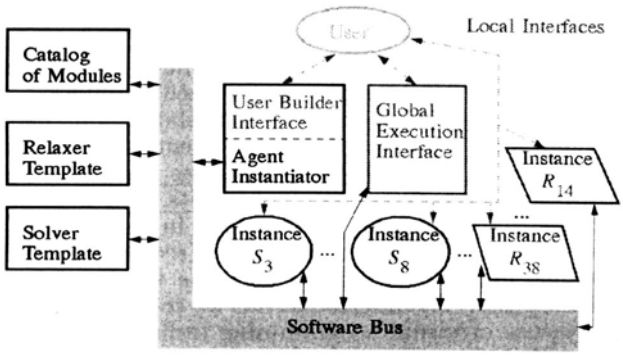
Nachdem die PDE´s mit den zugehörigen Schnittstellen festgelegt wurden, muß ein Netzwerk aus kooperierenden Agenten aufgebaut werden, die lokale Berechnungen ausführen und durch den Austausch von Daten mit anderen Agenten und dem Anwender zur Gesamtlösung kommen. Wie schon erwähnt gibt es zwei Arten von Agenten: Die „Solver“, die für die Lösung der lokalen Berechnungen zuständig sind und die „Relaxer“, die sich um die Zusammenarbeit der einzelnen Solver kümmern. Die Relaxer sorgen durch die Entspannung der Interface-Bedingungen für die globale Konvergenz der Algorithmen. Jeder Relaxer kontrolliert eine Schnittstelle zwischen zwei Subdomänen und jeder Solver ist für eine solche Domäne zuständig.
Durch die Kapselung der Software-Architektur wird der Anwender nur mit den, für die Definition des Problems und den Lösungsprozeß wichtigen Details konfrontiert (also nicht mit informatischen Details, was Nutzern ohne informatischen Background die Nutzung wesentlich erleichtert).
Jeder Solver-Agent bietet dem Anwender folgende Konfigurationsmöglichkeiten:

- Definition der Wertebereiche
- Definition des Gleichungssystems
- Visualisierungsoptionen
- Auswahl der Parameter

Der Nutzer kann für die Relaxer-Agenten die folgenden Optionen wählen:

- Beschreibung der Verbindung
- Interface-Bedingungen
- Schema der Schnittstellen-Entspannung
- Algorithmen für die Approximation der Werte
- Visualisierungsoptionen

Außerdem kann der User verschiedene Level der Interaktion zwischen sich und dem System wählen.
Um ein solches Netzwerk von kooperierenden Agenten aufzubauen, werden dem Benutzer nur Agenten-Vorlagen präsentiert, die zwar Informationen über die Solver und Relaxer enthalten, allerdings noch nichts „berechnen“ können. Diese Agenten-Vorlagen müssen durch den „Agent-Instantiator“ erst instantiiert werden. Wenn ein Agent erschaffen wurde, beginnt er mit der Kommunikation mit dem User und seiner Umwelt, um sich alle wichtigen Informationen, die er zur Erledigung seiner Aufgabe benötigt, zu beschaffen. Zum Beispiel versucht jeder Relaxer, nachdem er mit den Verbindungsinformationen versorgt wurde, seine zugehörigen Solver zu kontaktieren, damit er die entsprechenden Informationen vervollständigen kann (z.B. die Geometrie der Schnittstelle, die Visualisierungsmöglichkeiten des Solvers, etc.). Abbildung 2 zeigt das SciAgents-System aus der Sicht des Anwenders: Alle Objekte kommunizieren miteinander über den Software-Bus; der Benutzer kommuniziert über das User-Builder-Interface mit dem Agent-Instantiator, welcher wiederum über den Software-Bus die Agenten-Vorlagen kontaktiert; Agenten haben selbst ein eigenes User-Interface, mit dem sie Informationen vom Nutzer einholen können.



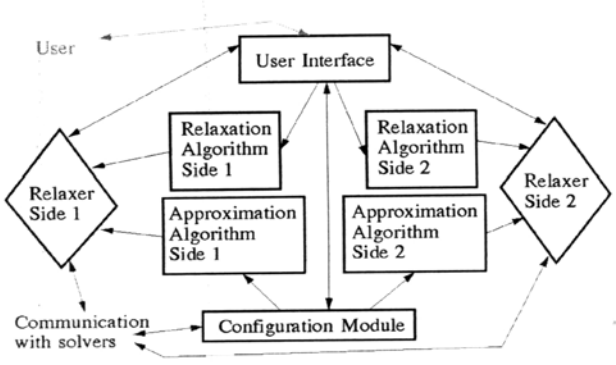
(Abb. 2 - SciAgents aus der Sicht des Nutzers [1])

Ein Interessanter Effekt der agentenbasierten Technologie ist die Art der „Synchronisation“ des Systems: Die Reihenfolge, in der die Agenten instantiiert werden ist völlig irrelevant, denn wenn beispielsweise ein Solver noch nicht alle Grenzwerte für seine Berechnungen erhalten hat (weil der zugehörige Relaxer noch fehlt), dann unterbricht er seine Arbeit und wartet auf den Relaxer, der ihn mit seinen noch fehlenden Werten versorgt. Da die Informationen, die bei der Erzeugung eines Agenten eingegeben werden, immer lokale Informationen eines Subsystems sind, muß der jeweilige Anwender nicht das gesamte Problem im Überblick haben, was es

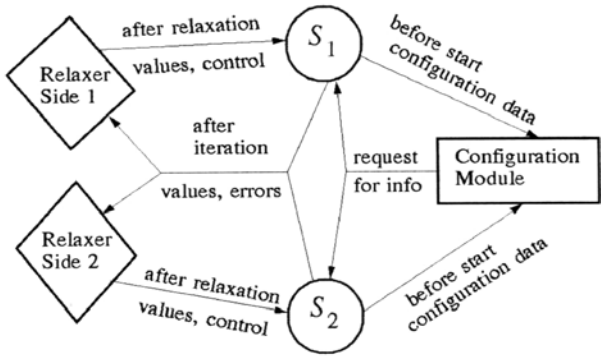
mehreren Spezialisten ermöglicht, zusammen an einem großen Problem zu arbeiten.
Der User muß den jeweiligen Agent nur mit den funktionalen Parametern versorgen, die restlichen internen Parameter besorgt sich der Agent selbst auf eine „intelligente“ Weise. Hierfür sollen drei repräsentative Beispiele gegeben werden: Ein lokaler Solver benötigt viele weitere, die internen Berechnungen betreffende Parameter (z.B. Diskretisierungsmethode, Gittergröße und –Konfiguration usw.); der Relaxer muß Algorithmen zur Interface-Entspannung auswählen; das Global Execution Interface muß die Verteilung der Agenten auf die vorhandene Hardware organisieren. Bei Entscheidungen der oben beschriebenen Art, verwendet SciAgents ein anderes agentenbasiertes System namens PYTHIA. Die Wahl der empfohlenen Parameter trifft PYTHIA auf der Basis von ermittelten Performanz-Daten unterschiedlicher Konfigurationen und Lösungsmethoden. [vgl. 2] Ein Nachteil dieser Entscheidungsstrategie ist der, daß immer eine gewisse „Erfahrung“ vorausgesetzt wird. Dieser Nachteil wird allerdings durch den agentenbasierten Ansatz des PYTHIA-Systems minimiert; denn wenn jeder PYTHIA-Agent verschiedene (aber ggf. ähnliche) Probleme löst, kann jeder Agent auf einen gewissen Erfahrungsschatz zurückgreifen, auf den auch andere Agenten zurückgreifen können. Auf diese Weise können die einzelnen Agenten ihre Informationen selbstständig vervollständigen.

Softwarearchitektur von SciAgents

Die lokalen Solver-Agenten bestehen aus einem Kern, der die Funktionalität des PDE implementiert, einem User Interface und einem Wrapper, welcher dem Solver das Aussehen und das Verhalten eines Agenten gibt. Dabei können die Kern-Implementierungen leicht angepaßt, ausgetauscht oder neue Agenten-Templates zum Agent-Instantiator hinzugefügt werden, da SciAgents als offenes System designed wurde.
Die Architektur der Relaxer unterstützt die Verteilung der Berechnungen und eine effiziente Implementierung. Da die Schnittstellen-Bedingungen auf den beiden Seiten des Interfaces sehr unterschiedlich sein können (z.B. verschiedene Entspannungs-Schemata, andere Approximationsalgorithmen usw.), wurden die Relaxer in zwei Subrelaxer aufgeteilt. Jeder Subrelaxer kontrolliert einen Solver mit eigenen Entspannungs- und Approximationsalgorithmen. Das gemeinsame User-Interface läßt die beiden Subrelaxer allerdings als eine Entität erscheinen. Das gemeinsame Konfigurationsmodul ist für die Orientierung des Agenten in seiner Umwelt verantwortlich, d.h. nach der Instantiierung sucht dieses Modul die zugehörigen Solver und initialisiert diese; außerdem überwacht es die Subrelaxer und terminiert den Prozeß, sobald Konvergenz erreicht ist. Abbildung 3 zeigt die Software-Architektur eines Relaxers. Abbildung 4 verdeutlicht den Informationsfluß zwischen den Subrelaxern, den Solvern und dem Konfigurationsmodul; zur High-Level-Kommunikation wird die Knowledge-Query-and-Manipulation-Language KQML benutzt.



(Abb. 3 – Architektur eines Relaxers [4])



(Abb. 4 – Kommunikation zwischen Agenten [4])

Die beschriebene Architektur ermöglicht eine ökonomische Verteilung der Agenten auf die vorhandenen Hardware-Ressourcen.
Die lokalen Berechnungen werden von einem Relaxer kontrolliert. Diese werten nach jedem Berechnungsdurchlauf die Ergebnisse bzw. die Fehler aus und setzen den lokalen Prozeß aus, d.h. sie veranlassen die Solver mit den gleichen Randbedingungen wie im vorherigen Durchlauf zu rechnen, wenn eine bestimmte Genauigkeit erreicht wurde. Wenn ein Solver von allen zugehörigen Relaxern die o.g. Order bekommen hat, stoppt dieser seine Berechnungen. Ein anderer Grund zum Abbruch des Prozesses ist das Erreichen der maximalen Anzahl an Durchläufen; dann wird ein Fehler an das Global Execution Interface gemeldet und der gesamte Rechenprozeß terminiert.

Zusammenfassung

Der modulare, agentenbasierte Aufbau des SciAgent-Systems bietet größtmögliche Flexibilität und hilft die enorme Komplexität wissenschaftlicher Probleme zu bewältigen. Ein besonderer Vorteil von SciAgents liegt in der Kapselung der Implementierungsdetails; der Fachanwender wird lediglich mit ihm bekannten Details konfrontiert, denn die Agenten nehmen ihm Entscheidungen, die außerhalb seines Fachwissens liegen ab.

Literatur

[1] T. Drashansky, A. Joshi and J. Rice, *SciAgents – An Agent Based Environment for Distributed, Cooperative Scientific Computing.* Technical Report TR-95-029, Dept. Comp. Sci., Purdue University, 1995.

[2] A. Joshi, *To Learn or Not to Learn …*, Proc. IJCAI´95 Workshop on Adaption and Learning in Multiagent Systems

[3] A. Joshi, T. Drashansky, J. Rice, S. Weerawarana and E. Houstis, *On Learning and Adaption in Multiagent Systems: A Scientific Computing Perspective*.

[4] A. Joshi, T. Drashansky, J. Rice, S. Weerawarana and E. Houstis, *Multi-Agent Simulation of Complex Heterogeneous Models in Scientific Computing.*