Teilprojekt

B8

Parallelisierung irregulärer numerischer Algorithmen

2.1 Teilprojekt B8

Parallelisierung irregulärer numerischer Algorithmen

2.1.1 Antragsteller

Prof. Dr. Gudula Rünger
Professur Praktische Informatik
Fakultät für Informatik
Technische Universität Chemnitz
09107 Chemnitz
Tel.: (0371) 531-1794
Fax: (0371) 531-1803
ruenger@informatik.tu–chemnitz.de

2.1.2 Projektbearbeiter

Dipl. Inf. Judith Hippold, Praktische Informatik, Projektstelle: 01/2002 - 12/2005

Dr. Robert Reilein-Ruß, Praktische Informatik, Grundausstattung: 07/2001 - 09/2005

2.2 Ausgangsfragestellung / Einleitung

Gegenstand des Teilprojektes ist die effiziente parallele Realisierung irregulärer Algorithmen auf Rechnern mit verteiltem Speicher, Clustern oder Clustern von SMPs *(symmetric multiprocessor)*. Die Klasse der irregulären Algorithmen umfasst Probleme mit unregelmäßigen oder laufzeitabhängigen Berechnungs- und Kontrollstrukturen, für die Standardparallelisierungstechniken, wie datenparallele oder SPMD-Abarbeitung, nicht die gewünschte Effizienz und Skalierbarkeit aufweisen oder die sich einer solchen regelmäßigen Parallelisierung ganz entziehen. Gründe für irreguläres Verhalten sind vielfältig und umfassen dünn-besetzte, blockstrukturierte, adaptive oder hierarchische Anwendungsalgorithmen mit unterschiedlicher Dynamik der Daten- und Berechnungsstrukturen. Entsprechend der Ausprägung der Irregularität variieren auch die anzuwendenden Parallelisierungsmethoden für die jeweilige Anwendungsklasse, die vom Einsatz alternativer Programmiermodelle mit planbaren Eigenschaften bis hin zu vollständig dynamischen Abarbeitungskonzepten zur Laufzeit reichen.

Ziel des Projektes war es, die Parallelisierung irregulärer Algorithmen insbesondere hinsichtlich der Möglichkeiten zur effizienten Realisierung der laufzeitabhängigen Komponenten auf Rechnern mit verteiltem Adressraum zu untersuchen und hierbei die Aspekte geeigneter Datenstrukturen und Analysefunktionen, den Einsatz von Taskpools für das Scheduling und die Lastverteilung sowie Kommunikationsoptimierung schwerpunktmäßig zu berücksichtigen. 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 Softwareerstel-

lung ergänzend einbringen können. Als Anwendungsalgorithmen wurden das hierarchische Radiosity-Verfahren, adaptive gitterbasierte Lösungsverfahren für partielle Differentialgleichungen, strukturiert irreguläre Probleme sowie Berechnungen im Zusammenhang mit nichtlinearen dynamischen Systemen untersucht. Letztere erfordern Kooperationen mit anderen Teilprojekten des SFB393. Die Untersuchung dieser Anwendungsklassen mündet in die Teilaufgaben:

- Task Pool Teams zur Parallelisierung hierarchischer Algorithmen,
- Daten- und Kommunikationsschicht für adaptive Algorithmen,
- Kommunikationsbibliothek für orthogonale Prozessorgruppen,
- Bibliotheksunterstützung für hierarchische Multiprozessor Tasks,
- Parallelisierung im Bereich nichtlinearer dynamischer Systeme.

2.3 Forschungsaufgaben / Methoden

2.3.1 Task Pool Teams zur Parallelisierung hierarchischer Algorithmen

Das hierarchische Radiosity Verfahren, ein globales Beleuchtungsverfahren der Computergrafik, dient zur Simulation der Beleuchtung 3-dimensionaler Szenen in geschlossenen Räumen. Aufgrund des hierarchischen Berechnungsansatzes erfolgt eine unregelmäßige Verfeinerung der zu Grunde liegenden Datenstrukturen, die stark von der Eingabeszene des Algorithmus abhängt und eine parallele Umsetzung erschwert. Für Architekturen mit gemeinsamem Speicher sind bereits parallele Implementierungen erfolgt [WOT+95, PRR98, KR02, KR04]. Dabei wurde der Algorithmus in Berechnungsaufgaben, sogenannte Tasks, unterteilt und diese mit Taskpools [SHT+95, But97, RR00c] abgearbeitet.

Taskpools werden zur dynamischen Rebalancierung der Last eingesetzt. Die Taskpool Datenstruktur dient hierbei zum Speichern und Verwalten der Tasks. Eine feste Anzahl von Threads entnimmt Tasks aus dem Pool zur Abarbeitung und stellt dynamisch neue Tasks ein. Bei entsprechendem Scheduling der Threads auf die verfügbaren Prozessoren erfolgt eine effiziente Auslastung, da nach Abarbeitung eines Tasks durch einen Taskpoolthread automatisch ein neuer Task aus dem Pool entnommen wird. Je nach Anwendungsproblem können verschiedene interne Umsetzungen der Taskpools von Nutzen sein. Gängig sind Pools mit einer zentralen Taskqueue aber auch mit separaten Queues für die einzelnen Threads. Vorteilhaft beim Einsatz mehrerer Taskqueues sind die fehlenden Zugriffskonflikte bei zeitgleich zugreifende Threads. Um hier jedoch Lastbalancierung zu erreichen, sind zusätzliche Mechanismen, z. B. Taskstealing, 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 für Rechner mit verteiltem Speicher gestaltet sich komplexer, da die dynamischen Datenzugriffsmuster variieren und auch Zugriffe auf Daten im Speicherbereich anderer Prozessoren nach sich ziehen können, was zu irregulären Kommunika-

tionsmustern führt. Für 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 über derselben Datenmenge bearbeiten. Damit ergibt sich die Anforderung der Realisierung von asynchronen, multi-threaded fähigen Kommunikationsroutinen, verteilten Verwaltungsdatenstrukturen für die effiziente Identifikation von Daten in nicht-lokalen Speicherbereichen und geeigneten Lastbalancierungsmechanismen für die Migration von Tasks zwischen den Clusterknoten. Zur Umsetzung der Anforderungen wurde im Projekt das Konzept der Task Pool Teams entwickelt.

2.3.2 Daten- und Kommunikationsschicht für adaptive Algorithmen

Adaptive Probleme zeichnen sich durch sich dynamisch verfeinernde, unregelmäßige Gitterstrukturen aus, auf denen Berechnungen ausgeführt werden, die die dynamische Verfeinerung hervorrufen. Ein typisches Anwendungsbeispiel sind adaptive Finite Elemente Methoden (FEM). Eine der Herausforderungen der parallelen Realisierung adaptiver gitterbasierter Lösungsverfahren für partielle Differentialgleichungen bildet die durch die Verteilung der adaptiven Gitterstrukturen auf die Prozessoren notwendige Organisation eines effizienten Datentransfers. Datenstrukturbestandteile des Gitters, die sich bzw. deren über- oder untergeordnete Strukturen sich 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 hängenden Knoten herausgestellt. Hier sind für die resultierende unregelmäßige Kommunikation geeignete Kommunikationsprotokolle und geeignete verteilte Verwaltungsdatenstrukturen notwendig, die die aufkommenden Kommunikationsanforderungen realisieren, verringern und optimieren. Darauf aufbauend werden Mechanismen zur Lastbalancierung benötigt, die eine Umverteilung der Daten berechnen und vornehmen.

Für die Berechnung balancierter Partitionen existieren eine Reihe von Algorithmen, die je nach Eingabeproblem und Anwendung unterschiedlich gute Leistung erzielen. [TDF06] gibt einen Überblick und unterscheidet nach geometrischen und graphbasierten Methoden. Space filling curves [GZ02] und rekursive Bisektionstechniken [Pot96, Els97] arbeiten mit den Koordinatenwerten der zu partitionierenden Elemente und zählen deshalb zu den geometrischen Methoden. Graphbasierte Methoden führen die Berechnung der Umverteilung auf das NP-vollständige Graphpartitionierungsproblem zurück. Die rekursive Graphbisektionen [KL70] betrachtet im Gegensatz zur rekursiven Spektralbisektionen [PSWB92], die mit Eigenwerten der Adjazenzmatrix arbeitet, lokale Grapheigenschaften. Multilevel-Algorithmen zur Graphpartitionierung [KK99, HL95, SKK00] stellen wie die rekursive Spektralbisektionen globale Methoden dar und ähneln der Idee der Mehrgitterverfahren. Algorithmen zur effizienten Partitionierung implementieren beispielsweise ParMETIS [KSK02], METIS [KK95] und CHACO [HL93]. Die tatsächliche effiziente Umverteilung anhand der neu berechneten Partitionen liegt meist in der Verantwortung des Anwendungsprogrammierers.

Mit Hinblick auf die variable Einsetzbarkeit adaptiver gitterbasierter Lösungsverfahren bietet ein gekapselt implementierter Lösungsansatz oben genannter Probleme erhebliche Vorteile. So können Erweiterungen und Optimierungen leichter eingebracht, Plattformunabhängigkeit erzielt und der Anwendungsprogrammierer von der Realisierung effizienter Kommunikationmechanismen und verteilter Datenstrukturen entlastet werden. Systeme mit Unterstützung irregulärer Anwendungen durch Aufsetzen eines globalen Adressraummodells sind beispielsweise Titanium [YSP+98] oder TreadMarks [ACD+96]. Die Sicht auf einen verteilten Adressraum für den Anwender zu erhalten, bietet jedoch durchaus Vorteile, da vom Anwendungsprogrammierer anwendungsspezifische Details und Optimierungen speziell für verteilte Berechnungen eingebracht werden können. Zudem existiert meist bereits eine sequentielle Programmversion, die dann mit nicht zu umfangreichen Modifizierungen auf verteilten Speicher portiert werden kann.

2.3.3 Kommunikationsbibliothek für orthogonale Prozessorgruppen

Obwohl die parallele Programmierung im SPMD-Stil mit Standardkommunikationsbibliotheken wie MPI oder PVM häufig zu guten Ergebnissen auf Rechnern mit verteiltem Speicher führt, kann dies auf Maschinen mit sehr großer Prozessoranzahl Skalierbarkeits- und Effizienzprobleme verursachen, insbesondere wenn kollektive Kommunikationsoperationen eingesetzt werden, etwa für Konvergenztests oder zur Akkumulation lokaler Teilergebnisse. Ein Grund ist die Ausführungszeit kollektiver Kommunikationsoperationen, die logarithmisch oder linear in der Anzahl der teilnehmenden Prozesse steigt [LYSK04, RR04e], so dass Kommunikation auf kleineren Teilgruppen von Prozessoren ausgeführt werden sollte, wenn dies aus algorithmischen Gründen möglich ist.

Eine solche Eigenschaft weisen Algorithmen mit zwei- oder höher-dimensionalen Taskgittern auf, in denen Berechnungs- und Kommunikationsphasen der Abarbeitung so strukturiert sind, dass Tasks jeweils nur mit Tasks innerhalb eines oder weniger Teilgitter 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 höherdimensionales Prozessorgitter bereit, für das eine feste Anzahl von Prozessorpartitionen betrachtet wird, die jeweils disjunkten mehrdimensionalen Teilgittern entsprechen. Ein entsprechendes Mapping von Tasks zu Prozessoren führt so zu Berechnungs- und Kommunikationsphasen auf den alternativ zur Verfügung stehenden Partitionen im Gruppen-SPMD-Stil. Zur Programmierung solcher Programmstrukturen wird eine komfortable Programmierumgebung benötigt, die einerseits eine leichte Spezifikation der Task-, Partitions- und Mappingstruktur erlaubt und andererseits die benötigten Gruppen, Kommunikatoren und Zuordnungen im Hintergrund effizient aufbaut und verwaltet.

2.3.4 Bibliotheksunterstützung für hierarchische Multiprozessor Tasks

Multiprozessor Tasks (M-Task) bezeichnen Teilaufgaben eines Anwendungsalgorithmus, die jeweils auf mehreren Prozessoren ausgeführt werden können. Der gesamte Algorithmus kann aus einer Menge solcher, miteinander kooperierender Multiprozessor Tasks bestehen und durch einen M-Task Graphen mit Abhängigkeiten dargestellt werden. Solche auch als strukturiert irregulär bezeichnete Algorithmen finden sich in großen gekoppelten Anwendungen, aber auch in neuen parallelen Lösungsverfahren für gewöhnliche Differentialgleichungssysteme oder in grobkörnigen hierarchischen Verfahren 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. Auch hier können Skalierbarkeit und Effizienz durch Nutzung von Prozessorteilgruppen erreicht werden. Zur Programmierung existieren eine Reihe von Ansätzen, u. a. Fx [SSOG93, SY97], Paradigm [BCG+95, RSB97], Braid, Opus und Orca. Einen Überblick gibt [BH98]. Die Ansätze haben spezifische integrierte Kostenmodelle mit deren Hilfe geeignete Implementierungen gewählt werden können. Dabei können Schedulingalgorithmen für Multiprozessor Taskscheduling genutzt werden [TWY92, TLW+94, RR96, RR00a]. Eine Herausforderung ist die dynamische Entstehung von M-Task Strukturen, beispielsweise bei hierarchischen Algorithmen. Hierfür muss die Erzeugung und Abarbeitung von M-Tasks mit Abhängigkeiten zur Laufzeit möglich sein, was die dynamische Kreierung von Prozessorteilgruppen und Kommunikationsstrukturen einschließt.

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 großen nichtlinearen dynamischen Systemen betrachtet. Die grobe Struktur des Anwendungsprogramms besteht in der simultanen Integration zweier großer Systeme von linearen bzw. nichtlinearen gewöhnlichen Differentialgleichungen, deren Integrationszeitschritte periodisch durch notwendige Reorthogonalisierung der erzeugten Matrix von Lyapunovvektoren unterbrochen werden.

Die Berechnung der Matrix der Lyapunovvektoren und die Reorthogonalisierung stellen den zeitaufwendigsten Programmteil dar, so dass die Ausnutzung paralleler Abarbeitung insbesondere im Hinblick der Lösung größerer Probleme wesentlich ist. Die Aufgaben der Überführung in eine effziente 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 möglicherweise mit Kommunikation verbundene Gestaltung der Schnittstelle gegeneinander abzuwägen, um hohe Effizienz zu erzielen.

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.

[Els97] U. Elsner. Graph Partitioning. Technical Report SFB 393/97_27, TU Chemnitz, Chemnitz, Germany, 1997.

[GZ02] M. Griebel and G. Zumbusch. In H. Rollnik and D. Wolf, editors, *NIC Symposium 2001*, volume 9, pages 479–492. Forschungszentrum Jülich, 2002.

[HL93] B. Hendrickson and R. Leland. The CHACO User's Guide. Technical Report 93-2339, Sandia National Lab, Albuquerque, NM, 1993.

[HL95] B. Hendrickson and R. Leland. A multi-level algorithm for partitioning graphs. In *Proc. of the ACM/IEEE Conf. on Supercomputing (http://www.supercomp.org/sc95/proceedings)*, San Diego, CA, USA, 1995.

[KK95] G. Karypis and V. Kumar. *METIS Unstructured Graph Partitioning and Sparse Matrix Ordering System.* http://www-users.cs.umn.edu/∼karypis/metis/, 1995.

[KK99] G. Karypis and V. Kumar. A Fast and High 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.* http://www-users.cs.umn.edu/∼karypis/metis/, 2002.

[LYSK04] G. R. Luecke, J. Yuan, S. Spanoyannis, and M. Kraeva. Performance and Scalability of MPI on PC Clusters. *Concurrency: Practice and Experience*, 16(1):79–107, 2004.

[Pot96] A. Pothen. Graph Partitioning Algorithms with Applications to Scientific Computing. In D. F. Keyws, A. H. Sameh, and V. Venkatakrishnan, editors, *Parallel Numerical Algorithms*. Kluwer, 1996.

[PRR98] A. Podehl, T. Rauber, and G. Rünger. A Shared-Memory Implementation of the Hierarchical Radiosity Method. *Theoretical Computer Science*, 196(1-2):215–240, 1998.

[PSWB92] A. Pothen, H. D. Simon, L. Wang, and S. T. Barnard. Towards a Fast Implementation of Spectral Nested Dissection. In *Proc. of the ACM/IEEE Conf. on Supercomputing*, pages 42–51, 1992.

[RR96] T. Rauber and G. Rünger. 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. Rünger. A Transformation Approach to Derive Efficient Parallel Implementations. *IEEE Transactions on Software Engineering*, 26(4):315–339, 2000.

[RR00b] T. Rauber and G. Rünger. Deriving Array Distributions by Optimization Techniques. *Journal of Supercomputing*, 15(3):271–293, 2000.

[RR00c] T. Rauber and G. Rünger. *Parallele und Verteilte Programmierung*. Springer, 2000.

[RR04] T. Rauber and G. Rünger. Modelling the Runtime of Scientific Programs on Parallel Computers. In Y. Pan and L. T. Yang, editors, *Parallel and Distributed Scientific and Engineering Computing: Practice and Experience*, volume 15 of *Advances in Computation: Theory and Practice*, pages 51–65. Nova Science Publishers, 2004.

[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. In *Proc. of the ACM/IEEE Conf. on Supercomputing, CD-ROM*, Dallas, TX, USA, 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.

[TDF06] J. D. Teresco, K. D. Devine, and J. E. Flaherty. Partitioning and Dynamic Load Balancing for the Numerical Solution of Partial Differential Equations. In A. M. Bruaset and A. Tveito, editors, *Numerical Solution of Partial Differential Equations on Parallel Computers, Lecture Notes in Computational Science and Engineering*, volume 51. Springer, 2006.

[TLW+94] J. Turek, W. Ludwig, J. Wolf, L. Fleischer, P. Tiwari, J. Glasgow, U. Schwiegelshohn, 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 ähnlicher Algorithmen wurde das hybride Programmiermodell der Task Pool Teams entwickelt. Task Pool Teams [HR06b] stellen ein Programmierkonzept für verteilten Adressraum bzw. hybriden Adressraum dar und können als Erweiterung und Verallgemeinerung des Taskpool Konzeptes verstanden werden. Dem Nutzer werden Task Pool Teams über eine einfache Schnittstelle zur Verfügung gestellt. Das hybride Programmiermodell kombiniert Shared-Memory- und Message-Passing-Programmierung durch Taskpools und asynchrone Kommunikationsroutinen. Das Konzept der Taskpools wird für Plattformen mit gemeinsamem Speicher zur dynamischen Rebalancierung der Last eingesetzt. Task Pool Teams stellen verschiedene Taskpool Realisierungen (zentrale, verteilte Taskqueues, LIFO/FIFO Zugriffsprinzip für die Taskqueues, Taskstealingmechanismen) zur Verfügung, die je nach Anwendungsproblem ausgewählt werden können.

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 unterstützt werden, Plattformunabhängigkeit aber erzielt werden soll, wurde durch den Einsatz eines expliziten Kommunikationsthreads die benötigte Funktionalität hergestellt und über entsprechende Schnittstellenfunktionen an den Nutzer weitergegeben. Als Implementierungsbasis dient eine Kombination aus Pthreads

und MPI. Durch den expliziten Aufruf der Kommunikationsroutinen können die taskverarbeitenden Threads Datenanforderungen oder Informationen an Taskpools auf anderen Clusterknoten senden. Diese werden von den Kommunikationsthreads nach Nutzervorgaben ermittelt und rückgesendet bzw. verarbeitet. Es stehen verschiedene Kommunikationsprotokolle zur Auswahl, die den Anforderungen der unterschiedlichen Anwendungen gerecht werden sollen [HR03a].

Task Pool Teams wurden für die Umsetzung des hierarchischen Radiosity Algorithmus [HR03b] eingesetzt. Zur Erfassung der dynamisch variierenden Kommunikationsmuster wurden zusätzlich verteilte Verwaltungsdatenstrukturen entwickelt und in den Radiosity Algorithmus eingebracht. Dadurch konnten die durch Task Pool Teams bereitgestellten Kommunikationsroutinen, z. B. durch vorgezogenes Laden von später benötigten Informationen oder die Zusammenfassung von Datenanforderungen, effizient ausgenutzt werden. Als weitere anwendungsspezifische Optimierung wurde ein Software-Cache implementiert, der nicht-lokale Daten im lokalen Speicher dupliziert und in festgelegten Abständen aktualisiert.

Task Pool Teams unterstützen zweistufige Lastbalancierung. So wird innerhalb eines Clusterknotens je nach eingesetzter Task Pool Implementierung eine Balancierung der Tasks unter den verfügbaren CPUs durch zentrale Taskqueques bzw. Taskstealingmechanismen erzielt. Zusätzlich können durch die Migration von Tasks die Lastimbalancen zwischen den verschiedenen Clusterknoten verringert werden. Der Nutzer hat durch Auswahl geeigneter Taskpool Implementierungen die Möglichkeit zwischen den verschiedenen Lastbalancierungsebenen je nach Erfordernis des Anwendungsalgorithmus zu wählen. Ein globales Optimierungsverfahren [HR02] wurde mit mehrstufiger Lastbalancierung realisiert und getestet [HR06b].

Task Pool Teams und Task Pool Team Konzepte wurden weiterhin zur parallelen Realisierung der Simulation von Diffusionsprozessen auf Fraktalen [Hof05, HHRS05] und einer auf Simulated Annealing basierenden Anwendung [Sch04] des SFB Teilprojektes C3 eingesetzt. Insbesondere bei der Simulation von Diffusionsprozessen konnte im Vergleich zur parallelen Implementierung mit MPI aufgrund des Einsatzes variablerer Kommunikationsmuster eine erhebliche Leistungssteigerung erzielt werden.

2.4.2 Kommunikations- und Datenverteilungsschicht für adaptive Algorithmen

Innerhalb des Teilprojektes wurde eine Datenverteilungs- und Kommunikationsschicht entwickelt, die die verteilte Adressraumsicht für den Anwender erhält, und in Zusammenarbeit mit Projektbereich A an einer adaptiven FEM mit hexaedrischen Elementen getestet [HMR04].

Die software-technischen Prinzipien der Datenverteilungsschicht [HR04] zur Erzeugung sogenannter Kohärenzlisten basieren auf den bereits bei den verteilten Verwaltungsdatenstrukturen des hierarchischen Radiosity Algorithmus eingesetzten Methoden. Die Kohärenzlisten spiegeln die Interaktion zwischen denen im Speicherbereich verschiedener Prozessoren befindlichen duplizierten Datenstrukturen wider und ermöglichen eine effiziente remote Identifikation dieser Datenstrukturen. Die interne Realisierung der Listen

ist vollständig gekapselt, durch den Nutzer über eine Schnittstelle zugreifbar. Die dynamische Anpassung aufgrund adaptiver Verfeinerung erfolgt aus Sicht des Anwenders automatisch und wird durch Schnittstellenaufrufe des Nutzers ausgelöst. Intern werden durch die Schnittstellenaufrufe korrektheitserhaltende Veränderungen der Datenstrukturen vorgenommen. Die Kohärenzlisten enthalten weitere Informationen über die sich dynamisch ändernden 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 kümmern zu müssen. Die benötigten Informationen für Datentransfers werden aus den Kohärenzlisten durch die Kommunikationsschicht ermittelt.

Die parallele Abarbeitung der FEM Implementierung wird insbesondere auf SMP Clustern durch eine Vielzahl von Hardware und anwendungsspezifischen Faktoren beeinflusst. Zur effizienten Ausführung und für dynamische Rebalancierungsentscheidungen wurden diese Faktoren und ihre Abhängigkeiten ermittelt [HR06a]. Die Kommunikations- und Datenverteilungsschicht wurden hinsichtlich der Funktionalität so gestaltet, dass zur Lastverteilung notwendige Datenumverteilungen unterstützt werden. Zur Berechnung der neuen Partitionen wurden verschiedene Algorithmen, u. a. aus ParMETIS, eingebunden und ihre Leistung hinsichtlich verschiedener Eingabeprobleme getestet sowie die Korrektheit der implementierten Umverteilungsfunktionen der Kommunikations- und Datenverteilungsschicht verifiziert.

2.4.3 Kommunikationsbibliothek für orthogonale Prozessorgruppen

Zur effizienten Implementierung von Algorithmen, die in Form eines zwei- oder mehrdimensionalen Taskgitters spezifiziert werden können, wurde das Programmiermodell orthogonale Prozessorgruppen entwickelt [RRR01a, RRR04a] und als Bibliothek umgesetzt [RRR01b]. Algorithmen können unter Verwendung der Bibliotheksfunktionen in Abschnitte mit orthogonaler Taskstruktur zerlegt werden, in denen jeweils ausgewählte Teilgitter des zu Grunde liegenden Taskgitters aktiv sind. Die Abschnitte werden zur Ausführung auf Hyperebenen eines Prozessorgitters abgebildet, wobei die Auswahl der Hyperebenen durch ein entsprechendes Schnittstellendesign unterstützt wird. Die Bibliothek beinhaltet dafür Funktionen zum Aufbau von orthogonalen Zerlegungen des Prozessorgitters und für die Zuordnung von Tasks zu Prozessoren. Die Schnittstelle der ORT Bibliothek ist an den Pthread Standard angelehnt. Sie bietet so dem Benutzer die Möglichkeit 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 Lösung von gewöhnlichen Differentialgleichungssystemen wurden die Einsatzmöglichkeiten der Bibliothek untersucht und für verschiedene parallele Plattformen mit verteiltem Speicher getestet [RRR01c]. Für beide Beispiele konnte eine deutliche Verbesserung der Laufzeit gegenüber herkömmlichen parallelen Implementierungen erreicht werden.

2.4.4 Bibliotheksunterstützung für hierarchische Multiprozessor-Tasks

Zur Abarbeitung modularer Programme mit hierarchisch strukturierten, dynamisch entstehenden M-Tasks wurde die Laufzeitbibliothek TLib entwickelt [RR04c], [RR05b], [RR05d]. Das bereitgestellte Bibliotheks-API [RR02] bietet im Wesentlichen zwei Arten von Funktionen an: Funktionen zur dynamischen Erzeugung von hierarchischen Prozessorgruppen, wobei verschiedene, gleichzeitig existierende Hierarchien für dieselbe Prozessormenge möglich sind, sowie Funktionen zur Koordination und Kooperation paralleler, geschachtelter M-Tasks. Ein Schachteln von Funktionen beider Gruppen ist möglich, d. h. neu erzeugte M-Tasks können Gruppen-Splittings initiieren, auf denen wiederum M-Tasks ausgeführt werden. Ebenso werden durch den dynamischen Charakter rekursive Gruppen-Splittings ermöglicht, so dass rekursive Algorithmen oder Divide & Conquer-Verfahren in natürlicher Weise ausgedrückt und entsprechend parallel abgearbeitet werden können. Der dabei entstehende Verwaltungsoverhead ist marginal.

Als Anwendungen wurden hierarchische Strukturen bei unterschiedlichen Verfahren zur parallelen Matrix-Matrix-Multiplikation [HRR04a, HRR04b], die das derzeit schnellste parallele Verfahren an Effizienz übertreffen, sowie Verfahren zur Lösung gewöhnlicher Differentialgleichungssysteme betrachtet [RR04a]. Zusätzliche Effizienzgewinne bei der Lösung gewöhnlicher Differentialgleichungssysteme konnten durch Lokalitätsausnutzung erzielt werden [RR03, RR04b, RR04d]. Grundlage der Programmierung mit M-Tasks ist eine inhärente 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 ergänzt werden. In [ORR04] wird ein funktionaler Ansatz vorgestellt, der die auszunutzende modulare Struktur zunächst wiedergibt, um dann eine effiziente Abbildung auf M-Tasks vorzubereiten und durch Transformationen bereitzustellen. Die prinzipielle Vorgehensweise eines solchen Transformationsansatzes wird in [OR04] vorgestellt.

Zur Unterstützung der M-Task Programmierung wurde eine Bibliothek für die Umverteilung feldbasierter Datenstrukturen realisiert [RR05a, RR06]. Es werden ein Datenverteilungsformat und Umverteilungsoperationen für Programme mit gemischter Task- und Datenparallelität auf verteiltem Speicher zur Verfügung gestellt. Der Einsatz von M-Task Programmierung für heterogene Systeme und Grids wurde durch eine Trennung der Beschreibung der M-Task Struktur von deren verteilter Ausführung ermöglicht [RR05c]. Hierfür werden die M-Tasks explizit durch einen speziellen Taskgraph repräsentiert, dessen Teilgraphen von den verschiedenen Servern der verteilten Architektur zur Ausführung gebracht bzw. re-scheduled werden.

[RRR04b, Rei05] stellen das TwoL Modell, ein Sprach- und Compileransatz zur Realisierung von Programmen mit gemischter Task- und Datenparallelität, vor. Ausgehend von der Spezifikation der inhärenten Parallelität eines Algorithmus wird in einem Transformationsprozess ein Koordinationsprogramm erzeugt, was an die spezifischen Parameter der Zielplattform angepasst ist.

2.4.5 Parallelisierung im Bereich nichtlinearer dynamischer Systeme

Die durchgeführten Arbeiten konzentrierten sich auf parallele Orthogonalisierungsverfahren und deren effiziente Einbindung in die Programmumgebung, wobei verschiedene Varianten entworfen, realisiert und getestet wurden.

Zur Orthogonalisierung wurden parallele Versionen der Gram-Schmidt Orthogonalisierung [RS05] 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 Datenaufteilung in spalten- und/oder zeilenzyklische Blockverteilung der Eingabematrix auf einem logisch zweidimensionalen Prozessorgitter und der verwendeten BLAS Basisoperationen. Weiterhin wurden Varianten von Gram-Schmidt und QR-Dekompositition entwickelt, die eine Überlappung von Kommunikation und Berechnung erlauben und somit die Effizienz der Orthogonalisierung deutlich erhöhen.

Es konnte eine unmittelbare Abhängigkeit zwischen der Laufzeit der verwendeten Orthogonalisierungsalgorithmen und den verwendeten Blocklängen sowie der Kantenlänge des Prozessorgitters festgestellt werden. Die Ursache sind zum einen erhöhte Lastimbalancen, die einer Steigerung der Effizienz der lokalen BLAS Operationen bei wachsender Blocklänge entgegenwirken. Zum anderen wirken sich unterschiedliche Ausdehnungen des Prozessorgitters entscheidend auf das Kommunikationsvolumen aus. Es wurde daher an der Schnittstelle zwischen Integrationsteil und Reorthogonalisierung eine Möglichkeit der Umverteilung der Matrix der Lyapunovvektoren vorgesehen, die durch Anpassung der Blocklängen und Gitterdimensionen eine effizientere Reorthogonalisierung unterstützt.

Je nach Kommunikationsoverhead der Umverteilung und Kosten der parallelen Reorthogonalisierung kann auch eine suboptimale Orthogonalisierungskomponente zur besten Gesamtleistung führen. Starken Einfluss hat hierbei die verwendete parallele Hardware. Es wurden daher Experimente auf verschiedenen Plattformen, wie dem Beowulf-Cluster CLiC und einem Dual-XEON-Cluster der TU-Chemnitz sowie dem IBM Regatta-System des NIC durchgeführt [RRSY06]. Die entstandene Bibliothek von parallelen Orthogonalisierungsverfahren und zugehörigen Schnittstellen [Sch05] erlaubt eine flexible, modulare Zusammensetzung des Gesamtprogramms zur effizienten Nutzung der Hardware.

Literaturverzeichnis

Referierte Zeitschriftenbeiträge

[HRT05] K. Hering, G. Rünger, and S. Trautmann. Modular Construction of Model Partitioning Processes for Parallel Logic Simulation, Int. Journal of Computational Science and Engineering, Inderscience, 1:22–33, 2005.

[HR06b] J. Hippold and G. Rünger. Task Pool Teams: A Hybrid Programming Environment for Irregular Algorithms on SMP Clusters. *Erscheint in: Concurrency and Computation: Practice and Experience*, 2006.

[KRR05] C. Koziar, R. Reilein, and G. Rünger. Load imbalance aspects in atmosphere simulations, Angenommen für Special Issue: Int. Journal of Computational Science and Engineering, Inderscience, 2005.

[OR04] J. O'Donnell and G. Rünger. Derivation of a Logarithmic Time Carry Lookahead Addition Circuit. *Journ. of Functional Programming, Special Issue on Functional Pearls*, 14(6):697–713, 2004.

[RR04b] T. Rauber and G. Rünger. Improving Locality for ODE Solvers by Program Transformations. *Scientific Programming*, 12(3):133–154, 2004.

[RR04d] T. Rauber and G. Rünger. Program-Based Locality Measures for Scientific Computing. *Int. Journ. of Foundations of Computer Science, Special Issue on Advances in Parallel and Distributed Computational Models*, 15(3):535–554, 2004.

[RR05d] T. Rauber and G. Rünger. Tlib - A Library to Support Programming with Hierarchical Multi-processor Tasks. *Journ. of Parallel and Distributed Comput.*, 65(4):347–360, 2005.

[RR06] T. Rauber and G. Rünger. A Data Re-Distribution Library for Multi-Processor Task Programming. *Angenommen für: International Journal of Foundations of Computer Science*, 2006.

[RRR04a] T. Rauber, R. Reilein, and G. Rünger. Group-SPMD programming with orthogonal processor groups. *Concurrency: Practice and Experience*, 16(2–3):173–195, 2004.

Buchbeiträge

[RR04c] T. Rauber and G. Rünger. Parallel Implementation Strategies for Algorithms from Scientific Computing. In Hergert W., Ernst A., and Däne M., editors, *Computational Materials Science, From Basic Principles to Material Properties, Series: Lecture Notes in Physics, Vol. 642*. Springer Verlag, 2004.

[RR04e] T. Rauber and G. Rünger. Modelling the Runtime of Scientific Programs on Parallel Computers. In Y. Pan and L. T. Yang, editors, *Parallel and Distributed Scientific and Engineering Computing: Practice and Experience*, volume 15 of *Advances in Computation: Theory and Practice*, pages 51–65. Nova Science Publishers, 2004.

[RR05b] T. Rauber and G. Rünger. Exploiting Multiple Levels of Parallelism in Scientific Computing. In A. Doncescu, T. Leng, M. K. Ng, and L. T. Yang, editors, *IFIP TC5 Workshop on High Performance Computational Science and Engineering (HPCSE), World Computer Congress*, volume 172 of *High Performance Computational Science and Engineering*. Springer, Toulouse, France, 2005.

Referierte Konferenz- und Workshopbeiträge

[HMR04] J. Hippold, A. Meyer, and G. Rünger. 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. Rünger. A Communication API for Implementing Irregular Algorithms on SMP Clusters. In J. Dongarra, D. Lafarenza, and S. Orlando, editors, *Proc. of the 10th EuroPVM/MPI*, volume 2840 of *LNCS*, pages 455–463, Venice, Italy, 2003. Springer.

[HR03b] J. Hippold and G. Rünger. Task Pool Teams for Implementing Irregular Algorithms on Clusters of SMPs. In *Proc. of the 17th Intl. Parallel and Distributed Processing Symp. (IPDPS'03)*, CD-ROM, Nice, France, 2003. IEEE Computer Society Press.

[HR04] J. Hippold and G. Rünger. A Data Management and Communication Layer for Adaptive, Hexahedral FEM. In M. Danelutto, D. Laforenza, and M. Vanneschi, editors, *Proc. of Euro-Par 2004*, volume 3149 of *LNCS*, pages 718–725, Pisa, Italy, 2004. Springer.

[HR06a] J. Hippold and G. Rünger. Performance Analysis for Parallel Adaptive FEM on SMP Clusters. In Dongarra, J.; Madsen, K.; Wasniewski, J., editors, *Applied Parallel Computing - State of the Art in Scientific Computing. Proc. of PARA'04*, volume 3732 of *LNCS*, pages 730–739, Lyngby (Copenhagen), Denmark, 2006. Springer.

[HRR04a] S. Hunold, T. Rauber, and G. Rünger. 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.

[HRR04b] S. Hunold, T. Rauber, and G. Rünger. Multilevel Hierarchical Matrix-Matrix Multiplication on Clusters. In *Proc. of the 18th Int. Conf. of Supercomputing (ICS'04)*, pages 136–145, Saint-Malo, France, 2004. ACM.

[ORR04] J. O'Donnell, T. Rauber, and G. Rünger. Functional Realization of Coordination Environments for Mixed Parallelism. In *Proc. of IPDPS'04 Workshop on Advances in Parallel and Distributed Computational Models (APDCM'04)*, CD-ROM, Santa Fe, New Mexico, USA, 2004. IEEE.

[RR02] T. Rauber and G. Rünger. Library Support for Hierarchical Multi-Processor Tasks. In *Proc. of ACM/IEEE Supercomputing Conf. (SC02), (CD-ROM)*, Baltimore, USA, 2002.

[RR03] T. Rauber and G. Rünger. Program-Based Locality Measures for Scientific Computing. In *Proc. of IPDPS: Workshop on Advances in Parallel & Distributed Computational Models (APDCM03), (CD-ROM)*, Nizza, Frankreich, 2003.

[RR04a] T. Rauber and G. Rünger. Execution Schemes for Parallel Adams Methods. In M. Danelutto, D. Laforenza, and M. Vanneschi, editors, *Proc. of Euro-Par 2004*, volume 3149 of *LNCS*, pages 708–717, Pisa, Italy, 2004. Springer.

[RR05a] T. Rauber and G. Rünger. A Data-Re-Distribution Library for Multi-Processor Task Programming. In *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 8*, Denver, CA, USA, 2005. IEEE.

[RR05c] T. Rauber and G. Rünger. M-Task-Programming for Heterogeneous Systems and Grid Environments. In *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 4*, Denver, CA, USA, 2005. IEEE.

[RRR01a] T. Rauber, R. Reilein, and G. Rünger. 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. Rünger. 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. Rünger. ORT – A Communication Library for Orthogonal Processor Groups. In *Proc. of ACM/IEEE Supercomputing Conf. (SC01), (CD-ROM)*, Denver, USA, 2001.

[RRR04b] T. Rauber, R. Reilein, and G. Rünger. On Compiler Support for Mixed Task and Data Parallelism. In G. R. Joubert, W. E. Nagel, F. J. Peters, and W. V. Walter, editors, *Proc. of 12th Int. Conf. on Parallel Computing (ParCo'03)*, Parallel Computing: Software Technology, Algorithms, Architectures & Applications, pages 23–30, Dresden, Germany, 2004. Elsevier.

[RRSY06] G. Radons, G. Rünger, M. Schwind, and H. Yang. Parallel Algorithms for the Determination of Lyapunov Characteristics of Large Nonlinear Dynamical Systems. In Dongarra, J.; Madsen, K.; Wasniewski, J., editors, *Applied Parallel Computing - State of the Art in Scientific Computing. Proc. of PARA'04*, volume 3732 of *LNCS*, pages 1131–1140, Lyngby (Copenhagen), Denmark, 2006. Springer.

[RS05] G. Rünger and M. Schwind. Comparison of Different Parallel Modified Gram-Schmidt Algorithms. In *Proc. of 11th International Euro-Par Conference*, LNCS 3648, pages 826–836, Lisboa, Portugal, 2005. Springer.

Interne Berichte und Arbeiten

[HHRS05] K. H. Hoffmann, M. Hofmann, G. Rünger, and S. Seeger. Task Pool Teams for Implementing Irregular Algorithms on Clusters of SMPs. Preprint SFB393/05, TU-Chemnitz, 2005.

[Hof05] M. Hofmann. Verwendung von Task Pool Team Konzepten zur parallelen Implementierung von Diffusionsprozessen auf Fraktalen, Studienarbeit TU-Chemnitz, Fakultät für Informatik, 2005.

[HR02] J. Hippold and G. Rünger. Task Pool Teams for Implementing Irregular Algorithms on Clusters of SMPs. Preprint SFB393/02-18, TU-Chemnitz, 2002.

[Rei05] R. Reilein. *Eine komponentenbasierte Realisierung der TwoL Spracharchitektur.* Dissertation, TU Chemnitz, Fakultät für Informatik, 2005.

[Sch04] C. Schuster. Evaluierung von Task Pool Teams für eine Master Worker Anwendung, Studienarbeit TU-Chemnitz, Fakultät für Informatik, 2004.

[Sch05] M. Schwind. Implementierung und Laufzeitevaluierung paralleler Algorithmen zur Gram-Schmidt Orthogonalisierung und zur QR-Zerlegung, Diplomarbeit TU-Chemnitz, Fakultät für Informatik, 2005.