2.1 Teilprojekt D4

Geometrie- und problemangepaßtes Pre- und Postprocessing für die parallele Lösung von Aufgaben der Kontinuumsmechanik

2.1.1 Antragsteller

Prof. Dr. rer. nat. habil. Arnd Meyer
Dr. rer. nat. Matthias Pester

Fakultät für Mathematik, Professur Numerische Mathematik

2.1.2 Projektbearbeiter

Dr. Matthias Pester, Professur Numerische Algebra
Dr. Magdalene Meyer, Projektmitarbeiterin bis 12/96
Dr. Gerhard Globisch, Projektmitarbeiter (Tp.A3)
Dipl.-Math. Dag Lohse, Projektmitarbeiter
Dipl.-Math. Christoph Israel, Professur Numerische Analysis
Dipl.-Math. Friedmar Lowke, Professur Numerische Analysis

2.2 Ausgangsfragestellung/Einleitung

Die Lösung komplexer Simulationsaufgaben auf MIMD-Mehrprozessorsystemen mit verteiltem Speicher erfordert eine durchgängige parallele Behandlung der anfallenden sehr umfangreichen Datenmengen. Im Rahmen dieses Teilprojektes waren in engem Zusammenwirken mit anderen Teilthemen spezifische Probleme beim Pre- und Postprocessing für massiv parallele Rechner zu lösen. Die gesamte Problematik läßt sich in drei Teilaufgaben fassen: die sehr eng miteinander verbunden sind:

(1) Benutzeroberfläche für ein Simulationssystem auf Parallelrechnern,
(2) Geometrie- und problemangepaßtes Preprocessing für Parallelrechner,
(3) Paralleles Postprocessing.

Diese sehr eng miteinander verbundenen Teilaufgaben bilden Rahmen und technisch-informatische Grundlage für Test- und Entwicklungsarbeiten in anderen Projekten: Sowohl Pre- als auch Postprocessing sind wichtige Bindeglieder für die Kopplung von parallelen Berechnungen und den notwendigerweise sequentiellen Aktionen bei interaktiver Arbeit an der Workstation. Eine der wichtigsten Fragestellungen war es daher, durch geeignete Schnittstellen einen effektiven Datenfluß zu erreichen, von der weitgehend interaktiven Aufbereitung der Testdaten für die Bearbeitung am Parallelrechner bis zur Visualisierung der Ergebnisse, sowohl unmittelbar während der numerischen Tests als auch im separaten Postprocessing als Nachbereitung. Letzteres gilt insbesondere für die zweckmäßige Darstellung zeitabhängiger Lösungen wie in der Strömungssimulation.

2.3 Forschungsaufgaben/Methoden

2.3.1 Teilaufgabe Entwicklungsumgebung

Bei numerischen rechnergestützten Simulationen bilden Pre- und Postprocessing den Rahmen für den Zugang sowohl des Entwicklers als auch des Anwenders zum Computer. Gegenüber herkömmlichen integrierten Systemen für Arbeitsplatzrechner ergeben sich bei der Nutzung von Parallelrechnern als Compute-Server auch für Pre- und Postprocessing neue Fragestellungen. Die Konzeption von verteilten, parallelen Anwendungssystemen erfordert ein grundsätzlich anderes Herangehen. Das gesamte Simulationssystem besteht aus Komponenten, die kooperierend zusammenarbeiten und unterschiedliche Hardware (Workstation, Parallelrechner) unter einer Benutzeroberfläche vereinen. Dabei sind sowohl Pre- als auch Postprocessing gerade diejenigen Teile eines Simulationssystems, bei denen diese Kopplung besonders in Erscheinung tritt.

Viele der existierenden Simulationsprogramme wurden vor 15-20 Jahren entwickelt und als Insellösung für sequentielle Rechner optimiert. Sie sind daher nicht so strukturiert, daß sie bei einer Portierung, ob von Hand oder mit automatischen Software-Werkzeugen, maximalen Nutzen aus den Möglichkeiten des Parallelrechners ziehen können. Die veränderte Herangehensweise bei der Konzeption verteilter, paralleler Anwendungssysteme erfordert, daß die kooperierenden Komponenten ihre Aktivitäten koordinieren und zu diesem Zwecke Informationen über ein Kommunikationsnetzwerk austauschen [SK89]. Neben der innerhalb paralleler Algorithmen auf einem als Compute-Server dienenden Parallelrechner notwendigen Kommunikation schließt dies auch die Kommunikation ein zu den auf heterogener Hardware agierenden Komponenten mit spezifischer Funktionalität wie Pre- und Postprocessing.

Client/Server-Modelle stellen eine tragfähige Organisationsform für derartige verteilte Systeme dar, weil hier eine klare Trennung zwischen den Komponenten und ihrer Kommunikation sowie Verwaltung vorgenommen wird [Neh90]. Dienste sind durch Server- und Anwendungen durch Client-Objekte realisiert. Bei problemgerechter Spezifikation der Server und Klienten können diese auf für ihre Funktionalität maßgeschneiderte Prozessoren im Netz verteilt werden.

Mit dem Betriebssystem UNIX wird die Basissoftware für offene Systeme bereitgestellt [Sto91]. Auch von seiten der Benutzeroberfläche GUI (Graphical User Interface) wird die Arbeit im Netzwerk unterstützt, sei es Motif von OSF oder Open Look von SUN, mit den drei Niveaus Window, Desktop und Network [Vie91]. Beides sind Werkzeuge für GUI in offenen Systemen, die auf X-Windows basieren. Mit der Entwicklung von Java zu einem industriellen Standard ergeben sich auch plattformübergreifende Gestaltungsmöglichkeiten.

Für den als Compute-Server verwendeten Parallelrechner gibt es insbesondere hinsichtlich der Entwicklungsumgebung mehrere Alternativen. Um hier flexible Testmöglichkeiten offenzulassen, wurde eine hierarchische Modulstruktur geschaffen, mit der die unmittelbare Schnittstelle zu systemspezifischen Leistungen minimal gehalten werden kann. Somit sind alle in den Anwendungsprojekten der Projektbereiche A und D entwickelten Algorithmen ohne spezielle Maßnahmen portierbar auf jegliche Art von Parallelrechnern, die Message Passing unterstützen. Ausgehend von Parix [PARS93], das speziell für GC- und Xplorer-Systeme (Parsytec) vorgesehen ist, wurden mit PVM auf heterogenen Workstationclustern [SGDM93] und MPI als anerkanntem Message-Passing-Standard [MPI93, MPI97] die wesentlichen Schnittstellen berücksichtigt.

2.3.2 Teilaufgabe Preprocessing

Das Konzept der Einbindung des Pre- und Postprocessing in die parallele Realisierung der gesamten Aufgabenstellung erforderte die Definition von Datenschnittstellen, die gegenüber dem klassischen Fall bei herkömmlichen FEM-Programmen die speziellen Bedingungen bei Mehrprozessorsystemen mit lokalem Speicher berücksichtigen müssen. So gliedert sich das Preprocessing in drei wesentliche Etappen:

I. Generierung eines Grobnetzes mit Geometriebeschreibung (Knoten, Kanten, Flächen, Volumenelemente) Materialparametern für einzelne Teilgebiete, sowie die notwendigen Angaben zu Randbedingungen (an Kanten bzw. Flächen).

II. Verteilung des Grobnetzes (Teilgebiete) auf die Prozessoren des Parallelrechners und lokale Netzverfeinerung unter Beibehaltung der Datenstruktur.

III. Erzeugung der typischen (knotenorientierten) FEM-Datenstruktur zur Generierung der Systemmatrizen.

Komponenten wie Netzgenerierung oder Auswertung diskreter Lösungsfunktionen sind offensichtlich parallelisierbare Teile, während die interaktive Geometriebeschreibung oder die unmittelbare grafische Ausgabe typischerweise sequentielle Prozeduren darstellen, die mit einer Workstation am Arbeitsplatz auszuführen sind. Die Realisierung dieser ersten Preprozessor-Etappe erfordert nicht zwingend den Parallelrechner, kann jedoch wesentlichen vorbereitenden Anteil an einem effizienten Einsatz des Parallelrechners haben. Hier handelt es sich überwiegend um interaktive, weniger rechenintensive Tätigkeit. So können auch sogenannte handelsübliche Preprozessoren genutzt werden, sofern eine geeignete Datenschnittstelle zur Ausgabe vorgesehen ist, die eine Weiterverarbeitung im oben genannten Sinne zuläßt, im allgemeinen nach einer Konvertierung auf die eigene Datenstruktur. Die Bestimmung geeigneter Datenschnittstellen für 3D-Geometrien war eine erste wesentliche Aufgabe, die in Verbindung mit anderen Anwendungsprojekten zu klären war. Das betrifft die Eingabe einer Startgeometrie (Grobnetz), ihre Verteilung über die Prozessoren, sowie die interne Behandlung bei der hierarchischen Netzverfeinerung.

Durch geeignete Geometriebeschreibungen, einschließlich elementarer Krümmungen bei Kanten und Flächen, sollte die parallele Erzeugung von gröberen Vernetzungen und Feinnetzen ermöglicht werden, die sich der Geometrie anpassen. Bei der Generierung ersterer kann unter Benutzung von a-priori Informationen bereits eine Balancierung der Last berücksichtigt werden, die im allgemeinen aber während des Ablaufs der Problemsimulation stark gestört wird. In der ersten Phase der Arbeiten war ein annähernd optimaler Lastausgleich an die Vernetzung zu binden und auf der Basis graphentheoretischer Zugänge über eine globale Umverteilung zu erreichen. Die Frage der Einbettung in eine adaptive Arbeitsweise auf Parallelrechnern ist eine nach wie vor kritische und für die Folgezeit wesentliche Problemstellung.

2.3.3 Teilaufgabe Postprocessing

Eine effiziente Arbeitsweise in der rechnergestützten Simulation hängt auch davon ab, wieviele Daten und Simulationsergebnisse der Anwender begreifen und verarbeiten kann. Es besteht somit ein großer Bedarf an grafisch-interaktiver Visualisierung [FK90], d. h. dem Anwender Instrumente bereitzustellen, damit er interaktiv die Darstellungsweise der Ergebnisse steuern, manipulieren und seinen speziellen Bedürfnissen anpassen kann. An die für die Generierung von Darstellungen benötigte Zeit werden besonders hohe Anforderungen gestellt. Eine in diesem Kontext ökonomisch vertretbare Lösung ist in verteilten Sytemen möglich, indem für die Bildberechnung (Erzeugung des Bildrasters) ein Parallelrechner eingesetzt wird und die eigentliche Bildgenerierung auf einem grafischen Terminal erfolgt [FK90, EKZ89]. Die beiden Hauptherangehensweisen zur Realisierung einer Applikationsgrafik sollten hier weiter verfolgt werden: zum einen die auf der Gebietszerlegung aufsetzende parallele Bildberechnung mit der Übertragung von Pixelinformationen zur Workstation des Nutzers, zum anderen eine (mehr oder weniger) vollständige Bereitstellung der FEM-Daten für einen separaten Visualisierungs-Server. Voraussetzung für die zweite Variante ist eine effiziente Netzwerkanbindung zwischen Parallelrechner und Grafikworkstation.

Der Visualisierung im Sinne eines sogenannten „Quick-Post“ wurde auch deshalb große Aufmerksamkeit gewidmet, weil ein klassisches Postprocessing mit entsprechenden handelsüblichen Tools nur über einen beträchtlichen Datentransfer zum Massenspeicher zu realisieren ist, sowie mit dem gleichen Aufwand die spätere Visualisierung an der Workstation erfolgt, abgesehen von der Schwierigkeit der Handhabung oder gleichzeitigen Verfügbarkeit solcher Software oder spezieller Hardware für viele Nutzer einer Forschergruppe. Oft ist eine sofortige grafische Auswertung unmittelbar während der Aktivität des parallelen Testprogramms wertvoller als die exakt gerenderte Darstellung mit Licht und Schatten. Letzteres kann jeweils dem Postprocessing zu Präsentationszwecken vorbehalten bleiben.

Für das Postprocessing bei der numerischen Simulation von strömungsmechanischen Aufgaben, wo zeitabhängige Lösungen auftreten, reicht eine statische Wiedergabe eines Zustandes zu einem bestimmten Zeitpunkt nicht mehr aus, um das Lösungsverhalten zu erfassen. Da aber trotz leistungsfähigster Parallelrechner keine Echtzeitsimulation denkbar ist, müssen hier viele Momentaufnahmen von Einzellösungen zu einem zeitlichen Ablauf zusammengefügt werden. Dies ist wiederum am zweckmässigsten durch eine Folge von visuellen Informationen (z.B. als Bitmap) zu realisieren, die in einem Nachbereitungsschritt zur Videosequenz komplettiert werden können. Solche Prozesse müssen, abgesehen von der Initialisierung verschiedener Parameter wie etwa der Auswahl bestimmter Darstellungsformen, automatisch realisierbar sein, also ohne weitere manuelle Aktionen während der numerischen Simulation.

Bei der Ergebnisdarstellung durch parallele Bildberechnung entsprechend der zur numerischen Simulation vorgenommenen Gebietszerlegung konnte auf vorhandenen Experimentallösungen aus dem Bereich der 2D-FEM-Rechnung aufgebaut werden [Pes94]. Zur Anbindung eines separaten Visualisierungstools wurde als Beispiel das vom SFB 256 zur Verfügung gestellte Programmpaket GRAPE genutzt, womit eine umfangreiche Funktionalität zur Auswertung von 3D-FEM-Daten vorhanden ist. Die kritische Stelle ist auch hier die Geschwindigkeit der Datenbereitstellung vom Parallelrechner über das Netzwerk, sowie die Leistungsfähigkeit der (sequentiellen) Grafikworkstation, wenn 3D-Netze mit mehreren 100 000 Knotenpunkten verarbeitet werden sollen.

Literatur

Fremdliteratur

[DR94] R. Van Driessche and D. Roose. A spectral algorithm for constrained graph partitioning I: The bisection case. TW Report 216, Dept. of Computer Science, Katholieke Universteit Leuven, Belgium, October 1994.

[EKZ89] J. Encarnacao, G. Köberle, and N. Zhang. Distributed supercomputing for graphics applications: A case study on an implementation of the radiosity approach. In *Supercomputer 1989*, pages 11–24, 1989. Informatik Fachberichte Band 211, Springer Verlag Heidelberg 1989.

[Fie75] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(100):619–633, 1975.

[FK90] M. Frühauf und K. Karlsson. Visualisierung von Volumendaten in verteilten Systemen. Sonderdruck, 1990. FhG-AG Darmstadt.

[HL92] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. Technical Report SAND92-1460, Sandia National Laboratories, Albuquerque, NM, 1992.

[HL94] B. Hendrickson and R. Leland. The Chaco User's Guide - Version 2.0. Technical Report SAND94-2692, Sandia National Laboratories, Albuquerque, NM, 1994.

[Kic96] F. Kickinger. Automatic mesh generation for 3D objects. Technical Report No. 96–1, Johannes Kepler Universität Linz, Institut für Mathematik, Feb. 1996.

[MPI93] DRAFT Document for a standard message-passing interface. ESPRIT project P6643, June 1993

[MPI97] W. Gropp and E. Lusk. MPICH Model MPI Implementation. Reference Manual. (Draft) Argonne National Laboratory, April 1997.

[Neh90] J. Nehmer. Einige Thesen zur zukünftigen Entwicklung verteilter Systemarchitekturen. In P. Müller-Stoy, editor, *11. ITG/GI - Fachtagung München*, S. 327 – 337, März 1990.

[PARS93] PARSYTEC Computer GmbH. *PARIX Release 1.2. Reference Manual*, 1993

[Sch95] J. Schöberl. An automatic mesh generator using rules for two and three space dimensions. Technical Report No. 95–3, Johannes Kepler Universität Linz, Institut für Mathematik, Optimierung, Aug. 1995.

[SGDM93] V. S. Sunderam, G. A. Geist, J. Dongarra, R. Manchek. *The PVM concurrent computing system: evolution, experiences, and trends.* Emery University Atlanta, Oak Ridge National Laboratory, University of Tennessee Knoxville, 1993

[SGI95] *IRIS Explorer User's Guuide*, Release 3, Silicon Graphics Inc., 1993, The Numerical Algorithms Group Ltd., 1995.

[SK89] M. Sloman und J. Kramer. *Verteilte Systeme und Rechnernetze.* C. Hanser Verlag München, 1989.

[SS85] Youcef Saad and Martin H. Schultz. Topological properties of hypercubes. Research Report 389, Yale University, Dept. Computer Science, 1985.

[SS89] Youcef Saad and Martin H. Schultz. Data communication in hypercubes. *Journal of parallel and distributed computing*, 6 : 115–135, 1989.

[Sun92] V. S. Sunderam. *PVM: A framework for parallel distributed computing.* Oak Ridge National Laboratory, 1992

[Sto91] B. Stolz. Unix-Workstation in lokalen Netzwerken. *Unix-Welt*, (1):44–52, 1991.

[Vie91] Th. Viessmann. Grafische Benutzeroberfläche für offene Systeme. *Unix-Welt*, (1): 22–26, 1991.

[WR92] Andreas Wierse und Martin Rumpf. GRAPE Eine objektorientierte Visualisierungs- und Numerikplattform. *Informatik Forsch. Entw.*, 7:145–151, 1992.

Eigene Vorarbeiten

[AHMP95] T. Apel, G. Haase, A. Meyer und M. Pester. Parallel solution of finite element equation systems: efficient inter-processor communication. Preprint SPC 95_5, TU Chemnitz-Zwickau, February 1995.

[Glo95a] G. Globisch. PARMESH - a parallel mesh generator. Parallel Computing, **21**(3) (1995), pp. 509–524.

[Glo95b] G. Globisch. On an automatically parallel generation technique for tetrahedral meshes. Parallel Computing, **21**(3) (1995), pp. 1979–1995.

[HHMP95] G. Haase, T. Hommel, A. Meyer, M. Pester. Bibliotheken zur Entwicklung paralleler Algorithmen. Preprint SPC 95_20, TU Chemnitz-Zwickau, Juni 1995.

[HL89a] K.-H. Hartwig und H. Liebermann. Pre- und Postprocessing für dünnwandige Bauteile. *Wiss. Z. d. TU Karl-Marx-Stadt*, 31(1): 60–65, 1989.

[HL89b] K.-H. Hartwig und H. Liebermann. Geometrische Objekte in Randrepräsentation und deren automatische Vernetzung. In *Studientexte Computergeometrie*, S. 53–63. WBZ Computermathematik, TU Dresden, Heft 106/89, 1989.

[Loh91] D. Lohse. Exakte Polytree-Repräsentation für Polyedermodelle in Rand-Repräsentation. Jahresarbeit, TU Chemnitz, 1991.

[Loh92] D. Lohse. Dynamische hierarchische Datenstrukturen für effizientes Preprocessing auf Transputertechnik. Diplomarbeit, TU Chemnitz, 1992.

[Mey95] M. Meyer. Grafik-Ausgabe vom Parallelrechner für 3D-Gebiete. Preprint SPC 95_4, TU Chemnitz-Zwickau, Januar 1995.

[Pes91a] M. Pester. Implementation und Test paralleler Basisalgorithmen der linearen Algebra. In Reinhard Grebe und Christian Ziemann, Hrsg., *Parallele Datenverarbeitung mit dem Transputer. 2. Transputer–Anwender–Treffen TAT'90. Proceedings X*, *Informatik–Fachberichte*, Bd. 272, S. 111–118. Springer–Verlag, 1991.

[Pes94] M. Pester. Grafik-Ausgabe vom Parallelrechner für 2D-Gebiete. Preprint SPC 94_24, TU Chemnitz-Zwickau, November 1994.

[Sei94] M. Seibt. Dokumentation zu NET.EXE v2.2 - Werkzeug zur Eingabe und Bearbeitung 2-dimensionaler Netze. Belegarbeit, TU Chemnitz-Zwickau, Januar 1994.

2.4 Ergebnisse

2.4.1 Teilaufgabe Entwicklungsumgebung

Als Entwicklungsumgebung für parallele Algorithmen stand im Berichtszeitraum aufgrund der Hardwareausstattung der TU Chemnitz insbesondere Parix im Vordergrund, sowie Unix als lokale Arbeitsumgebung der einzelnen Benutzer. Durch die modulare Struktur der am häufigsten eingesetzten Testsoftware mit einer nicht system-, sondern problembezogenen Kommunikationsschnittstelle ist es gelungen, daß die einzelnen Anwender ohne Manipulation an Quelltexten zwischen Parix, PVM und MPI als Kommunikationsbasis ebenso wechseln können wie zwischen den (homogenen) Parallelrechnern GCPowerPlus oder GC-el und den (heterogenen) Workstation-Clustern, sei es HP-, Sun-, SGI- oder Pentium-Architektur [Pes97a]. Die Basissoftware ist dabei stets bezüglich der Prozessoranzahl skalierbar, was insbesondere zur Folge hat, daß kleinere Tests auch an einer Workstation, d. h. auf einem einzelnen Prozessor, ausführbar sind.

Die Modulbasis zur numerischen Simulation hat damit etwa den in Abb. 2.1 dargestellten hierarchischen Aufbau, wobei der gestrichelt eingerahmte Teil die wesentlichen algorithmischen Bestandteile enthält, die von der konkreten Parallelrechnerschnittstelle unabhängig sind.

Wichtige Entwicklungen zur Integration neuer Rechnerarchitekturen und zur effizienten Nutzung von MPI für FEM-spezifische Kommunikationsabläufe wurden in Zusammenarbeit mit Projektbereich B geleistet [GEW97, Gra97].

Zahlreiche Hilfsmittel erleichtern die Beurteilung von Algorithmen. So enthalten alle Kommunikationsroutinen akkumulierende Zeitmessungen, wodurch für bestimmte Teile der Simulationsrechnung eine einfache Analyse der Verteilung der Gesamtrechenzeit bezüglich Kommunikation und Arithmetik auf den einzelnen Prozessoren möglich ist. Durch die grafische Darstellung dieser Information wird ein optimaler und schnell erfaßbarer Eindruck über die Effizienz der Parallelisierung vermit-

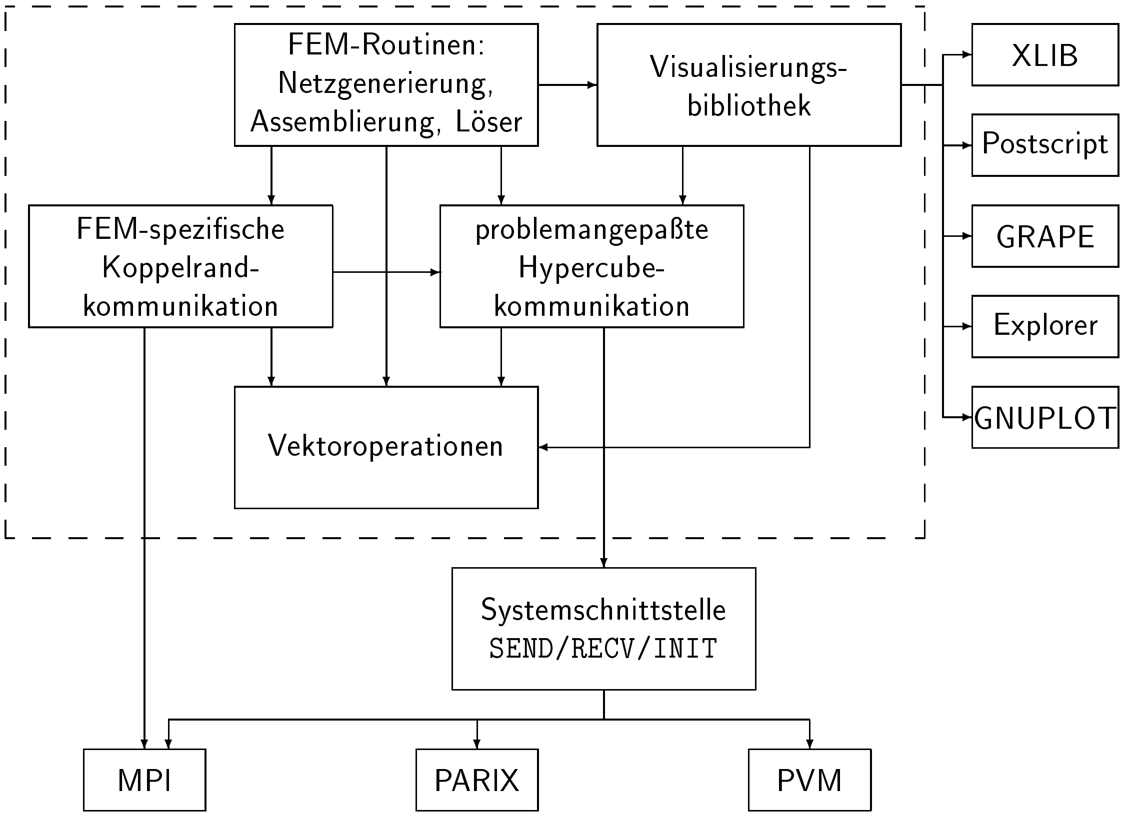


Abbildung 2.1: Bibliotheksstruktur für Entwicklung und Test paralleler Algorithmen

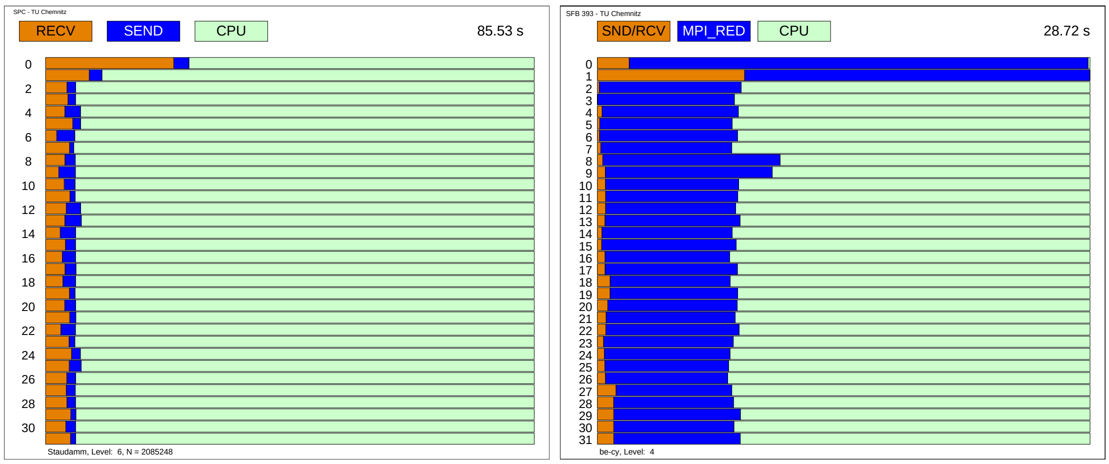


Abbildung 2.2: Grafische Auswertung der Effizienz von Algorithmen auf dem Parallelrechner, Kommunikationszeiten hier dunkel wiedergegeben

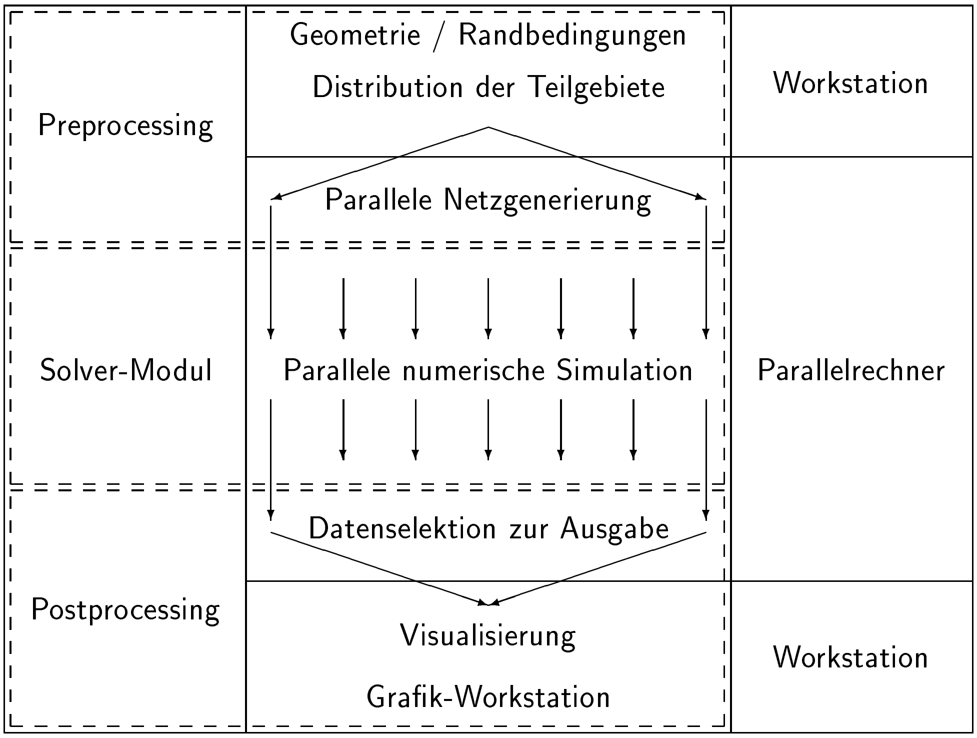


Abbildung 2.3: Aufgabenverteilung zwischen Workstation und Parallelrechner

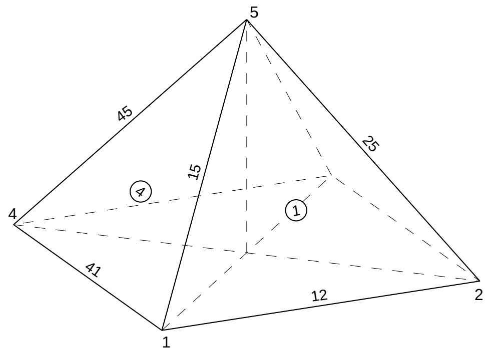
telt. In Abbildung 2.2 sind zugleich die typischen Unterschiede bei 2D- (links) und 3D-Rechnung (rechts) hinsichtlich Kommunikationsanteil zu erkennen. Der höhere Kommunikationsanteil bei dreidimensionalen Problemen ergibt sich aus der um eine Größenordnung höheren Anzahl von Koppelknoten in Relation zur Anzahl der inneren Knoten, die den Anteil der (parallelen) Arithmetik bestimmen.

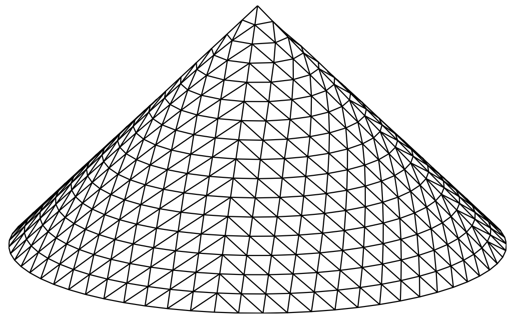
Die spezifischen Probleme des Pre- und Postprocessing für numerische Simulationen auf MIMD-Rechnern mit verteiltem Speicher werden aus Abb. 2.3 deutlich [Pes96]. Aus Gründen der Effizienz auf dem Parallelrechner macht sich eine zusätzliche Schnittstelle innerhalb des Preprocessings notwendig, die bei herkömmlicher Arbeitsweise so nicht vorkommt. Zu einem möglichst frühen Zeitpunkt ist das geometriebeschreibende Netz auf die Prozessoren des Parallelrechners zu verteilen, so daß die Generierung des FEM-Netzes bereits durch vollständig parallele Verfeinerung in den einzelnen Teilgebieten erfolgt. Eine ähnliche Situation tritt innerhalb des Abschnitts Postprocessing auf. Um eine nachträgliche Auswertung von Ergebnissen mit allgemein verfügbaren Grafikwerkzeugen zu ermöglichen, müßte bei jeder Rechnung die gesamte Datenmenge von allen Prozessoren als File abgelegt, gegebenenfalls in ein passendes Format umgewandelt und dann an dem entsprechenden Arbeitsplatz wieder eingelesene werden. Der effektivere Weg besteht darin, bereits zu dem Zeitpunkt, wo die Daten noch verteilt im Speicher der einzelnen Prozessoren vorliegen, die jeweils interessierenden Teile auszuwählen und möglichst sofort zur Darstellung zu bringen, so daß interaktiv auch schnell andere Informationen vom Parallerechner abgerufen werden können, ohne damit das Filesystem unnötig zu belasten.

2.4.2 Teilaufgabe Preprocessing

Eine erste Teilaufgabe bestand in der Festlegung von Datenstrukturen, die flexibel genug sind, für verschiedene Testfälle oder für künftige Anforderungen eingesetzt zu werden. Somit bestand diese Aufgabenstellung gemeinsam für verschiedene Teilprojekte, vor allem Tp. A3, D1 und D3. Die typische Arbeitsweise bei der Algorithmenentwicklung für die numerische Simulation besteht darin, daß häufig gleiche oder ähnliche Testbeispiele verwendet werden, um vergleichende Untersuchungen anstellen zu können.

So ist es klar, daß eine wesentliche Schnittstelle ein Datenfile mit Standardinformationen zur Beschreibung der Geometrie und Randbedingungen des jeweiligen





```
  ...
#VERTEX: 6
1   0.0  -1.0   0.0          Knotenname, x, y, z-Koordinaten
  ...
6   0.0   0.0   0.0
#EDGE: 13
12  0   1  2                 Kantenname, Typ, Knotennamen
23  0   2  3
  ...
56  0   5  6
#FACE: 12
1   2  3   12 25 15          Facename, Typ, Anzahl, Kantennamen
  ...
12  0  3   46 45 56
#SOLID: 4
1   1  4   1 5 9 10          Elementname, Typ, Anzahl, Facenamen
  ...
#FACE_GEO: 1
2  31  7   0. 0. 1.  0. 0. 0.  1.
  ...
```

Abbildung 2.4: Illustration zum Aufbau des Standardeingabefiles

Testbeispiels ist. Die Einleseroutine dafür könnte zwar unmittelbar durch ein interaktives Programm zur (grafischen oder numerischen) Eingabe dieser Informationen ersetzt werden, aber der Bedarf hierzu ist nicht vordergründig vorhanden - auch angesichts der unnötigen Belastung des Parallelrechners mit solchen Aufgaben. Vorzuziehen ist in dieser Hinsicht eine integrierte Benutzeroberfläche für verteilte Systeme, bei der ein Menüpunkt „Preprocessing“ die interaktive Geometrieeingabe oder -modifikation als Einzelprozeß auf einer Workstation startet, während anschließend der Berechnungsmodul als Parallelrechnerapplikation läuft, wobei die Daten wie bisher eingelesen werden, entweder nach Ablage in einem File (geringer Größe) oder durch direkte Netzverbindung zum Preprozessor auf der Workstation. Zu dieser Problematik bestehen weiterhin Kontakte mit Prof. W. Mach (Schmalkalden), bei denen eine Web-basierte Problemlösung auf verteilten Systemen angestrebt wird.

Die Struktur des Standard-Eingabefiles für 3D-Geometrien wurde durch Verwendung von Schlüsselworten und ein hinsichtlich des inneren Aufbaus relativ großzügiges freies Format lesbar und editierbar gestaltet [Loh98].
Unter den Schlüsselworten `#VERTEX`, `#EDGE`, `#FACE`, `#SOLID` wird der topologische Zusammenhang des dreidimensionalen Gebildes beschrieben. Durch symbolische Referenzen auf „Namen“ von Objekten der nächst niedrigeren Dimension gibt es keine Abhängigkeiten von der Reihenfolge der Auflistung der einzelnen Strukturen. Die durch jeweils ein Schlüsselwort eingeleiteten Eingabeblöcke können auch mehrfach vorkommen. Auszugsweise zeigt das Beispiel (Abb. 2.4) dieses Format.

Eine Typangabe verknüpft beispielsweise ein Volumenelement mit einer Materialparameterdefinition oder ein Face mit der Definition einer Flächengeometrie, in obigem Beispiel wird durch die Codierung nach `#FACE_GEO` dem Face-Typ 2 ein Kegelmantel zugeordnet. Diese Face-Geometriebeschreibung wird bei der lokalen Netzverfeinerung ausgewertet und ermöglicht somit eine grobe Anfangsvernetzung auch bei relativ komplizierten Geometrien. Details sind in [Pes97] dokumentiert. Eine Sammlung von Beispielen zur Geometrieanpassung findet man im WWW unter [Pes97c]. Randbedingungen werden unter den Schlüsselworten `#DIRICHLET` bzw. `#NEUMANN` in weiteren Blöcken für einzelne Rand-Faces angegeben. Dabei ist die Angabe konstanter oder linearer Funktionen möglich oder der Verweis auf eine im Programm aufzurufende spezielle Funktion. Beispielsweise würde folgende Funktion eine Punktlast in Richtung der z-Koordinate bei $x = y = 0$ mit exponentiellem Abklingverhalten simulieren (s. Abb. 2.5):

```
DOUBLE PRECISION FUNCTION G(IFG,X,Y,Z)
IMPLICIT NONE
DOUBLE PRECISION FUNCTION X,Y,Z ! Koordinaten
INTEGER IFG ! Nr. des Freiheitsgrades
G=0.0D0
IF (IFG .EQ. 3) G = -2.917857D0 * EXP(-2D0*(X*X+Y*Y))
RETURN
END
```

In den letzten Jahren entstand eine Reihe nützlicher Werkzeuge zur Bearbeitung solcher Standard-Eingabefiles. Dazu gehört vor allem das Programm `xbc` zur Eingabe von Randbedingungen mit grafischer Unterstützung, indem Randfaces per

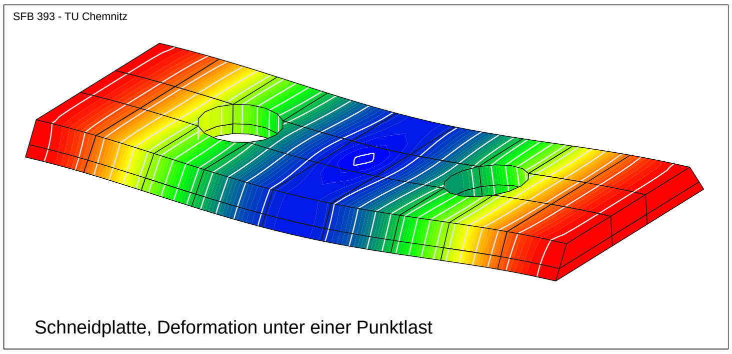


Abbildung 2.5: Simulation einer Punktlast

Maus-Click selektiert werden können. Ein weiteres Programm `geo_trans` erlaubt die Konstruktion neuer Netze auf der Basis vorhandener Standard-Eingabefiles durch geometrische Elementartransformationen und eine Merge-Funktion zum Verknüpfen verschiedener Teilkörper. Abbildung 2.6 zeigt Beispiele solcher Geometrien, die sich auf diese Art aus relativ wenigen „Bauteilen“ durch Spiegelung, Drehung und Verschiebung konstruieren lassen.

Ebenso wurden in Zusammenarbeit mit J. Schöberl und F. Kickinger (Universität Linz) Experimente zur Anwendung allgemeinerer 3D-Netzgeneratoren ([Sch95], [Kic96]) durchgeführt, indem eine Anpassung der Ausgabedaten an das beschriebene Fileformat vorgenommen wurde. Mit dem Ziel eines möglichst groben Startnetzes kommt man jedoch bei solchen von der Oberflächenvernetzung ausgehenden Verfahren nicht immer zu brauchbaren Ergebnissen. Eine gewisse Mindestzahl von Startelementen ist erforderlich, die schon nahe an der Feinheit des Rechennetzes liegt, so daß der Parallelisierungseffekt bei der Netzgenerierung verlorengeht.

Zur Verarbeitung der in diesem Standard-Format gegebenen Nutzerdaten werden diese von einer speziellen Einleseroutine aufbereitet. Auch dabei besteht wiederum ein sehr freies Format hinsichtlich der internen Darstellung der aus dem Eingabefile gewonnen Informationen. Mit Hilfe eines Formatstrings wird der Leseroutine vorgegeben, welche Daten des Eingabefiles zu selektieren sind und in welcher Reihenfolge sie im Arbeitsspeicher abgelegt werden sollen. Betrachten wir als Beispiel den folgenden Aufruf:

```
     dformat= 'G(type+dataf:8) '//
    *         'V(x+y+z) '//
    *         'D(name+info(type+dataf:4)) '//
    *         'N(name+info(type+dataf:5)) '//
    *         'Sdataf:6 '//
    *         'Stype '//
    *         'Edataf:2:3 '//
    *         'F(dataf:4+type:1)')

     call data_read(dfile,datformat,feld,offset,counter,...)
```

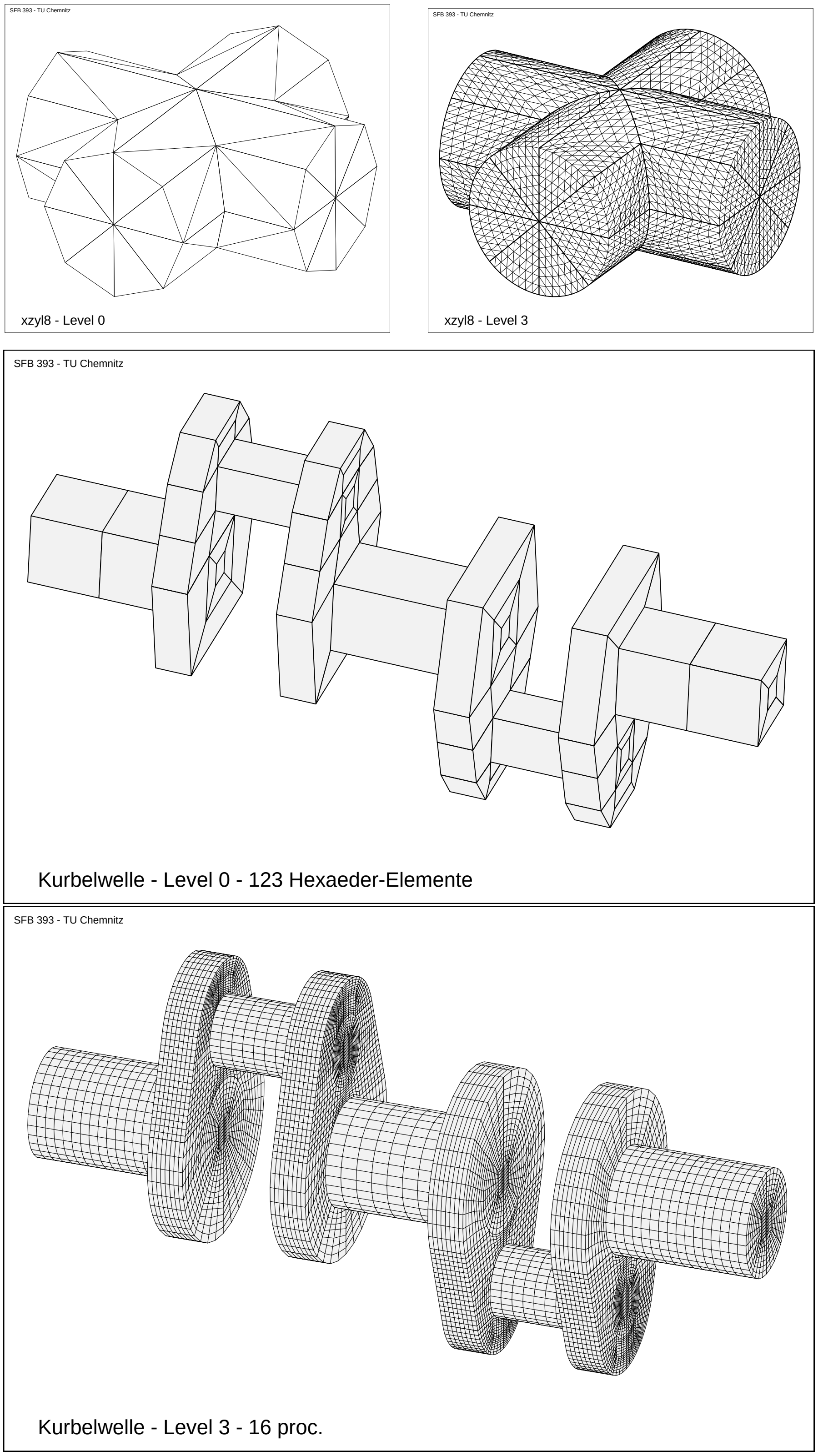


Abbildung 2.6: Beispiele zur automatischen Geometrieanpassung durch Netzverfeinerung

Damit werden auf `feld` in aufeinanderfolgenden Blöcken die Informationen aus dem Eingabefile bereitgestellt:

- Geometrieparameter mit insgesamt je 9 Einträgen,
- Knotenkoordinaten (x, y, z),
- Definitionen der Dirichlet- bzw. Neumann-Randbedingungen,
- Liste von Hexaeder-Elementen mit je 6 Einträgen (Zeiger auf Faces),
- Materialzuordnungen der einzelnen Elemente, sowie
- Kanten- und Face-Listen.

Startindex und Länge der einzelnen Teilfelder gehen aus den Indexfeldern `offset` und `counter` hervor.

Das Einlesen des Grobnetzes erfolgt durch einen Prozessor des Parallelrechners, der es dann mittels Broadcast-Funktion (Tree_Down im Hypercube) an alle anderen Prozessoren verteilt. Jeder Prozessor arbeitet von nun an lokal auf seinem Teilgebiet des Ausgangsnetzes. Die Zuordnung der Teilgebiete zu den Prozessoren, also die Verteilungsstrategie, kann die Effizienz der nachfolgenden Simulationsrechnung entscheidend beeinflussen. Neben der angestrebten gleichmäßigen Lastverteilung in Bezug auf arithmetischen Aufwand hat insbesondere die konkrete Zuordnung von Teilgebieten zu bestimmten Prozessoren Auswirkungen auf den Umfang der Koppelränder und damit auf den Kommunikationsbedarf während der Simulation. Es wurden verschiedene Strategien zur Verteilung des Grobnetzes angewendet, die durch Programm-Parameter auswählbar sind:

- Lineare Verteilung der Elemente,
- Verteilung durch spektrale Bisektion,
- Externe Festlegung mit modifizierten Spektralmethoden.

Bei der einfachsten Variante, der *linearen Verteilung*, bestimmt die Reihenfolge der Definition der Elemente im Eingabefile ihre Zuordnung zu den Prozessoren. Dabei wird auch der Aspekt berücksichtigt, daß Prozessor 0 des Hypercubes eventuell einige Sonderaufgaben zu erfüllen hat (z. B. Grobgitterlöser). Ist die Anzahl N der Elemente des zu verteilenden Grobgitters durch die Anzahl p der Prozessoren teilbar, so ist klar, daß jeder Prozessor die gleiche Anzahl N/p erhält. Anderenfalls wird der Prozessor 0 weitgehend entlastet, indem die Prozessoren $p-1, p-2, \ldots$ jeweils $\left\lceil \frac{N}{p} \right\rceil$ Elemente erhalten. So bleibt für Prozessor 0 ein Teilgebiet mit weniger Elementen, eventuell auch gar keines (vgl. Abb. 2.2). Bei dieser Verteilungsvariante sind alle a-priori durchführbaren Abschätzungen zur Lastverteilung bereits bei der Erstellung des Eingabefiles zu berücksichtigen. Dennoch ist auf diese Art im allgemeinen keine Anordnung der Elemente möglich, die für beliebige Prozessorzahlen eine optimale lineare Verteilung zur Folge hat.

Als Alternative ist eine Verteilung nach der Methode der *rekursiven Spektralbisektion* [Fie75, DR94, Rei96, Els97] möglich, die über einen Eigenvektor der Adjazenzmatrix des Gebietszusammenhangsgraphen eine Aufteilung in jeweils zwei Teilgebiete vornimmt. Betrachtet man einen Schritt dieser Methode, so liefert sie meist eine gute Lösung (Abb. 2.7), d. h. zwei zusammenhängende Teilgebiete mit

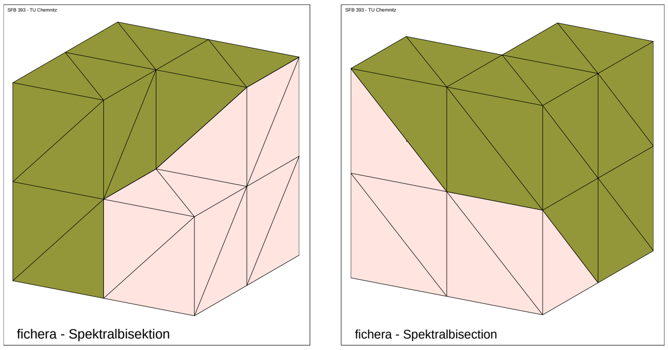


Abbildung 2.7: Ergebnis einer einfachen spektralen Bisektion (zwei verschiedene Ansichten desselben Gebietes)

geringem Koppelrand. Das ist jedoch im allgemeinen schon nicht mehr gewährleistet, wenn man das Ergebnis nach mehreren Rekursionsschritten betrachtet, was durch einfache Beispiele in [Rei96] illustriert wird.

Weitere Verbesserungen ergeben sich in der dritten Variante mit der Einführung verschiedener Gewichtsfaktoren, durch die vor allem der zu erwartende Kommunikationsaufwand an den Koppelrändern sowie die Länge der Kommunikationswege (Anzahl der *Hops*) minimiert werden sollen. Solche verbesserten spektralen Algorithmen werden durch das Programm `chaco` [HL92, HL94] realisiert. Die dabei zu treffenden Entscheidungen sind auch von der jeweiligen Kommunikationstopologie des Parallelrechners abhängig. Wegen der begrenzten Verfügbarkeit des Programms muß dieses als separater Preprocessing-Schritt auf einer Workstation laufen. Es berechnet zu einem gegebenen Eingabefile für eine bestimmte Prozessoranzahl die optimale Verteilung. Die Verteilungsliste kann als zusätzlicher Input am Parallelrechner zur Anfangsverteilung des Gebietes verwendet werden. Die positiven Auswirkungen dieses Verfahrens auf die Laufzeit der Algorithmen zur numerischen Simulation sind teilweise beachtlich, ein besonders drastisches Beispiel wurde bereits in Teilprojekt D3 genannt (vgl. Tabelle 2.1 in Tp. D3, S. 261).

Bei der Betrachtung der Verteilung der Rechenzeiten (Abb. 2.8) wird deutlich, daß nicht nur die *Kommunikationen* über die Koppelränder verantwortlich für die hohen Zeiten bei ungünstiger Netzverteilung sind, sondern auch die damit zusammenhängenden *CPU-Operationen* deutlich anwachsen.

Dies läßt sich mit der in [AHMP95] dargelegten Vorgehensweise der Koppelrandkommunikation begründen, die für 2D-Probleme meist sehr gute Ergebnisse liefert, aber im 3D-Bereich schneller zu den oben genannten Effekten führt. Zum einen ist die Anzahl der Koppelknoten im Verhältnis zur Gesamtknotenzahl wesentlich größer, zum anderen sind bei Koppelkanten wesentlich mehr Prozessoren an der Kommunikation beteiligt. Mit der Strategie, jeweils einen minimalen Sub-Hypercube zu bilden, in dem alle an einer Kante beteiligten Prozessoren enthalten sind, kann bei ungünstiger Gebietszerlegung der gesamte Hypercube an der Kommunikation beteiligt sein, während bei günstiger Verteilung mehrere Sub-Hypercubes gleichzeitig und unabhängig voneinander arbeiten können.

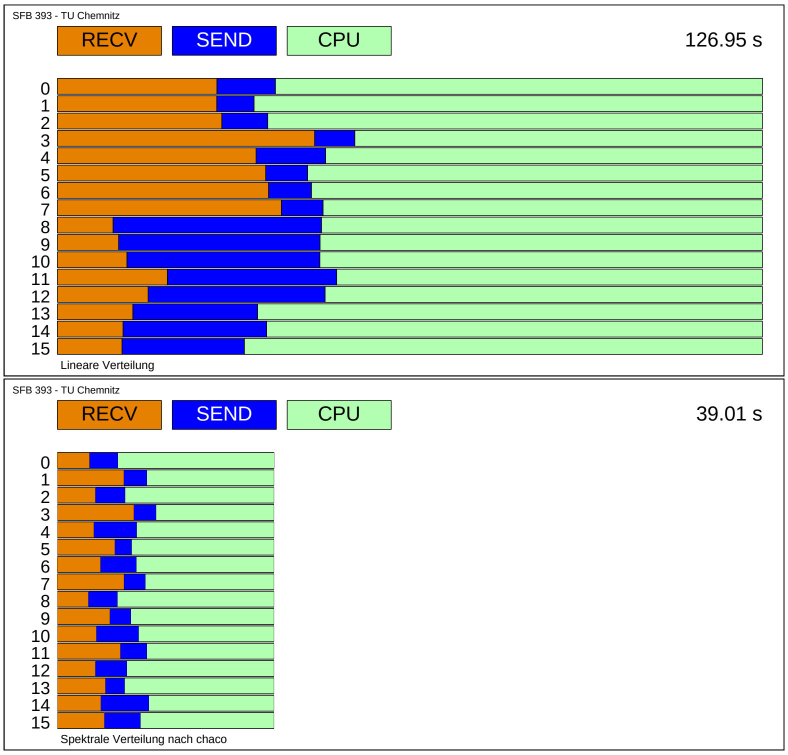


Abbildung 2.8: Gegenüberstellung der Rechenzeiten bei unterschiedlicher Verteilungsstrategie (Bsp. aus Tp. D3)

Unter MPI besteht mit der Möglichkeit zur Bildung von Kommunikatoren für beliebige Prozessorgruppen eine gute Voraussetzung zur Verbesserung der gesamten Koppelrandkommunikation durch noch stärkere Ausnutzung der Parallelität bei diesem Vorgang [GEW97]. Abhängig vom konkreten Verteilungszustand eines Gebietes sind disjunkte Prozessorgruppen (mit jeweils gemeinsamer Koppelkante) so zu finden, daß in möglichst wenigen, parallel ausführbaren Kommunikationsschritten die gesamte FEM-Akkumulation über den Koppelrändern bewältigt werden kann. Erste Experimente berechtigen zu vorsichtigem Optimismus.

2.4.3 Teilaufgabe Postprocessing

Auf dem Gebiet des Postprocessing, insbesondere der Visualisierung von Ergebnissen direkt vom Parallelrechner aus, konnte auf einigen zurückliegenden Ergebnissen aufgebaut werden. Für die Visualisierung dreidimensionaler FEM-Gebiete wurden mehrere Varianten verfolgt:

(1) Übergabe der 3D-FEM-Daten an einen externen Postprocessor;
(2) parallele Aufbereitung der Pixelinformation bis hin zu einer X11-basierten Schnittstelle (Zeichnen und Füllen von Polygonen);
(3) Speicherung der Daten in einem File für eine *post-mortem*-Visualisierung.

Bei Variante (1) steht als externer Postprocessor das im SFB 256 entwickelte Visualisierungspaket GRAPE zur Verfügung [WR92]. Bereits zu Beginn des Berichts-

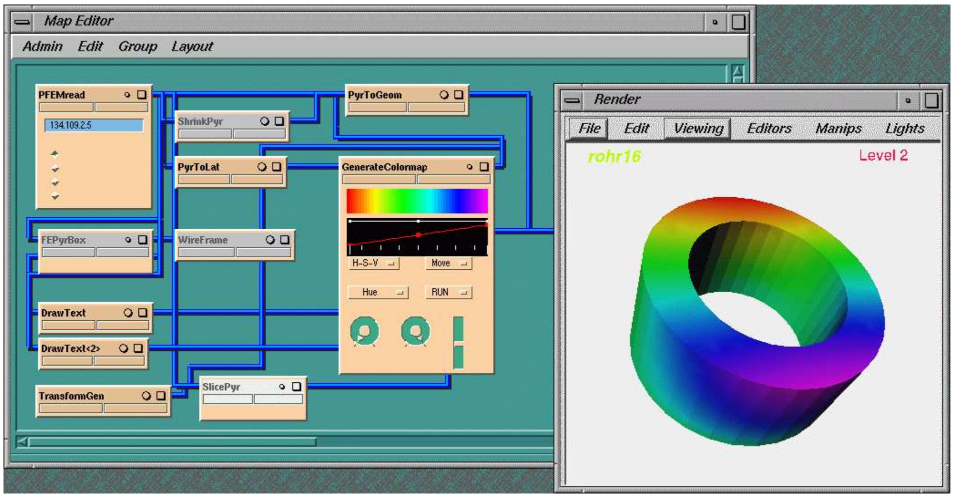


Abbildung 2.9: Nutzung des IRIS Explorers zum Postprocessing, (Client-Server-Schnittstelle im Modul links oben)

zeitraums wurde eine Schnittstelle vom Parallelrechner zu diesem Programm geschaffen (z.B. Abb. 2.10 in Tp. A3, S. 69). Mit einem einfachen, über Sockets realisierten Protokoll lassen sich an der Grafikworkstation die Netz- und Lösungsdaten vom Parallelrechner interaktiv abrufen. Der Vorteil besteht darin, daß das Parallelrechnerprogramm nach der Übergabe der Daten (im Hintergrund) weiterarbeitet, während der Nutzer mit den Möglichkeiten des Grafikprogramms experimentiert, um eine geeignete Lösungsdarstellung zu finden. Als Nachteil stellt sich heraus, daß dieses Grafikprogramm nur an leistungsstarken Grafikworkstations sinnvoll genutzt werden kann, die i.a. nicht als Arbeitsplatzrechner bereitstehen. An Arbeitsplatzrechnern der mittleren Leistungsklasse können nur grobe Vernetzungen in vertretbarer Reaktionszeit visualisiert werden.

Ähnliche Aussagen gelten auch für Variante (3). Hier ist es vor allem der hohe Speicherbedarf, sowohl für die Datenfiles auf Festplatte, als auch zur Visualisierung an der Grafikworkstation. Zum Beispiel wurden Experimente mit dem Iris Explorer [SGI95] an einer Silicon Graphics Workstation durchgeführt. Das komfortable und modular konfigurierbare Grafikprogramm bietet dem Nutzer eine Vielzahl von Möglichkeiten zum Postprocessing. Aber bereits bei einer FEM-Knotenzahl von etwa 60 000 traten Probleme mit dem Arbeitsspeicher auf (bei einer Ausstattung mit 256 MB RAM).

Schließlich wurde für diese Maschine eine auf dem Client-Server-Modell basierende Methode zur direkten Datenübergabe an den Iris-Explorer ohne Zwischenspeicherung (ähnlich Variante (1)) realisiert. Dabei wird das Programm am Parallelrechner zum Compute-Server der Explorer zum Grafik-Server und als (Mini-)Client arbeitet ein in den Explorer eingefügter Schnittstellenmodul (Abb. 2.9), über den der Nutzer die vom Compute-Server zu übertragenden und dann im *Render*-Modul darzustellenden Daten auswählt. Diese Arbeiten wurden durch die über die DFG finanzierte Anschaffung einer Grafikworkstation O2 (Silicon Graphics) ermöglicht.

Damit ist die prinzipielle Lösung der Anbindung kommerzieller Tools mit entsprechender Bedienoberfläche an einen auf dem Parallelrechner laufenden Compute-Server gelungen. Die direkte Schnittstelle zwischen Grafikmaschine und Compute-Server bringt hier zwar den erhofften Vorteil hinsichtlich des Plattenspeichers, aber nicht in bezug auf die Leistung der Grafikmaschine selbst, die bei sehr feiner Vernetzung des Gebietes schnell überfordert werden kann.

Wegen der geschilderten Performance-Nachteile mit komfortableren, aber Workstation-basierten Grafik-Programmen wurde Variante (2) als eine schnell und an jeder Workstation einsetzbare Visualisierungsplattform weitergeführt. Die Leistungsmerkmale dieser Variante sind:

- Einzige Grafik-Schnittstelle ist die X11-Bibliothek, damit ist die Visualisierung ohne spezielle Anforderungen an den Arbeitsplatzrechner möglich.

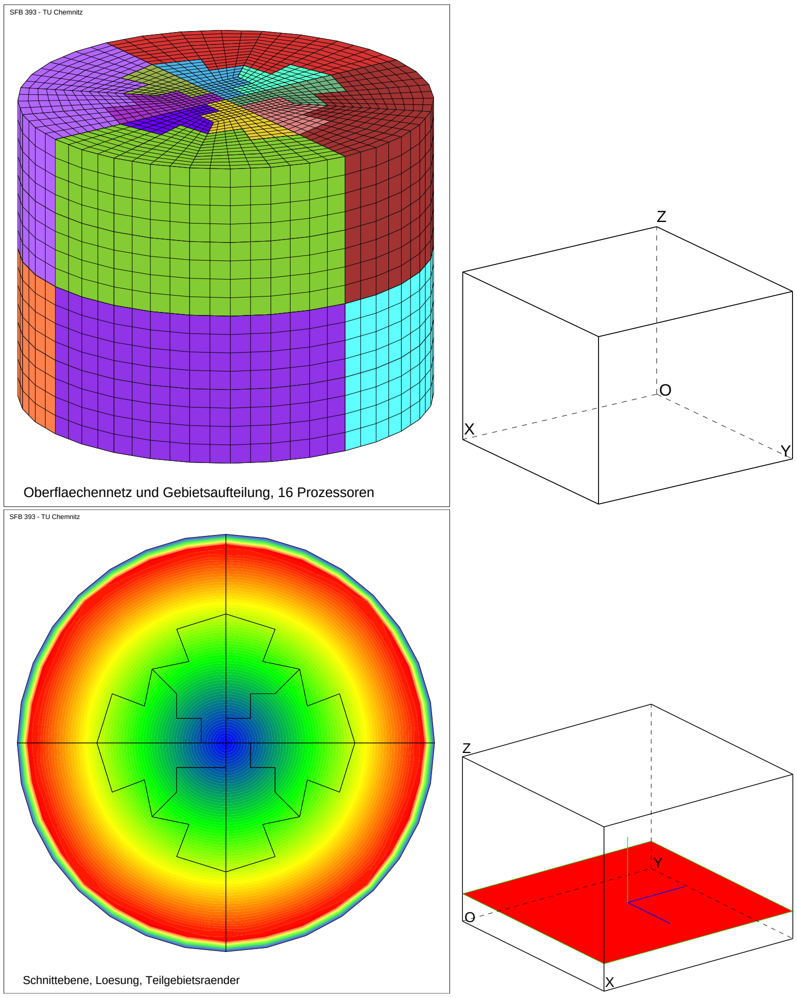


Abbildung 2.10: Darstellung von 3D-Gebieten als Oberfläche oder Schnittebene, hier wurden jeweils die Teilgebietsgrenzen mit eingezeichnet; rechts: die jeweilige Orientierungshilfe zur Auswahl der gewünschten Darstellung

- Ein geringer Nachteil davon ist eine relativ primitive Nutzerschnittstelle, teils über ein grafisches Menü, teils über Texteingaben.
- Die Auswertung der berechneten Daten bis hin zur Bildinformation erfolgt auf dem Parallelrechner, dort wo die Daten entstanden sind.
- Die Darstellungsroutinen sind derart modular aufgebaut, daß sich die 3D-Visualisierung auf die Routinen der 2D-Visualisierung zurückführen läßt.
- Der Grundbaustein ist eine Unterprogrammbibliothek zur Visualisierung von 2D-FEM-Netzen und darauf berechneten Lösungen (Anzahl und Bezeichnung der einzelnen Freiheitsgrade bzw. abgeleiteter Größen sind beliebig vorgebbar). Die Darstellungsmöglichkeiten umfassen dabei u.a.:
  - Netzdarstellung, farbige Hervorhebung von Kanten mit Randbedingungen,
  - Isoliniendarstellung einzelner Freiheitsgrade,
  - Einfärbung des Gebietes entsprechend eines Freiheitsgrades,
  - vektorielle Darstellung von Geschwindigkeits- oder Verschiebungskomponenten,
  - Lösungsdarstellung als Gebirge über dem 2D-Netz,
  - farbige Anzeige verschiedener Materialbereiche oder der Zuordnung zu den Prozessoren,
  - Beschränkung der Ausgabe auf einzelne Prozessoren,
  - alle Darstellungen auch als Postscript, wobei eine gewisse Optimierung den Umfang der Postscript-Dateien minimiert,
  - Umschaltung auf *Dauerbetrieb*, d.h. die zuletzt gewählte Darstellungsart wird bei erneutem Aufruf des Programms ohne Rückfrage wieder gewählt (Verfolgung des zeitlichen Verlaufs einer Größe bei Strömungssimulationen).
- Zur Visualisierung von 3D-Gebieten wählt der Nutzer interaktiv zunächst eine Darstellungsform (Schnittebene oder Oberflächendarstellung) und entsprechende Parameter wie Normalenvektor der Bildebene bzw. Blickrichtung (Abb. 2.10) u. a., danach wird die Projektion des 3D-Gebietes als 2D-Netz mit dem o. g. Modul zur Darstellung gebracht.
- vektorielle Lösungskomponenten werden zusätzlich auf die lokalen Koordinaten der Schnittebene transformiert, wodurch das Lösungsverhalten innerhalb der gewählten Schnittebene besser beurteilt werden kann.
- Die Darstellung von 3D-Oberflächennetzen erfolgt mit sehr eingeschränkter Berücksichtigung von Verdeckungen, so daß zwar bei konvexen Körpern eine korrekte Darstellung gewährleistet ist, in anderen Fällen aber verdeckte Flächen teilweise als sichtbar erscheinen. Auf einen echten Hidden-Line-Algorithmus auf dem Parallelrechner wurde aus Effizienzgründen verzichtet. Er würde erhebliche Kommunikationen erfordern, um lokale und globale Verdeckungen zu bestimmen, und außer einem *ästhetischen* Aspekt keine wesentliche Funktion erfüllen.

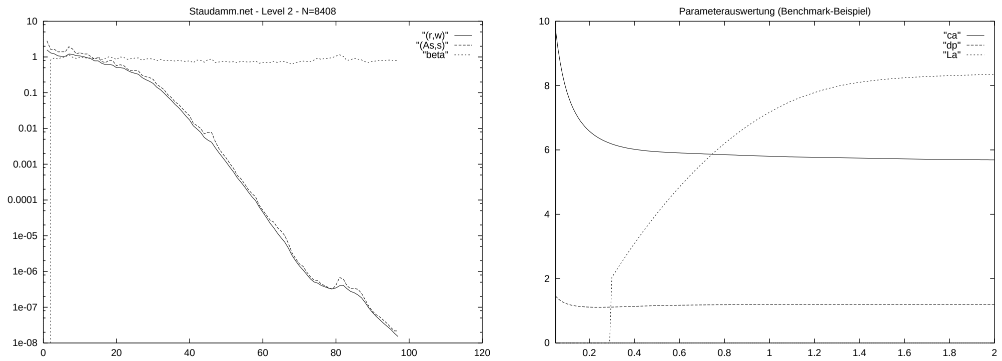


Abbildung 2.11: Anwendung der GNUPLOT-Schnittstelle, am Beispiel des Konvergenzverhaltens eines iterativen Lösers oder der Auswertung von Auftriebs- und Druckbeiwert, sowie Nachlauflänge bei einem Benchmark-Test

Da insbesondere bei Strömungssimulationen ganze Zeitabläufe einzelner Lösungskomponenten interessieren, wurde ein Hilfsmittel zur automatischen Generierung von Videosequenzen geschaffen und in die Visualisierungsvariante (2) integriert. Dabei wird jeweils eine Darstellung aus dem X-Window vom Bildschirm in eine Bitmap-Datei (GIF-Format) kopiert. Diese Aufgabe übernimmt ein separates, an einer Workstation laufendes Programm, das als studentische Arbeit entwickelt wurde. Auf ein Signal hin, das vom Parallelrechnerprogramm nach vollendeter Visualisierung eines konkreten Zustandes gesendet wird, wird der Fensterinhalt mittels `grab`-Funktion kopiert und als Datei gespeichert. Währenddessen kann das Programm auf dem Parallelrechner bereits weiterarbeiten und auch schon neue grafische Darstellungen erzeugen. Eine Synchronisation erfolgt zum Zeitpunkt der Absendung des erwähnten Signals vom Parallelrechner.

Die so erzeugten Einzelbilder können später mit bekannten Grafikwerkzeugen beispielsweise ins MPEG-Format überführt werden oder zu animierten GIF-Dateien zusammengefaßt (Beispiele im WWW unter [MP97, Pes97b]).

Bei der Untersuchung dreidimensionaler zeitlich veränderlicher Abläufe ist klar, daß im allgemeinen mit nur einer Ansicht (z. B. einer speziellen Schnittebene) kein ausreichender Gesamteindruck des dreidimensionalen Lösungsverhaltens entsteht. Simulationsrechnungen können auch auf dem Parallelrechner durchaus mehrere Stunden oder Tage in Anspruch nehmen. Deshalb wurde es notwendig, nicht nur eine, sondern mehrere Darstellungen je Zeitschritt auch automatisch abspeichern zu können, z. B. Schnittebenen in verschiedenen Raumrichtungen oder verschiedenen Abständen. Damit ist es nicht mehr erforderlich, den gesamten Simulationsprozeß mehrfach zu wiederholen, um die verschiedenen Bildfolgen zu generieren.

Eine zusätzliche Funktionalität zur Auswertung verschiedener diskreter Parameter während einer Simulationsrechnung wurde durch die Kopplung mit dem Programm GNUPLOT geschaffen (Abb. 2.11). Ein GNUPLOT-Rahmenprogramm wird separat auf einer Workstation gestartet und interpretiert Kommandos, die vom Simulationsprogramm auf dem Parallelrechner per Unterprogrammaufruf und Socket-Stream übermittelt werden.

Literaturverzeichnis / Eigene Veröffentlichungen im Berichtszeitraum

[Els97] U. Elsner. Graph partitioning - a survey. Preprint SFB393/97-27, TU Chemnitz, December 1997.

[GEW97] L. Grabowsky, Th. Ermer, and J. Werner. Nutzung von MPI für parallele FEM-Systeme. Preprint SFB393/97-08, TU Chemnitz-Zwickau, März 1997.

[Gra97] L. Grabowsky. MPI-basierte Koppelrandkommunikation und Einfluß der Partitionierung im 3D-Fall. Preprint SFB393/97-17, TU Chemnitz, August 1997.

[Loh98] D. Lohse. Ein Standard-File für 3D-Gebietsbeschreibungen - Definition des Fileformates V 2.1. Preprint SFB393/98-11, TU Chemnitz, April 1998.

[Pes96] M. Pester. On-line visualization in parallel computations. In: W. Borchers, G. Domick, D. Kröner, R. Rautmann, D. Saupe (eds.), *Visualization Methods in High Performance Computing and Flow Simulation*, VSP Utrecht, The Netherlands / TEV Vilnius, Lithuania, 1996. pp. 91–98.

[Pes97] M. Pester. Behandlung gekrümmter Oberflächen in einem 3D-FEM-Programm für Parallelrechner. Preprint SFB393/97-10, TU Chemnitz-Zwickau, April 1997. `http://www.tu-chemnitz.de/sfb393/ftp/sfb97-10.ps.gz`.

[Rei96] U. Reichel. Partitionierung von Finite-Elemente-Netzen. Preprint SFB393/96-18, TU Chemnitz-Zwickau, November 1996.

[MP97] S. Meinel, M. Pester. Examples for animated solutions of flow simulations. `http://www.tu-chemnitz.de/~pester/exmpls.html`

[Pes97a] M. Pester Bibliotheken zur Entwicklung paralleler Algorithmen. `http://www.tu-chemnitz.de/~pester/par_lib.html`

[Pes97b] M. Pester. Animated GIFs of numerical solutions. `http://www.tu-chemnitz.de/~pester/exmpls/index.html`

[Pes97c] M. Pester. Examples for special face geometry by mesh refinement. `http://www.tu-chemnitz.de/~pester/facegeo.html`

2.5 Offene Fragen/Ausblick

Ein bisher ungelöstes Problem des Preprocessings für die im SFB zu behandelnden Experimentalgebiete (teils akademisch, teils anwendungsnah) ist die Bereitstellung geeigneter grafisch unterstützter 3D-Konstruktionstools. Um hierfür kommerzielle Pakete nutzen zu können, mit denen eine CAD-Funktionalität bereitgestellt werden kann, sind in Zukunft entsprechende Anpassungen der Datenstrukturen, z. B. über Hilfsassistententätigkeit zu realisieren. Finanzielle Mittel für die Lizenzierung kommerzieller Pakete sind einzuplanen.

Die bisherigen Erfahrungen des parallelen Postprocessings, bzw. des Postprocessings für parallele numerische Simulationen bestätigen, daß man künftig Abstand davon nehmen muß, alle auf dem Parallelrechner berechneten, an die FEM-Knotenpunkte gebundenen Lösungen auch zur Visualisierung zu verwenden. Bei hoher

Prozessoranzahl und großem lokalem Speicher entsteht eine Detailauflösung, die für die Erfassung des Wesentlichen meist nicht erforderlich ist, aber einen sehr hohen Datenfluß vom *Compute-Server* zum *Grafik-Server* produziert.

Trotzdem zeigt sich auch bereits bei „kleineren“ Aufgaben, daß für eine adäquate Visualisierung der zu betrachtenden Simulationsaufgaben auf massiv parallelen Rechnern eine Grafikworkstation der unteren Mittelklasse, die im Bewilligungszeitraum angeschafft wurde, nicht ausreichend ist.

Zugleich muß über eine Modifikation der intern verwendeten Datenstrukturen nachgedacht werden, die es ermöglicht, von einem sehr feinen Berechnungs-Netz zu einem gröberen, für die Visualisierung ausreichenden Netz zurückzuschalten. Ansätze dieser Art werden auch im Zusammenhang mit der adaptiven Netzverfeinerung bzw. -vergröberung benötigt.

Die Problematik der Lastverteilung bei adaptiver Arbeitsweise auf Parallelrechnern ist eine nach wie vor kritische und für die künftige Forschung wesentliche Fragestellung, die insbesondere in den Teilprojekten A3 und D3 verfolgt wird, aber auch eine enge Beziehung zum Pre- und Postprocessing aufweist. Ein zentrales Problem ist dabei, die durch lösungsabhängige lokale Netzverfeinerung entstehenden Balance-Unterschiede auf dem Parallelrechner abzufangen. Mit den bisherigen Parallelisierungsstrategien erfordert der Versuch einer Lastbalancierung im wesentlichen eine Neuverteilung des Problems und damit aller Daten über den Parallelrechner. Entscheidungskriterien für eine lokale bzw. globale Um-/Neuverteilung der Last können unter Berücksichtigung geeigneter Kommunikationsmethoden ebenso lokal oder global bestimmt sein. Hierzu ist einerseits die Frage zu untersuchen, ob durch spezielle Kommunikationsstrategien der Aufwand für den Lastausgleich zwischen den Prozessoren durch dessen Nutzen wieder aufgewogen wird. Andererseits entsteht die Frage nach neuen, flexibleren Architekturen, die eine dynamische Zuordnung von Prozessoren des Parallelrechners unterstützen.