padfem2 – An Efficient, Comfortable Framework for Massively Parallel FEM-Applications*

Stephan Blazy, Odej Kao and Oliver Marquardt

Paderborn Center for Parallel Computing
Paderborn University, Fuerstenallee 11, 33102 Paderborn, Germany
{blazy,okao,marquardt}@uni-paderborn.de
http://www.upb.de/pc2/

Abstract. In this paper a new MPI-based framework *padfem2* for finite elements methods (FEM) is presented allowing an efficient execution of FEM applications on clusters with SMP nodes. This paper focuses on the global architecture and on an unique parallel data structure for high-performance object management on structured and unstructured grids. The efficiency and scalability of the parallel FEM processing is evaluated using Poisson and Navier-Stokes numerical solvers as an example.

1 Introduction

In this paper we present an MPI-based framework for massively parallel simulations of finite elements named *padfem2* which is based on the completely re-designed FEM-Tool PadFEM [1]. A re-design was necessary to integrate a highly modular architecture and to support modern software engineering methods. The main *padfem2* targets are the efficient implementation of FEM codes for massively parallel architectures (currently clusters and SMPs) combined with plug-in based environments for creation, specification, and development of modern numerical FEM algorithms, e.g. for computational fluid dynamics or crack simulation [2]. The abstraction of the complex architecture details, the support for the parallelization of novel algorithms and the transparent access to high-performance resources allow researchers – currently from chemistry, physics, mathematics, engineers, etc. – to focus on research-specific details.

This paper gives an overview about the global architecture of *padfem2* and demonstrates the performance characteristics by considering the Poisson and Navier-Stokes solvers as an example. The performance measurements are executed on a cluster with 32 SMP nodes. Due to the limited space details on the modules for numerical algorithms, adaptation, load balancing, mesh generation, pre- and post-processing, and visualization are omitted.

* This work is partly supported by the German Science Foundation (DFG) project SFB-376.

2 The *padfem2* Architecture

The *padfem2* architecture is based on a modular concept with two application levels – *padfem2* kernel and user level – which are depicted in Fig. 1. The kernel level provides core functionality to all FEM applications and consists of the following modules: a runtime environment, communication management in SMPs (threads) and tightly-coupled architectures (MPI, hive), a parallel and distributed data structure, and a module for basic and complex numerical algorithms/techniques. Additional modules support the management of massively parallel architectures dealing with grid partitioning or load-balancing, etc.

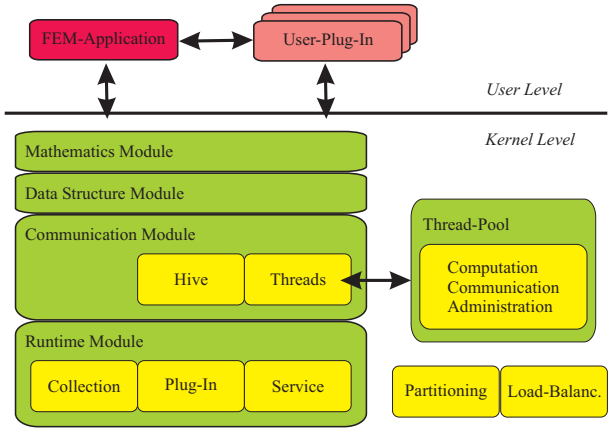

Fig. 1. Schematic overview of the *padfem2* architecture components

The *padfem2* user level allows the specification of FEM applications by using a combination of a-priori given and user-defined numerical algorithms/techniques. Development of the latter is supported by the data structure kernel module, which contains C++ classes for efficient memory management and an iterator concept for traversing objects such as vertices, volumes, etc. The user level modules are developed as plug-ins and loaded from the kernel during runtime. Thus, a flexible *padfem2* extension with an additional functionality such as a special grid management or the support for parallel tasks is provided.

A unique property and functionality of *padfem2* is provided by a sophisticated, machine-close data structure, which forms the building block for a highly efficient, parallel and distributed object management. The corresponding module determines and manages information about the partitioning of grids and its partition halo with different depths for parallel computation. Moreover, computations on structured and unstructured grids with basic (vertex, edge, face, volume) and complex (region, halo) objects are supported. This data structure also provides an automatically determinable but user configurable neighborhood relationship detection. The user can select if he/she needs a partial or full knowl-

edge of the neighborhood relation for each object type. Using this application-dependent knowledge the specification as well as the performance of numerical algorithms can be improved significantly. Furthermore, special requirements on physical or chemical data can be modeled by the user through a container concept, so called attachment. Attachments are defined in plug-ins, bundled with *padfem2* object types, and can be accessed in the numerical algorithms for further usage. The supporting functions for partitioning and load balancing are currently based on well-known methods such as PARTY [6]. The adaptation is executed by tetrahedron refinement based on problem-dependent error estimators.

Building a complete FEM application out of the framework is done with less effort. Firstly, one has to define the set of numerical data to compute on (e.g. temperature or velocity values). This data is enclosed in an attachment class and appended to each mesh object. The framework provides easy-to-use functions for that purpose. Secondly, own numerical algorithms may be added in a user-defined plug-in working on the attachments. These algorithms are supported by basic or complex help functions from the *padfem2* kernel. Finally, the user has to specify the FEM domain for the problem (i.e. mesh including boundary condition specifications). The user-defined plug-in and the FEM domain are the inputs for the *padfem2* main program.

3 Parallelism with MPI and Threads

The *padfem2* system combines two parallelism paradigms in one application framework and thus supports the current trends in parallel processing, where powerful SMP compute nodes are combined into a cluster with high-speed interconnects and message passing.

The SMP parallelism for the compute nodes is realized using the POSIX thread model, as this standard allows a manual optimization of the program code. A grid partition is mapped exclusively onto a single SMP compute node and the data structure takes care of the partition and global grid consistency. The provided C++ algorithm classes and functions hide the complex thread management from the user and ease the development of numerical algorithms significantly.

The message passing interface (MPI) is used to run *padfem2* on massively parallel systems. Analogously to the SMP thread model the MPI mechanism is also hidden in C++ classes. Hence, the underlying communication software can be replaced by alternative low-level communication packages (GM, GM-2, VMI, etc.) in order to exploit the characteristics of the application and HPC-architecture in the most suitable and efficient way. The inter node communication is optimized by minimizing the submitted information for communication steps, as the transmission of redundant data is avoided. The communication latency is hidden by thread-based communication architecture, where the computation and communication tasks are run in parallel. The provided numerical algorithms and management modules assist this concept in a consistent way.

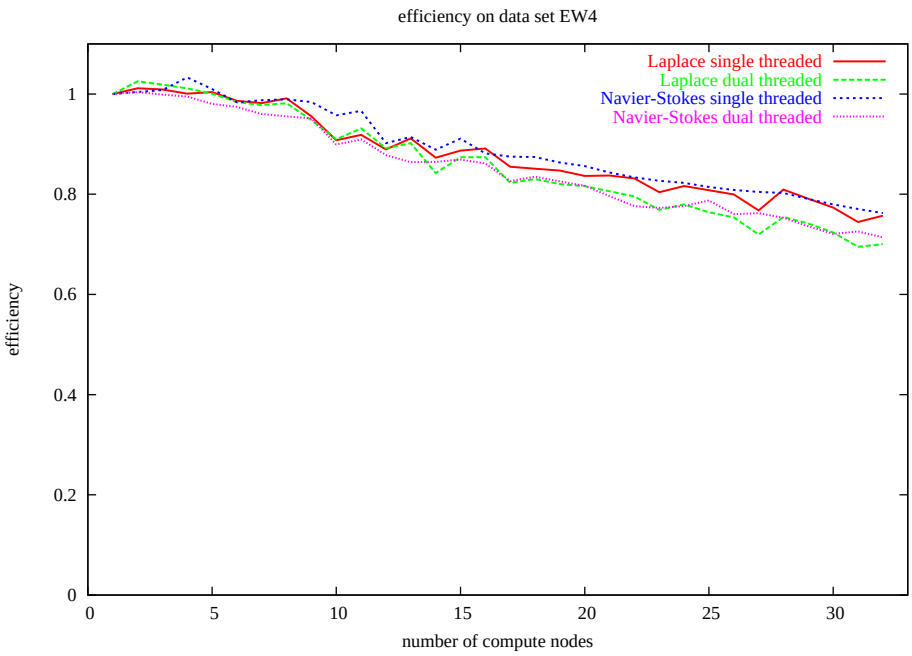

Fig. 2. Efficiency on FEM data set

A thread pool is allocated in the *padfem2* kernel level managing several tasks during runtime (see Fig. 1). The pool determines autonomously how many threads will be created and maps the tasks of computation, communication and miscellaneous services to them. This, together with the latency hiding approach, leads to a balanced load on each SMP compute node in a tightly-coupled environment with up to 256 nodes.

4 Numerical Applications

In *padfem2* we provide finite element methods using linear ansatz-functions on three dimensional tetrahedral grids. The generated linear systems of equations are solved by using a conjugate gradient method. *padfem2* also supports other iterative solvers (e.g. algebraic multi-grid methods). Furthermore, it is possible to implement own method as external modules.

We consider two numerical examples to measure the performance of the conjugate gradient method and the efficiency of the data structure. For the first, we solve the standard 3D Poisson problem in a cube with zero Dirichlet boundary conditions: $-\Delta u(x) = f(x)\ x \in \Omega, \quad u_{|\partial\Omega} = 0$. In the second test case we consider the 3D Navier-Stokes equation:

$$\left.\begin{aligned} \tfrac{\partial}{\partial t} v + (v \cdot \nabla) v - \nu \Delta v + \nabla p = 0 \\ div\ v = 0 \end{aligned}\right\} \text{ in } \Omega.$$

For solving this equation, we use a characteristic pressure correction scheme, which leads to a three step algorithm [3–5].

For the performance measurements of these examples we used a FEM data set consisting of a 3D cube with 48423 vertices forming 236762 tetrahedrons. During the simulation the FEM data set was handled as a static grid with no 3D adaptation. The simulation is executed on a Siemens hpc*L*ine cluster system with 32 SMP compute nodes, each of them with two Pentium3-850 MHz, 512 MByte RAM. The measurement results in Fig. 2 show that the efficiency of the parallel FEM applications ranges from 70% to 75% using all 32 compute nodes. Please note, that we worked with a static grid, thus the efficiency results will be improved significantly, as soon as dynamic grid adaptation is integrated. The efficiency values greater than 100% can be explained by operating system activities (e.g. swapping of concurrent processes) during the performance measurement on a single compute node.

5 Conclusion and Future Work

This paper presented the efficient and comfortable framework *padfem2* for specification and execution of FEM applications on massively parallel architectures. This work-in-progress is supported by a number of users in science and industry, thus the individual specification and computational needs of the users are considered during the design and implementation.

Future work is related to the development of further massively parallel algorithms for the three dimensional Navier Stokes equation on unstructured, adaptive grids using a self-organizing 3D grid adaptation technique with motion support. The technical effort is currently directed to optimization of *padfem2* for 64bit architectures in tightly-coupled clusters with modern high speed interconnects.

References

1. Diekmann, R., Dralle, U., Neugebauer, F.: PadFEM: A Portable Parallel FEM-Tool, Proc. Int. Conf. High-Performance Computing and Networking, Apr. 1996.
2. *padfem2* A Parallel adaptive Finite Element Method tool box, http://www.padfem.de.
3. S. Blazy, O. Marquardt: A characteristic algorithm for the 3D Navier-Stokes equation using *padfem2*, TR-RSFB-03-74, 2003.
4. Blazy, S., Borchers, W., Dralle, U.: Parallelization methods for a characteristic's pressure correction scheme. In: E.H. Hirschel (ed), Flow Simulation with High-Performance Computers II, Notes on Numerical Fluid Mechanics, 1995.
5. Gilles, F.: Une méthode des caractéristiques d'ordre deux sur maillages mobiles pour la résolution des équations de Navier-Stokes incompressible par élements finis, INRIA, Report RR 4448, 2002.
6. Preis,R., Diekmann, R.: PARTY - A Software Library for Graph Partitioning, Advances in Computational Mechanics with Parallel and Distributed Processing, Civil-Comp Press, 1997, pp. 63-71