Quality Balancing for Parallel Adaptive FEM *

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

[1] Department of Computer Science, University of Paderborn,
Fürstenallee 11, D-33102 Paderborn, Germany
{diek, schlimbo}@uni-paderborn.de
[2] School of Computing and Mathematical Sciences,
The University of Greenwich, London, UK.
C.Walshaw@gre.ac.uk

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 can calculate 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 [2,34]. 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 [34]. 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 systems of linear equations [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 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

* This work is partly supported by the DFG-Sonderforschungsbereich 376 "Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen" and the EC ESPRIT Long Term Research Project 20244 (ALCOM-IT).

in regions where they are not necessary, or the refinement takes place during the calculation based on error estimates of the current solution [30]. 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 [12]. In the case of adaptive refinement, the distribution of data on the processors becomes unbalanced and the partition has to be altered in order to re-establish a balanced distribution.

Preconditioning is a powerful method to speed up linear solvers [2, 5, 34]. *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), mesh partitioning algorithms [12], triangular and tetrahedral element formulations for Poisson and Navier-Stokes problems [34], different solvers (especially DD-PCGs) [2, 5], error estimators [30], mesh refinement algorithms [21], 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. It can calculate a *balancing flow* using different kinds of diffusive methods [4, 9], in particular first and second order diffusion iterations [13, 17], 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

Need Level 2 to print this file. Exit.