

Teilprojekt

B8

Parallelisierung irregulärer numerischer Algorithmen

2.1 Teilprojekt B8

Parallelisierung irregulerer numerischer Algorithmen

2.1.1 Antragsteller

Prof. Dr. Gudula Runger
08.08.1959
Professur Praktische Informatik
Fakultat fur Informatik
Technische Universitat Chemnitz
09107 Chemnitz
Tel.: (0371) 531-1794
Fax: (0371) 531-1803
ruenger@informatik.tu-chemnitz.de

2.1.2 Projektbearbeiter

Dipl. Inf. Judith Hippold, Professur Praktische Informatik, Projektstelle seit 01/2002
Dipl. Inf. Robert Reilein-Ru, Professur Praktische Informatik, Grundausrattung
Dr. Klaus Hering, Professur Praktische Informatik, Grundausrattung bis 10/2002

2.2 Ausgangsfragestellung / Einleitung

Gegenstand des Teilprojektes ist die effiziente parallele Realisierung irregulerer Algorithmen auf Rechnern mit verteiltem Speicher, Clustern oder Clustern von SMPs (symmetric multiprocessor). Die Klasse der irreguleren Algorithmen umfasst dabei Probleme mit unregelmaigen oder laufzeitabhangigen Berechnungs- und Kontrollstrukturen, fur die Standardparallelisierungstechniken, wie datenparallele oder SPMD-Abarbeitung, nicht die gewunschte Effizienz und Skalierbarkeit aufweisen oder die sich einer solchen regelmaigen Parallelisierung ganz entziehen. Grunde fur irregulares Verhalten sind vielfaltig und umfassen dunn-besetzte, blockstrukturierte, adaptive oder hierarchische Anwendungsprobleme und Algorithmen mit sehr unterschiedlichen dynamischen Eigenschaften der zu Grunde liegenden Daten- und Berechnungsstrukturen. Entsprechend der Auspragung der Irregularitat variieren auch die Parallelisierungsmethoden fur die jeweilige Anwendungsklasse, die vom Einsatz alternativer Programmiermodelle mit planbaren Eigenschaften bis hin zu vollstandig dynamischen Abarbeitungskonzepten zur Laufzeit reichen.

Ziel des Projektes war es, die Parallelisierung irregulerer Algorithmen insbesondere hinsichtlich der Moglichkeiten zur effizienten Realisierung der laufzeitabhangigen Komponenten auf Rechnern mit verteiltem Adressraum zu untersuchen und hierbei die Aspekte geeigneter Datenstrukturen und Analysefunktionen, den Einsatz von Task Pools fur das Scheduling und Lastverteilung sowie Kommunikationsoptimierung schwerpunktmaig zu berucksichtigen. Neben der Entwicklung dieser Konzepte und ihrer Einbettung in die jeweilige Anwendung sollten Methoden entwickelt werden, die Eigenschaften eines speziellen Anwendungsalgorithmus zur Effizienzverbesserung in die parallele Softwareerstellung erganzend einbringen konnen. Als Anwendungsalgorithmen sollten das hierar-

chische Radiosity-Verfahren, adaptive gitterbasierte Losungsverfahren fur partielle Differentialgleichungen, strukturiert irregulare Probleme sowie Berechnungen im Zusammenhang mit nichtlinearen dynamischen Systemen untersucht werden, wobei letztere als Kooperationen mit anderen Teilprojekten angesiedelt sind. Die Untersuchung dieser Anwendungsklassen mundet in die Teilaufgaben:

- Task Pool Teams zur Parallelisierung hierarchischer Algorithmen,
- Daten- und Kommunikationsschicht fur adaptive Algorithmen,
- Kommunikationsbibliothek fur orthogonale Prozessorgruppen,
- Bibliotheksunterstutzung fur hierarchische Multi-Prozessor Tasks,
- Parallelisierung im Bereich nichtlinearer dynamischer Systeme.

2.3 Forschungsaufgaben / Methoden

2.3.1 Task Pool Teams zur Parallelisierung hierarchischer Algorithmen

Zur Parallelisierung hierarchischer Algorithmen oder baumartiger Branch & Bound Algorithmen wurde das Konzept der Task Pool Teams entwickelt, modular realisiert und auf verschiedene Anwendungsalgorithmen und parallele Plattformen angewendet. Als konkretes Beispiel wurde insbesondere der hierarchische Radiosity Algorithmus, ein globales Beleuchtungsverfahren der Computergrafik zur Simulation der Beleuchtung 3-dimensionaler Szenen in geschlossenen Rumen, betrachtet. Aufgrund des hierarchischen Berechnungsansatzes erfolgt eine unregelmaige Verfeinerung der zu Grunde liegenden Datenstrukturen, die stark von der Eingabeszene des Algorithmus abhangt und eine parallele Umsetzung erschwert. Fur Architekturen mit gemeinsamem Speicher sind bereits parallele Implementierungen erfolgt [WOT⁺95, KR02, PRR98, KR04]. Dabei wurde der Algorithmus in Berechnungsaufgaben, so genannte Tasks, unterteilt und diese mit Task Pools [But97, RR00c, SHT⁺95] abgearbeitet. Task Pools werden zur dynamischen Rebalancierung der Last eingesetzt. Die Task Pool Datenstruktur dient hierbei zum Speichern und Verwalten der Tasks. Eine feste Anzahl von Threads kann aus dem Pool Tasks entnehmen und dynamisch neue Tasks einstellen. Bei entsprechendem Scheduling der Tasks auf die Prozessoren erfolgt eine effiziente Auslastung, da nach Abarbeitung eines Tasks automatisch ein neuer Task aus dem Pool entnommen wird. Je nach Anwendungsproblem konnen verschiedene interne Umsetzungen der Task Pools von Nutzen sein. Gangig sind Pools mit einer zentralen Taskqueue aber auch mit mehreren Taskqueues. Vorteilhaft beim Einsatz mehrerer Taskqueues sind die fehlenden Zugriffskonflikte durch gleichzeitig zugreifende Threads. Um hier jedoch Lastbalancierung zu erreichen, sind zusatzliche Mechanismen, z. B. Task Stealing, bei dem nach dem Leerlaufen der eigenen Taskqueue ein Thread versucht aus anderen Taskqueues Tasks zu stehlen, notwendig. Ein detaillierter Leistungsvergleich verschiedener Implementierungen ist in [KR02, KR04] zu finden.

Eine Implementierung fur Rechner mit verteiltem Speicher gestaltet sich komplexer, da die dynamischen Datenzugriffsmuster variieren und auch Zugriffe auf Daten im Speicherbereich anderer Prozessoren nach sich ziehen konnen, was zu irregularen Kommunikationsmustern fuhrt. Fur hybride Architekturen, wie z. B. SMP-Cluster, erscheint daher ein hybrides Programmiermodell geeignet, bei dem die Kommunikation mittels Message-Passing nur zwischen den Clusterknoten erfolgt, SMP-intern jedoch mehrere Threads

verschiedene Tasks uber derselben Datenmenge bearbeiten. Damit ergibt sich die Anforderung der Realisierung von multi-threaded, asynchronen Kommunikationsroutinen und zusatzlichen, geeigneten, verteilten Verwaltungsdatenstrukturen fur die effiziente Identifikation von Daten in entfernten (remoten) Speicherbereichen.

2.3.2 Daten- und Kommunikationsschicht fur adaptive Algorithmen

Adaptive Probleme zeichnen sich durch sich dynamisch verfeinernde, unregelmaige Gitterstrukturen aus, auf denen Berechnungen ausgefuhrt werden, die die dynamische Verfeinerung hervorrufen. Ein typisches Anwendungsbeispiel sind adaptive Finite Element Methoden (FEM). Eine Haupt-Herausforderung der parallelen Realisierung adaptiver gitterbasierter Losungsverfahren fur partielle Differentialgleichungen bildet die durch eine Verteilung der adaptiven Gitterstrukturen auf die Prozessoren notwendige Organisation eines effizienten Datentransfers. Datenstrukturbestandteile des Gitters, die sich (bzw. deren uber- oder untergeordnete Strukturen) nach einer Verteilung im Speicherbereich verschiedener Prozessoren befinden, erfordern zumeist eine global einheitliche Sicht, variieren aber gleichzeitig durch die adaptive Anpassung des Gitters in den lokalen Speichern. Als besonders kostenintensiv haben sich hierbei komplex strukturierte Gitter in drei Dimensionen mit hangenden Knoten herausgestellt. Hier sind fur die resultierende unregelmaige Kommunikation geeignete Kommunikationsprotokolle und geeignete verteilte Verwaltungsdatenstrukturen notwendig, die die aufkommenden Kommunikationsanforderungen realisieren, verringern und optimieren. Darauf aufbauend werden Mechanismen zur Lastbalancierung benotigt, die eine Umverteilung der Daten berechnen und vornehmen. Ersteres kann auf das NP-vollstandige Graphpartitionierungsproblem zuruckgefuhrt werden, fur das eine Reihe von Algorithmen [KL70, PSW93, PSL90, KK99, KK98, HL93a, SKK00] und Werkzeuge [KK95, HL93b, KSK02] existieren. Die tatsachliche effiziente Umverteilung anhand der neu berechneten Partitionen liegt dann meist in der Verantwortung des Anwendungsprogrammierers.

Mit Hinblick auf die variable Einsetzbarkeit adaptiver gitterbasierter Losungsverfahren bietet ein gekapselt implementierter Losungsansatz oben genannter Probleme erhebliche Vorteile. So konnen Erweiterungen und Optimierungen leichter eingebracht, Plattformunabhangigkeit erzielt und der Anwendungsprogrammierer von der Realisierung effizienter Kommunikationsmechanismen und verteilter Datenstrukturen entlastet werden. Systeme mit Unterstutzung irregularer Anwendungen durch Aufsetzen eines globalen Adressraummodells sind beispielsweise Titanium [YSP⁺98] oder TreadMarks [ACD⁺96]. Die Sicht auf einen verteilten Adressraum fur den Anwender zu erhalten, bietet jedoch durchaus Vorteile, da vom Anwendungsprogrammierer anwendungs-spezifische Details und Optimierungen speziell fur verteilte Berechnungen eingebracht werden konnen. Zudem existiert meist bereits eine sequentielle Programmversion, die mit nicht zu umfangreichen Modifizierungen auf verteilten Speicher portiert werden kann.

2.3.3 Kommunikationsbibliothek fur orthogonale Prozessorgruppen

Obwohl die parallele Programmierung im SPMD-Stil mit Standardkommunikationsbibliotheken wie MPI oder PVM fur unregelmaige Anwendungsprobleme haufig zu guten Parallelisierungsergebnissen auf Rechnern mit verteiltem Speicher fuhrt, kann dies auf

Maschinen mit sehr groer Prozessoranzahl Skalierbarkeits- und Effizienzprobleme verursachen, insbesondere wenn kollektive Kommunikationsoperationen eingesetzt werden, etwa fur Konvergenztests oder zur Akkumulation lokaler Teilergebnisse. Ein Grund ist die Ausfuhrungszeit kollektiver Kommunikationsoperationen, die logarithmisch oder linear in der Anzahl der teilnehmenden Prozesse steigt, so dass Kommunikation auf kleineren Teilgruppen von Prozessoren ausgefuhrt werden sollte, wenn dies aus algorithmischen Grunden moglich ist. Eine solche algorithmische Eigenschaft weisen Algorithmen mit zwei- oder hoher-dimensionalen Taskgittern auf, in denen Berechnungs- und Kommunikationsphasen der Abarbeitung so strukturiert sind, dass Tasks jeweils nur mit Tasks innerhalb eines (oder weniger) Teilgitters kooperieren und kommunizieren. Hier kann es zu erheblichen Effizienzgewinnen kommen, wenn die Zuordnung von Tasks zu Prozessoren diese Eigenschaft durch Kommunikation auf Teilgruppen von Prozessoren widerspiegelt [RR00b]. Das Programmiermodell der Orthogonalen Prozessorgruppen stellt ein zwei- oder hoher-dimensionales Prozessorgitter bereit, fur das eine feste Anzahl von Prozessorpartitionen betrachtet wird, die jeweils disjunkten mehrdimensionalen Teilgittern entsprechen. Ein entsprechendes Mapping von Tasks zu Prozessoren fuhrt so zu Berechnungs- und Kommunikationsphasen auf den alternativ zur Verfugung stehenden Partitionen im Gruppen-SPMD-Stil. Zur Programmierung solcher Programmstrukturen wird eine komfortable Programmierumgebung benotigt, die einerseits eine leichte Spezifikation der Task-, Partitions- und Mappingstruktur erlaubt und andererseits die benotigten Gruppen, Kommunikatoren und Zuordnungen im Hintergrund effizient aufbaut und verwaltet.

2.3.4 Bibliotheksunterstutzung fur hierarchische Multi-Prozessor Tasks

Multi-Prozessor-Tasks (M-Task) bezeichnen Teilaufgaben eines Anwendungsalgorithmus, die jeweils auf mehreren Prozessoren ausgefuhrt werden konnen. Der gesamte Algorithmus kann aus einer Menge von solchen Multi-Prozessor-Tasks bestehen, die miteinander kooperieren und so die Gesamtstruktur des Programms bestimmen. Die M-Task-Struktur kann gewisse unregelmaige Strukturen aufweisen und spiegelt modulare Eigenschaften der Anwendung wieder, die in einen M-Task-Graphen mit Abhangigkeiten dargestellt werden konnen. Solche, auch als strukturiert irregular bezeichnete, Strukturen finden sich in groen gekoppelten Anwendungsalgorithmen, aber auch in neuen parallelen Losungsverfahren fur gewohnliche Differentialgleichungssysteme oder in grobkornigen hierarchischen Algorithmen wie der Strassen-Multiplikation. Je nach Anwendung liegt eine M-Task-Struktur statisch fest und kann durch Scheduling und Laufzeitvorhersagemechanismen zur Planung einer effizienten Realisierung genutzt werden, wobei auch hier Skalierbarkeit und Effizienz durch Prozessorteilgruppen erreicht werden konnen. Zur Programmierung werden eine Reihe von Ansatzen vorgestellt, u. a. Fx [SSOG93, SY97], Paradigm [BCG⁺95, RSB97], Braid, Opus und Orca [BH98]. Einen Uberblick gibt [BH98]. Die Ansatze haben spezifische integrierte Kostenmodelle mit deren Hilfe geeignete Implementierungen gewahlt werden konnen. Dabei konnen Schedulingalgorithmen fur Multiprozessor-Task-Scheduling genutzt werden [TWY92, TLW⁺94, RR96, RR00a].

Eine weitere Herausforderung ist die dynamische Entstehung von M-Task-Strukturen, beispielsweise bei hierarchischen Algorithmen. Hierfur muss die Kreierung und Abarbei-

tung von M-Tasks mit Abhangigkeiten zur Laufzeit moglich sein, was die dynamische Kreierung von Prozessorteilgruppen und Kommunikationsstrukturen einschliet.

2.3.5 Parallelisierung im Bereich nichtlinearer dynamischer Systeme

In Kooperation mit Teilprojekt C8 wurde die Parallelisierung eines sequentiell vorliegenden Programms zur Bestimmung von Lyapunovexponenten und -vektoren in groen nichtlinearen dynamischen Systemen betrachtet. Die grobe Struktur des Anwendungsprogramms besteht in der simultanen Integration zweier groer Systeme von linearen bzw. nichtlinearen gewohnlichen Differentialgleichungen, deren Integrationszeitschritte periodisch durch notwendige Reorthogonalisierungen der Vektoren unterbrochen wird. Genau dieser Reorthogonalisierungsschritt stellt im sequentiellen Programm den zeitaufwendigsten Anteil dar, so dass die Ausnutzung paralleler Abarbeitung insbesondere im Hinblick der Losung groerer Probleme wesentlich ist. Die Aufgaben der Uberfuhrung in eine effiziente parallele Bearbeitung umfassen die programmtechnische Analyse des sequentiellen Algorithmus sowie die darauf aufbauende modulare Parallelisierung, was die Reorthogonalisierung selber als auch die parallele Schnittstelle zum Integrationsteil, also die periodisch wiederkehrende Kopplung verschiedener Programmierparadigmen, beinhaltet. Wichtiger Aspekt ist hierbei die Lastbalancierung im Integrationsteil, die Parallelisierungsstrategie der Reorthogonalisierung und die moglicherweise mit Kommunikation verbundene Gestaltung der Schnittstelle gegeneinander abzuwagen, um zu einer persistenten Leistung zu gelangen.

Literaturverzeichnis zu 2.3 (eigene Vorarbeiten und Fremdliteratur)

- [ACD⁺96] C. Amza, A. L. Cox, S. Dwarkadas, P. Keleher, H. Lu, R. Rajamony, W. Yu, and W. Zwaenepoel. TreadMarks: Shared Memory Computing on Networks of Workstations. *IEEE Computer*, 29(2):18–28, 1996.
- [BCG⁺95] P. Banerjee, J. Chandy, M. Gupta, E. Hodge, J. Holm, A. Lain, D. Palermo, S. Ramaswamy, and E. Su. The Paradigm Compiler for Distributed-Memory Multicomputers. *IEEE Computer*, 28(10):37–47, 1995.
- [BH98] H. Bal and M. Haines. Approaches for Integrating Task and Data Parallelism. *IEEE Concurrency*, 6(3):74–84, 1998.
- [But97] D. R. Butenhof. *Programming with POSIX Threads*. Addison-Wesley, 1997.
- [HL93a] B. Hendrickson and R. Leland. A Multilevel Algorithm for Partitioning Graphs. Technical Report 93-1301, Sandia National Lab, Albuquerque, NM, 1993.
- [HL93b] B. Hendrickson and R. Leland. The CHACO User’s Guide. Technical Report 93-2339, Sandia National Lab, Albuquerque, NM, 1993.
- [KK95] G. Karypis and V. Kumar. METIS Unstructured Graph Partitioning and Sparse Matrix Ordering System. Technical Report <http://www.cs.umn.edu/metis>, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1995.
- [KK98] G. Karypis and V. Kumar. Multilevel k-way Partitioning Scheme for Irregular Graphs. *Journal of Parallel & Distributed Computing*, 48:96–129, 1998.

- [KK99] G. Karypis and V. Kumar. A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal of Scientific Computing*, 20(1):359–392, 1999.
- [KL70] B. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 29:291–307, 1970.
- [KR02] M. Korch and T. Rauber. Evaluation of Task Pools for the Implementation of Parallel Irregular Algorithms. In *Proc. of ICPP: Workshop on Compile & Runtime Techniques for Parallel Computing (CRTPC02)*, pages 597–604, Vancouver, Canada, 2002.
- [KR04] M. Korch and T. Rauber. A Comparison of Task Pools for Dynamic Load Balancing of Irregular Algorithms. *Concurrency and Computation: Practice and Experience*, 16(1):1–47, 2004.
- [KSK02] G. Karypis, K. Schloegel, and V. Kumar. ParMetis, Parallel Graph Partitioning and Sparse Matrix Ordering Library, Version 3.0. Technical report, University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN, 2002.
- [PRR98] A. Podehl, T. Rauber, and G. Runger. A Shared-Memory Implementation of the Hierarchical Radiosity Method. *Theoretical Computer Science*, 196(1-2):215–240, 1998.
- [PSL90] A. Pothen, H. D. Simon, and K. P. Liou. Partitioning Sparse Matrices with Eigenvectors of Graphs. *SIAM Journal on Matrix Analysis and Applications*, 11:430–452, 1990.
- [PSW93] A. Pothen, H. D. Simon, and L. Wang. Spectral Nested Dissection. Technical Report CS-92-01, Computer Science, Pennsylvania State University, University Park, PA, 1993.
- [RR96] T. Rauber and G. Runger. The Compiler TwoL for the Design of Parallel Implementations. In *Proc. of the 4th Int. Conf. on Parallel Architectures & Compilation Techniques (PACT96)*, pages 292–301, Boston, MA, 1996.
- [RR00a] T. Rauber and G. Runger. A Transformation Approach to Derive Efficient Parallel Implementations. *IEEE Transactions on Software Engineering*, 26(4):315–339, 2000.
- [RR00b] T. Rauber and G. Runger. Deriving Array Distributions by Optimization Techniques. *Journal of Supercomputing*, 15(3):271–293, 2000.
- [RR00c] T. Rauber and G. Runger. *Parallele und Verteilte Programmierung*. Springer, 2000.
- [RSB97] S. Ramaswamy, S. Sapatnekar, and P. Banerjee. A Framework for Exploiting Task and Data Parallelism on Distributed-Memory Multicomputers. *IEEE Transactions on Parallel & Distributed Systems*, 8(11):1098–1116, 1997.
- [SHT⁺95] J. P. Singh, C. Holt, T. Totsuka, A. Gupta, and J. Hennessy. Load Balancing and Data Locality in Adaptive Hierarchical N-body Methods: Barnes-Hut, Fast Multipole, and Radiosity. *Journal of Parallel & Distributed Computing*, 27(2):118–141, 1995.

- [SKK00] K. Schloegel, G. Karypis, and V. Kumar. A Unified Algorithm for Load-balancing Adaptive Scientific Simulation. Technical Report 00-033, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2000.
- [SSOG93] J. Subhlok, J. Stichnoth, D. O'Hallaron, and T. Gross. Exploiting Task and Data Parallelism on a Multicomputer. In *Proc. of the 4th ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming (PPOPP93)*, pages 13–22, San Diego, CA, 1993.
- [SY97] J. Subhlok and B. Yang. A New Model for Integrating Nested Task and Data Parallel Programming. In *Proc. of the 6th ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming (PPOPP97)*, pages 1–12, Las Vegas, Nevada, 1997.
- [TLW⁺94] J. Turek, W. Ludwig, J. Wolf, L. Fleischer, P. Tiwari, J. Glasgow, U. Schwiegels-hohn, and P. Yu. Scheduling Parallelizable Tasks to Minimize Average Response Time. In *Proc. of the 6th ACM Symposium on Parallel Algorithms & Architecture (SPAA94)*, pages 200–209, Cape May, New Jersey, 1994.
- [TWY92] J. Turek, J. L. Wolf, and P. S. Yu. Approximate Algorithms for Scheduling Parallelizable Tasks. In *Proc. of the 4th ACM Symposium on Parallel Algorithms & Architecture (SPAA92)*, pages 323–332, San Diego, CA, 1992.
- [WOT⁺95] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proc. of the 22nd Annual Int. Symposium on Computer Architecture*, pages 24–36, 1995.
- [YSP⁺98] K. A. Yelick, L. Semenzato, G. Pike, C. Miyamoto, B. Liblit, A. Krishnamurthy, P. Hilfinger, S. L. Graham, D. Gay, P. Colella, and A. Aiken. Titanium: A High-Performance Java Dialect. *Concurrency: Practice and Experience*, 10(11–13):873–877, 1998.

2.4 Ergebnisse

2.4.1 Task Pool Teams zur Parallelisierung hierarchischer Algorithmen

Zur parallelen Abarbeitung des hierarchischen Radiosity Verfahrens und ahnlicher Algorithmen wurde das hybride Programmiermodell der Task Pool Teams entwickelt. Task Pool Teams [HR03a, Hip01] stellen ein Programmierkonzept fur verteilten Adressraum bzw. hybriden Adressraum dar und konnen als eine Erweiterung und Verallgemeinerung des Task Pool Konzeptes verstanden werden. Dem Nutzer werden Task Pool Teams uber eine einfache Schnittstelle zur Verfugung gestellt. Das hybride Programmiermodell [HR] kombiniert Shared-Memory Programmierung mittels von mit Pthreads realisierten Task Pools und Message-Passing Programmierung durch bereitgestellte asynchrone Kommunikationsroutinen. Das Konzept der Task Pools wird fur Plattformen mit gemeinsamem Speicher zur dynamischen Rebalancierung der Last eingesetzt werden. Task Pool Teams stellen verschiedene Task Pool Realisierungen (zentrale, verteilte Task-Schlangen, LI-FO/FIFO Zugriffsprinzip fur die Task-Schlangen, Taskstealingmechanismen) zur Auswahl, die je nach Anwendungsproblem ausgewahlt werden konnen.

Da an die Kommunikation in Task Pool Teams Anforderungen wie multi-threaded und asynchron gestellt werden und diese durch die meisten Implementierungen des MPI bzw. MPI-2 Standards nicht unterstutzt werden, Plattformunabhangigkeit aber erzielt werden soll, wurde durch den Einsatz eines expliziten Kommunikationsthreads die benotigte Funktionalitat hergestellt und uber entsprechende Schnittstellenfunktionen an den Nutzer weitergegeben. Als Implementierungsbasis dient eine Kombination aus Pthreads und MPI. Durch den expliziten Aufruf der Kommunikationsroutinen konnen die task-verarbeitenden Threads Datenanforderungen oder Informationen an Task Pools auf anderen Clusterknoten senden. Diese werden von den Kommunikationsthreads nach Nutzervorgaben ermittelt und ruckgesendet bzw. verarbeitet. Es stehen verschiedene Kommunikationsprotokolle zur Auswahl, die den Anforderungen der unterschiedlichen Anwendungen gerecht werden sollen [HR03b].

Task Pool Teams wurden bereits fur eine Umsetzung des hierarchischen Radiosity Algorithmus und eines baumartigen globalen Optimierungsverfahren auf verteilten Speicher eingesetzt [HR02]. Zur Erfassung der dynamisch variierenden Kommunikationsmuster wurden zusatzlich verteilte Verwaltungsdatenstrukturen entwickelt und in den Radiosity Algorithmus eingebracht. Dadurch konnten die durch Task Pool Teams bereitgestellten Kommunikationsroutinen, z.B. durch vorgezogenes Laden von spater benotigten Informationen oder die Zusammenfassung von Datenanforderungen, effizient ausgenutzt werden. Als weitere anwendungs-spezifische Optimierung wurde ein Software-Cache implementiert, der entfernte Daten mit hohen Zugriffsraten im lokalen Speicher dupliziert und in festgelegten Abstanden aktualisiert.

2.4.2 Kommunikations- und Datenverteilungsschicht fur adaptive Algorithmen

Im Teilprojekt wurde eine Datenverteilungs- und Kommunikationsschicht entwickelt, die die verteilte Adressraumsicht fur den Anwender erhalt, und in Zusammenarbeit mit Projektbereich A an einer adaptiven FEM mit hexaedrischen Elementen getestet [HMR04]. Die software-technischen Prinzipien der Datenverteilungsschicht [HR04a] zur Erzeugung so genannter Koharenzlisten basieren auf den bereits bei den verteilten Verwaltungsdatenstrukturen des hierarchischen Radiosity Algorithmus eingesetzten Methoden. Die Koharenzlisten spiegeln die Interaktion zwischen denen im Speicherbereich verschiedener Prozessoren befindlichen duplizierten Datenstrukturen wider und ermoglichen eine effiziente remote Identifikation dieser Datenstrukturen. Die interne Realisierung der Listen ist vollstandig gekapselt, durch den Nutzer aber uber eine Schnittstelle zugreifbar. Die dynamische Anpassung aufgrund adaptiver Verfeinerung erfolgt aus Sicht des Anwenders automatisch und wird durch Schnittstellenaufrufe des Nutzers ausgelost. Intern werden durch die Schnittstellenaufrufe korrektkeiterhaltende Veranderungen der Datenstrukturen vorgenommen. Die Koharenzlisten enthalten weitere Informationen uber die sich dynamisch andernden Hierarchien zwischen den Datenstrukturen.

Der Austausch von Daten ist in eine gekapselte Kommunikationsschicht eingebettet. Durch die bereitgestellten Schnittstellenfunktionen zum Senden und Empfangen von Daten kann der Nutzer Daten zwischen den verschiedenen Prozessen austauschen, ohne sich um deren Verteilung auf die Prozessoren und ihre Identifikation kummern zu mussen. Die benotigten Informationen fur Datentransfers werden aus den Koharenzlisten innerhalb der Kommunikationsschicht ermittelt.

Die gesamte Kommunikations- und Datenverteilungsschicht wurde hinsichtlich der Funktionalitat so gestaltet, dass eventuelle, zur Lastverteilung notwendige Umverteilungen moglich sind. Die parallele Abarbeitung der FEM Implementierung wird insbesondere auf SMP Clustern durch eine Vielzahl von Hardware und anwendungs-spezifischen Faktoren beeinflusst. Zur effizienten Ausfuhrung und fur dynamische Rebalancierungsentscheidungen wurden diese Faktoren und ihre Abhangigkeiten ermittelt [HR04b].

2.4.3 Kommunikationsbibliothek fur orthogonale Prozessorgruppen

Zur effizienten Implementierung von Algorithmen, die in Form eines zwei- oder mehrdimensionalen Taskgitters spezifiziert werden konnen, wurde das Programmiermodell Orthogonale Prozessorgruppen entwickelt [RRR01a, RRR04] und als Bibliothek umgesetzt [RRR01b]. Algorithmen konnen unter Verwendung der Bibliotheksfunktionen in Abschnitte mit orthogonaler Taskstruktur zerlegt werden, in denen jeweils ausgewahlte Teilgitter des zu Grunde liegenden Taskgitters aktiv sind. Die Abschnitte werden zur Ausfuhrung auf Hyperebenen eines Prozessorgitters abgebildet, wobei die Auswahl der Hyperebenen durch ein entsprechendes Schnittstellendesign unterstutzt wird. Die Bibliothek beinhaltet dafur Funktionen zum Aufbau von orthogonalen Zerlegungen des Prozessorgitters und fur die Zuordnung von Tasks zu Prozessoren. Die Schnittstelle der ORT Bibliothek ist an den Pthread Standard angelehnt. Sie bietet so dem Benutzer die Moglichkeit der Strukturierung des Algorithmus durch die Spezifikation von Abschnitten mit orthogonaler Taskstruktur als separate Funktionen.

Anhand einer Variante der LU-Zerlegung und eines explizit-iterierten Runge-Kutta Verfahrens zur Losung von gewohnlichen Differentialgleichungssystemen wurden die Einsatzmoglichkeiten der Bibliothek untersucht und fur verschiedene parallele Plattformen mit verteiltem Speicher getestet [RRR01c]. Fur beide Beispiele konnte eine deutliche Verbesserung der Laufzeit gegenuber Implementierungen erreicht werden, welche auf herkommlichen parallelen Implementierungen beruhen.

2.4.4 Bibliotheksunterstutzung fur hierarchische Multiprozessor-Tasks

Zur Abarbeitung modularer Programme mit hierarchisch strukturierten dynamisch entstehenden M-Tasks wurde die Laufzeitbibliothek TLib entwickelt [RR02]. Das bereitgestellte Bibliotheks-API bietet im Wesentlichen zwei Arten von Funktionen an: Funktionen zur dynamischen Erzeugung von hierarchischen Prozessorgruppen, wobei verschiedene, gleichzeitig existierende Hierarchien fur dieselbe Prozessormenge moglich sind, sowie Funktionen zur Koordination und Kooperation paralleler, geschachtelter M-Tasks. Ein Ineinanderschachteln von Funktionen beider Gruppen ist moglich, d. h. neue erzeugte M-Tasks konnen wiederum Gruppen-Splittings initiieren, auf denen wiederum M-Tasks ausgefuhrt werden. Ebenso werden durch den dynamischen Charakter rekursive Gruppen-Splittings ermoglicht, so dass rekursive Algorithmen oder Divide & Conquer-Verfahren in naturlicher Weise ausgedruckt und entsprechend parallel abgearbeitet werden konnen. Der dabei entstehende Verwaltungsoverhead ist marginal. Als Anwendungen werden neuere Verfahren zur Losung gewohnlicher Differentialgleichungssysteme betrachtet [RR04] sowie hierarchische Strukturen bei unterschiedlichen Verfahren zur parallelen Matrix-Matrix-Multiplikation [HRR04b, HRR04a], die das derzeit schnellste par-

alle Verfahren an Effizienz ubertreffen. Grundlage der Programmierung mit M-Tasks ist eine inharente M-Task-Struktur, die jedoch durchaus auf verschiedene Arten in einem parallelen Programm realisiert werden kann. Die Gestaltung von M-Task-Programmen kann durch eine vorgeschaltete Spezifikationsphase erganzt werden. In [ORR04] wird ein funktionaler Ansatz vorgestellt, der die auszunutzende modulare Struktur zunachst wiedergibt, um dann eine effiziente Abbildung auf M-Tasks vorzubereiten und durch Transformationen bereitzustellen. Die prinzipielle Vorgehensweise eines solchen Transformationsansatzes wird in [OR04] vorgestellt.

2.4.5 Parallelisierung im Bereich nichtlinearer dynamischer Systeme

Die durchgefuhrten Arbeiten konzentrierten sich auf parallele Orthogonalisierungsverfahren in Isolation und deren Einbindung in die Programmumgebung, wobei verschiedene Varianten entworfen, realisiert und getestet wurden. Zur Orthogonalisierung wurden parallele Versionen der Gram-Schmidt Orthogonalisierung und der QR-Dekomposition realisiert, in die effiziente Basisoperationen aus BLAS eingebunden wurden. Zum Vergleich wurden Algorithmen aus Bibliotheken wie ScaLAPACK herangezogen. Die parallelen Varianten unterscheiden sich hinsichtlich der Datenaufteilungen in spalten- und/oder zeilenweise Blockverteilung der Eingabematrix auf einem logisch zweidimensionalen Prozessgitter. Die Einbindung der Reorthogonalisierungskomponente benotigt je nach Datenverteilung einen nicht unerheblichen Kommunikationsaufwand zur Erhaltung einer Datenverteilungsvariante fur die korrekte Programmabarbeitung. Je nach Kommunikationsoverhead der Schnittstelle und paralleler Kosten der Reorthogonalisierung kann auch eine suboptimale Orthogonalisierungskomponente zur besten Gesamtleistung fuhren. Stark beeinflusst wird dies auch durch die genutzte parallele Hardware und es wurden daher Experimente auf verschiedenen Rechnern durchgefuhrt, dem Beowulf-Cluster CLiC, einem Dual-XEON-Cluster und dem IBM Regatta-System des NIC [RRSY04]. Die entstandene Bibliothek von parallelen Orthogonalisierungsverfahren und zugehorigen Schnittstellen [Sch04] erlaubt eine flexible, modulare Zusammensetzung des Gesamtprogramms zur effizienten Nutzung der Hardware.

Literaturverzeichnis

Referierte Zeitschriftenbeitrage

- [HRT04] K. Hering, G. Runger, and S. Trautmann. Modular Construction of Model Partitioning Processes for Parallel Logic Simulation, Erscheint in: Special Issue: Int. Journal of Computational Science and Engineering, Inderscience, 2004.
- [KRR04] C. Koziar, R. Reilein, and G. Runger. Load Imbalance Aspects in Atmosphere Simulations, Erscheint in: Special Issue: Int. Journal of Computational Science and Engineering, Inderscience, 2004.
- [OR04] J. O'Donnell and G. Runger. Derivation of a Logarithmic Time Carry Lookahead Addition Circuit, Erscheint in: Journal of Functional Programming, Cambridge University Press, 2004.

- [RR04b] T. Rauber and G. Runger. Program-Based Locality Measures for Scientific Computing, Erscheint in: *Int. Journal of Foundations of Computer Science*, World Scientific, 2004.
- [RRR04] T. Rauber, R. Reilein, and G. Runger. Group-SPMD programming with orthogonal processor groups. *Concurrency: Practice and Experience*, 16(2–3):173–195, 2004.

Buchbeitrage

- [RR04a] T. Rauber and G. Runger. Parallel Implementation Strategies for Algorithms from Scientific Computing. In Hergert W., Ernst A., and Dane M., editors, *Computational Materials Science, From Basic Principles to Material Properties, Series: Lecture Notes in Physics, Vol. 642*. Springer Verlag, 2004.

Referierte Konferenz- und Workshopbeitrage

- [HMR04] J. Hippold, A. Meyer, and G. Runger. An Adaptive, 3-Dimensional, Hexahedral Finite Element Implementation for Distributed Memory. In J. J. Dongarra M. Bubak, G. D. van Albada, editor, *Proc. of Int. Conf. on Computational Science (ICCS04), LNCS 3037*, pages 149–157. Springer Verlag, Poland, Krakau, 2004.
- [HR03a] J. Hippold and G. Runger. A Communication API for Implementing Irregular Algorithms on Clusters of SMPs. In J. Dongarra, D. Laforenza, and S. Orlando, editors, *Proc. of the 10th EuroPVM/MPI2003, LNCS 2840*, pages 455–463. Springer Verlag, Venedig, Italien, 2003.
- [HR03b] J. Hippold and G. Runger. Task Pool Teams for Implementing Irregular Algorithms on Clusters of SMPs. In *Proc. of the 17th Int. Parallel & Distributed Processing Symposium (IPDPS03), (CD-ROM)*, Nizza, Frankreich, 2003.
- [HR04a] J. Hippold and G. Runger. A Data Management and Communication Layer for Adaptive, Hexahedral FEM. Erscheint in: *Proc. of Euro-Par 2004, Pisa, Italien, LNCS*, Springer Verlag, 2004.
- [HR04b] J. Hippold and G. Runger. Interaction of Cache, Communication, and Load Increase on SMP Clusters for Parallel Adaptive FEM, Angenommen fur: PARA04, Workshop on State-of-the-Art in Scientific Computing, Lyngby, Danemark, 2004.
- [HRR04a] S. Hunold, T. Rauber, and G. Runger. Multilevel Hierarchical Matrix Multiplication on Cluster. Erscheint in: *Proc. of Int. Conf. on Supercomputing (ICS04)*, Saint-Malo, Frankreich, 2004.
- [HRR04b] S. Hunold, T. Rauber, and G. Runger. Hierarchical Matrix-Matrix Multiplication Based on Multiprocessor Tasks. In J. J. Dongarra M. Bubak, G. D. van Albada, editor, *Proc. of Int. Conf. on Computational Science (ICCS04), LNCS 3037*, pages 3–11. Springer Verlag, Poland, Krakau, 2004.
- [ORR04] J. O’Donnell, T. Rauber, and G. Runger. Functional Realization of Coordination Environments for Mixed Parallelism. In *Proc. of IPDPS: 6th Workshop on Advances in Parallel & Distributed Computational Models (APDCM04), (CD-ROM)*, Santa Fe, New Mexico, 2004.
- [RR02] T. Rauber and G. Runger. Library Support for Hierarchical Multi-Processor Tasks. In *Proc. of ACM/IEEE Supercomputing Conf. (SC02), (CD-ROM)*, Baltimore, USA, 2002.

- [RR04] T. Rauber and G. Runger. Execution Schemes for Parallel Adams Methods. Erscheint in: Proc. of Euro-Par 2004, Pisa, Italien, LNCS, Springer Verlag, 2004.
- [RRR01a] T. Rauber, R. Reilein, and G. Runger. Orthogonal Processor Groups for Message-Passing Programs. In *Proc. of HPCN Europe 2001, LNCS 2110, Amsterdam, Niederlande*, pages 363–372. Springer Verlag, 2001.
- [RRR01b] T. Rauber, R. Reilein, and G. Runger. Library Support for Orthogonal Processor Groups. In *Proc. of the 13th ACM Symposium on Parallel Algorithms & Architectures (SPAA)*, pages 316–317, Kreta, Griechenland, 2001. ACM Press.
- [RRR01c] T. Rauber, R. Reilein, and G. Runger. ORT – A Communication Library for Orthogonal Processor Groups. In *Proc. of ACM/IEEE Supercomputing Conf. (SC01), (CD-ROM)*, Denver, USA, 2001.
- [RRSY04] G. Radons, G. Runger, M. Schwind, and G. Yang. Parallel Algorithms for the Determination of Lyapunov Characteristics of Large Nonlinear Dynamical Systems, Angenommen fur: PARA04, Workshop on State-of-the-Art in Scientific Computing, Lyngby, Danemark, 2004.

Eingereichte Zeitschriftenbeitrage

- [HR] J. Hippold and G. Runger. Task Pool Teams: A Hybrid Programming Model for Implementing Irregular Algorithms on Cluster of SMPs, Eingereicht als Zeitschriftenbeitrag.

Interne Berichte und Arbeiten

- [Hip01] J. Hippold. *Dezentrale Taskpools auf Rechnern mit verteiltem Speicher*. Diplomarbeit, TU-Chemnitz, 2001.
- [HR02] J. Hippold and G. Runger. *Task Pool Teams for Implementing Irregular Algorithms on Clusters of SMPs*. SFB-Bericht 02-18, TU Chemnitz, SFB 393, 2002.
- [Sch04] M. Schwind. *Diplomarbeit, Voraussichtliche Fertigstellung: 2004*. TU-Chemnitz, 2004.

2.5 Offene Fragen / Ausblick

Die Betrachtung und Untersuchung verschiedener irregularer Algorithmen hinsichtlich ihrer effizienten Parallelisierung auf Rechnern mit verteiltem Speicher bzw. Clustern oder Clustern von SMPs hat zur Entwicklung der oben aufgefuhrten Programmierumgebungen, -komponenten und -bibliotheken gefuhrt, die jeweils charakteristische Merkmale in der Parallelisierung unterstutzen. Dabei konnten verschiedene Klassen von irregularen Anwendungen identifiziert werden, deren jeweiliger Grad an Irregularitat in die entsprechende Softwareentwicklungsstrategie eingeflossen ist, etwa in Form spezieller Kommunikationsprotokolle oder zusatzlicher Verwaltungsdatenstrukturen, wodurch fur jede Klasse die Moglichkeit zur Entwicklung effizienter paralleler Programme bereitgestellt wurde. Die tatsachlich resultierende Effizienz hangt zuletzt jedoch immer noch vom jeweiligen Anwendungsalgorithmus und dem verwendeten parallelen Hardwaretyp, ja sogar der speziellen parallelen Maschine, ab. Bei den Task Pool Teams etwa hat die Cachestruktur Einfluss auf die zu verwendende Taskqueue-Variante oder bei

adaptiven Verfahren ist die Netzgeschwindigkeit fur Lastbalancierung einzubeziehen. Ebenso hat die Netzwerk- und Prozessorgeschwindigkeit Auswirkungen fur die effiziente M-Task-Programmierung und die Ausnutzung von M-Tasks. Dies gilt auch fur die dargestellte Einbindung von Orthogonalisierungsalgorithmen in komplexe Gesamtalgorithmen. Ausfurhrliche Tests haben diese und weitere Phanomene gezeigt und zum Tuning der jeweiligen Anwendung mit der passenden Programmierumgebung bzw. -bibliothek gefurhrt. Eine weiterfuhrende Fragestellung ist nun, wie diese jeweils separat durchgefuhrten Tuningschritte in Softwareentwicklungsumgebungen, -werkzeuge und -komponenten aufgenommen werden konnen.

Software, die charakteristische Merkmale einer parallelen Plattform zunachst ermittelt und dann fur eine effiziente Abarbeitung nutzt, wird auch selbstadaptierend genannt und wurde bisher eher fur regelmaige Probleme eingesetzt, etwa zur Cacheausnutzung. Selbstadaptierende Software fur irregulare Probleme bzw. fur auszuwahlende Teilklassen stellt hier eine Herausforderung dar. Adaptives Verhalten der Software sollte Speicherhierarchien und Parallelitat berucksichtigen und ist mit dem hierarchischen und adaptiven Verhalten des eigentlichen Algorithmus abzustimmen. Besondere Wichtigkeit kommt adaptiver Software auf heterogen parallelen Plattformen zu.

