TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

"Anwendungsbezogene Lastverteilung" ALV'98

25.-26. März 1998

München

Arndt Bode, Andreas Ganz, Claudia Gold, Stefan Petri
Niels Reimer, Björn Schiemann, Thomas Schnekenburger
(Herausgeber)

TUM-I9806
SFB-Bericht Nr. 342/01/98 A
Februar 98

TUM–INFO–02-I9806-100/1.–FI

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

Workshop
"Anwendungsbezogene Lastverteilung"
ALV'98

25.-26.März 1998
München

Veranstaltet vom Sonderforschungsbereich 342
"Werkzeuge und Methoden für die Nutzung paralleler Rechnerarchitekturen"
Querschnittsthema "Anwendungsbezogene Lastverteilung"
und vom Graduiertenkolleg
"Kooperation und Ressourcenmanagement in verteilten Systemen"

Herausgeber, Organisationskomitee:

Arndt Bode, TU München, Sprecher SFB 342
Andreas Ganz, TU München
Claudia Gold, TU München
Stefan Petri, MU Lübeck
Niels Reimer, TU München
Björn Schiemann, Siemens AG
Thomas Schnekenburger, Siemens AG

Vorwort

Die anwendungsbezogene Lastverteilung ist das zentrale Thema bei der Optimierung von Anwendungsprogrammen auf parallelen und verteilten Systemen. Das Thema hat daher auch eine besondere Rolle im Sonderforschungsbereich 342 „Werkzeuge und Methoden für die Nutzung paralleler Rechnerarchitekturen“ und im Graduiertenkolleg „Kooperation und Ressourcenmanagement in verteilten Systemen“ der Technischen Universität München. Als Querschnittsthema im Sonderforschungsbereich 342 wurde die anwendungsbezogene Lastverteilung zunächst bezogen auf Multiprozessorsysteme mit verteiltem Speicher untersucht. In den letzten Jahren hat sich der Schwerpunkt der Arbeiten dann auf Netze von (Arbeitsplatz-)Rechnern verschoben, die ebenfalls durch verteilte Speicher gekennzeichnet sind, intern jedoch auch das Modell des gemeinsamen Speichers benutzen können. Fragen des Lastausgleichs und der Ressourcenverwaltung sind dabei ebenso eng verknüpft wie die des anwendungsintegrierten und des systemintegrierten (anwendungsübergreifenden) Lastausgleichs. Je nach Zielsystem, Betriebssystem, Programmiermodell und Anwendung sind hier unterschiedlichste Verfahren vorgeschlagen worden. Es ist das Ziel des Workshops, Forscher und Anwender, die im Bereich der dynamischen anwendungsbezogenen Lastverteilung tätig sind, zusammenzubringen und dabei insbesondere die Arbeiten im Sonderforschungsbereich 342 und dem Graduiertenkolleg „Kooperation und Ressourcenmanagement in verteilten Systemen“ mit externen Kooperationspartnern eng zu verknüpfen.

München, den 28.1.1998

Arndt Bode
Sprecher SFB 342

Inhaltsverzeichnis

Load Balancing for Problems with Good Bisectors, and Applications in Finite Element Simulations
Stefan Bischof, Ralf Ebner, Thomas Erlebach (Technische Universität München) **1**

An Algorithm for Load Distribution in Interactive Theorem Proving
Thomas Honigmann (Humboldt-Universität Berlin) Andreas Wolf (Technische Universität München) **13**

Skew-tolerantes, dynamisches LPT-Scheduling zur Join-Verarbeitung in parallelen Shared-Disk-Datenbanksystemen
Holger Märtens (Universität Leipzig) **23**

A Decentral, Adaptive Load Sharing Approach for Workstation Clusters using Fuzzy Logic
Thomas Böhme (Technische Universität Ilmenau) Herwig Unger (Universität Rostock) **33**

Load Balancing for Parallel Adaptive FEM
Ralf Diekmann, Frank Schlimbach (Universität Paderborn) Chris Walshaw (University of Greenwich, London) **41**

Load Balancing in High-Performance Distributed DB/DC-Systems: Modeling, Analysis and Clustering of Workload for Data-Affinity Based Load Distribution
Reinhard Riedl (Universität Zürich) **53**

Towards a Scalable Parallel Algorithm for Polynomial Real Root Isolation
Thomas Decker, Werner Krandick (Universität Paderborn) **65**

Dynamic Load Distribution with the WINNER System
Olaf Arndt, Bernd Freisleben, Thilo Kielmann, Frank Thilo (Universität Siegen) **77**

The PLB-Library: Dynamic Load Balancing on NOWs
Frank Meisgen, Ewald Speckenmeyer (Universität Köln) **89**

Object Mobility in ARTS
Lars Büttner (GMD-FIRST) **101**

Lastverwaltung in Workstation-Clustern auf Basis Mobiler Agenten
Claus Grewe, Wolfgang Obelöer, Holger Pals (Medizinische Universität zu Lübeck) **111**

Tree Shaped Computations as a Model for Parallel Applications
Peter Sanders (Max-Planck-Institut Saarbrücken) **123**

NSR - A Tool for Load Measurement in Heterogeneous Environments
Christian Röder, Thomas Ludwig, Arndt Bode (Technische Universität München) **133**

Load Balancing for Parallel Adaptive FEM *

Ralf Diekmann,[1] Frank Schlimbach,[1] and Chris Walshaw[2]

[1] Department of Computer Science, University of Paderborn, Germany.

[2] School of Computing and Mathematical Sciences,

The University of Greenwich, London, UK.

Extended Abstract

Abstract

We present a dynamic distributed load balancing algorithm for parallel, adaptive finite element simulations using preconditioned conjugate gradient solvers based on domain-decomposition. The load balancer is designed to maintain good partition aspect ratios. It calculates a balancing flow using different versions of diffusion and a variant of breadth first search. Elements to be migrated are chosen according to a cost function aiming at the optimization of subdomain shapes. We show how to use information from the second step to guide the first. Experimental results using Bramble's preconditioner and comparisons to existing state-ot-the-art load balancers show the benefits of the construction.

1 Introduction

Finite Elements (or Finite Differences or Finite Volumes) can be used to numerically approximate the solutions of partial differential equations (PDEs) describing, for example, the flow of air around a wing or the distribution of temperature on a plate which is partially heated [2, 30]. The domain on which the PDE is to be solved is discretized into a mesh of *finite elements* (triangles or rectangles in 2D, tetrahedra or hexahedra in 3D) and the PDE is transformed into a set of linear equations defined on these elements [30]. The coupling between equations is given by the neighborhood in the mesh. Usually, iterative methods like Conjugate Gradient (CG) or Multigrid (MG) are used to solve the linear equational systems [2].

The quality of solutions obtained by such numerical approximation algorithms depends heavily on the discretization accuracy. In particular, in regions with steep solution gradients, the mesh has to be refined sufficiently, i.e. the elements should be small

*This work is partly supported by the DFG-Sonderforschungsbereich 376 "Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen".

in order to allow a good approximation. Unfortunately the regions with large gradients are usually not known in advance. Thus, meshes either have to be refined regularly resulting in a large number of small elements in regions where they are not necessary, or the refinement takes place during the calculation based on error estimates of the current solution [28]. Clearly, the second variant should be favored because it generates fine discretizations only in those regions where they are really needed. The resulting solutions have the same quality as if the mesh was refined regularly but the number of elements, and thus the time needed to determine the solution, is only a small fraction of the regular case.

The parallelization of numerical simulation algorithms usually follows the single-program multiple-data (SPMD) paradigm: The same code is executed on each processor but on different parts of the data. This means that the mesh is partitioned into P subdomains where P is the number of processors, and each subdomain is assigned to one processor [12]. Iterative solution algorithms mainly perform local operations, i.e. new values at certain parts of the mesh only depend on the values of nearby data (modulo some global dependencies in certain iteration scemes which we neglect here as they are independent of the way the mesh is partitioned). Thus, the parallelization requires communication mainly at the partition boundaries.

The efficiency of such a SPMD parallelization mainly depends on the equal distribution of data on the processors. As the individual iteration steps are synchronized by a data exchange at partition boundaries the time the processors need to perform one iteration step should be the same on each processor. This can be achieved by splitting the mesh into equal sized subdomains. If each processor contains the same number of elements, the time each needs for one iteration will be approximately the same [12].

In the case of adaptive refinement, the distribution of data on the processors becomes unbalanced if the number of newly generated elements is not the same on each processor (which usually is the case with solution adaptive refinement). Therefore, the partition has to be altered in order to re-establish a balanced distribution.

As already mentioned, iterative solvers are easy to parallelize and they mainly require communication of data at partition boundaries. Thus, most existing FEM partitioners and load balancing algorithms aim at minimizing the boundary length which is a valid measure for the time needed to complete a communication step.

A powerful method to speed up linear solvers is to improve the condition of the equational system [2, 5, 30]. *Domain Decomposition* methods (DD) belong to the most efficient preconditioners. Unfortunately, their speed of convergence depends on the way the partition is generated. The shape (the Aspect Ratio) of subdomains determines the condition of the preconditioned system and has to be considered if partitions are changed due to load balancing.

PadFEM (**P**arallel **ad**aptive **FEM**) is an object-oriented environment supporting parallel adaptive numerical simulations [3, 24]. *PadFEM* includes graphical user interfaces, mesh generators (2D and 3D, sequential and parallel) [22], mesh partitioning algorithms [12], triangular and tetrahedral element formulations for Poisson and Navier-Stokes problems [30], different solvers (especially DD-PCGs) [2, 5], error es-

timators [28], mesh refinement algorithms [20], and load balancers [12].

The load balancing module in *PadFEM* determines the amount of load to move between different subdomains in order to balance the distribution globally. The neighborhood between subdomains defines the *cluster-* or *quotient-graph* [12]. On this graph the algorithm determines a *balancing flow* which tells the partitions how many data items (i.e. elements) they have to move to each of their neighbors. This balancing flow calculation can optionally use different kinds of diffusive methods [4, 9], in particular first and second order diffusion iterations [13, 15], and a heuristic for convex non-linear min-cost flow [1].

In a second step, the elements to be moved are identified. When choosing these elements, the load balancer tries to optimize partition Aspect Ratios in addition to load balance. The elements at partition boundaries are weighted by a cost function consisting of several parts. The migration chooses elements according to their weight and moves them to neighbors.

The element weight functions are used to "guide" the balancing flow calculation. We consider the existing partition and define weights for the edges of the cluster graph expressing a kind of "cost" for moving elements over the corresponding borders. The aim is to avoid a large flow demand between partitions whose common border is very small.

The next section gives an introduction into the field of balancing adaptive meshes. Section 3 describes the element migration strategies used within *PadFEM*. This second step in load balancing is presented first, as parts of it are used in Section 4 where the balancing flow calculation is described. Section 5 finishes the paper with a number of results about Aspect Ratios and iteration numbers of the DD-PCG solver within *PadFEM*. Comparisons are made to *Jostle*, the Greenwich parallel partitioning tool [29].

2 The Problem

In most cases, the computational load of a finite element problem can be measured in terms of numbers of elements. Thus, a load balanced parallel execution requires a partition of the mesh into subdomains containing the same amount of elements each. Such an initial partition can be generated using any of the existing very efficient graph partitioning algorithms (see e.g. [12] for an overview) or by generating the mesh in parallel [7].

If the mesh is refined adaptively, the existing partition will probably become unbalanced. Scalable load balancing algorithms can take benefit from an existing partition by moving elements on the *quotient graph* defined by the adjacency between subdomains. The quotient graph contains one node for each partition. Edges denote common borders between the corresponding partitions. Figure 1 shows an unbalanced partition of a simple mesh (domain "Square") into four subdomains (left) and the resulting quotient graph (center). Any reasonable load balancing taking the existing distribution into account has to move data (elements) via edges of this graph. We add weights to nodes

Need Level 2 to print this file. Exit.