Available on-line at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Improving the scalability and applicability of Elmer finite element software for industrial problems

Mikko Byckling[a], Mika Malinen[a], Juha Ruokolainen[a], Peter Råback[a,*]

[a]CSC – IT Center for Science, Keilaranta 14, 02101 Espoo, Finland

Abstract

Recent developments of Elmer finite element solver are presented. The applicability of the code to industrial problems has been improved by introducing features for handling rotational boundary conditions with mortar finite elements. The scalability of the code has been improved by making the code thread-safe and by multithreading some critical sections of the code. The developments are described and some scalability results are presented.

Project ID: 012345

1. Introduction

Elmer is one of the most popular multiphysical simulation software published under open source [1]. Elmer has been mainly developed by CSC - IT Center for Science (CSC). The development was started in 1995 in collaboration with Finnish Universities, research institutes and industry. In 2005 Elmer was published under open source and thereafter the use and development have become largely international.

Elmer consists of several components: ElmerGUI is the graphical user interface, ElmerPost is the post-processing tool, ElmerGrid the mesh manipulation and partitioning tool, and most importantly ElmerSolver the finite element solver. ElmerSolver includes generic finite element library functionalities that may be called by dynamically linked modules describing the particular physical phenomena. The modules include models for fluid dynamics, structural mechanics, electromagnetism, heat transfer and acoustics etc. The physical models may be weakly coupled without any a priori defined way. Due to the modular structure new equations may be added to Elmer without touching the main library. ElmerSolver library includes its own standard preconditioned iterative and multilevel methods but also an extensive list of available linear algebra libraries, such as Hypre and Trilinos.

The user base of Elmer may be counted in thousands. For example, during the last year Windows binary from sf.net was downloaded about 20,000 times. In industry Elmer has been used to model crystal growth [2], acoustics [3], micro-electro-mechanical systems [4], fluid-structure interaction, for example. In the early years of Elmer project the main motivation for industrial use was the complex physical couplings. However, in recent years the good parallel performance has become more important driver for the industrial applications. This is natural development since the commercial codes master many application fields more thoroughly but often do not demonstrate sufficient parallel scaling needed in large-scale computations.

Elmer has been since the beginning written as a parallel code and has demonstrated reasonable scalability for more than a decade. However, as the number of computational cores and the size of the problems grow, the old methods are not sufficient as new bottlenecks start to appear. Also, there may be new application areas including their specific features that must be implemented to enable the use of the code.

Within this paper we review the developments in two themes within PRACE project that aim to improve the applicability of the code for industrial applications. These are the implementation of the rotating boundary conditions based on mortar finite elements, and the hybridization of the code to take use of the multicore processors that may in the future provide the most cost-efficient way of obtaining good performance.

Within PRACE project one of the major contributions to Elmer has also been the development of FETI (finite element tear and interconnect) for the solution of elliptic systems. This work has continued and has some synergies also with this work. In this report the emphasis is on the completely new developments and the more incremental work on the FETI will be omitted.

*Corresponding author.
tel. +358503305225 fax. +35894572302 e-mail. peter.raback@csc.fi

2. Boundary conditions for rotating devices

2.1. Motivation for rotating boundary conditions

Many evolutionary partial differential equation (PDE) models can be posed on fixed domains. In such cases a spatial region Ω corresponding to the model does not depend on the time t. This usually simplifies the process of computational modelling by using finite elements, since a single finite element mesh created before starting the actual simulation may then be employed.

When a rotating device such as the rotor of an electric machine is modelled, handling physical interactions between the rotating part and its surroundings necessitates the use of more complicated approaches in order to take into account the change in the spatial configuration. In principle the spatial discretization could then be done newly when advancing in time to the next step of numerical time integration. However, this strategy is usually costly in practice. In addition, the associated work-flow is then complicated, since the computational solution procedure for the fully discretized equations has to be interfaced with software which is used for mesh generation.

If the rotating part can be associated with a solid of revolution and performs a prescribed rigid body motion, an alternate approach is to employ a moving finite element mesh which is obtained by mapping a mesh created initially for the rotating part in its reference configuration Ω_0^R at the time $t = 0$ to the current configuration Ω_t^R. If the rotating part may additionally be considered to be embedded in a surrounding body Ω^S which remains fixed during the motion of the device, remeshing may then be avoided completely if a fixed mesh for the spatial dicretization on Ω^S is employed.

The principal complication associated with the moving mesh approach is that the finite element meshes are then generally nonmatching across the model boundaries which represent the interaction surface Γ shared by the rotating and surrounding bodies. It follows that the usual continuity requirements to obtain a conforming finite element approximation over the entire domain cannot be satisfied. In Elmer finite element software, problems of this type have traditionally been handled by using domain decomposition in combination with a solution algorithm of the Dirichlet–Neumann type. That is, the coupled linear systems arising from the discretizations over the two domain Ω_t^R and Ω^S are not handled monolithically but they are solved iteratively by employing a segregated solution method. In this project, an alternate monolithic solution strategy based on the mortar finite element method (see, for example, [5]) has been implemented into the solver of Elmer.

2.2. Mortar finite element method

In the special case of having two independent meshes, a mortar finite element discretization based on the Lagrange multiplier formulation generally leads to handling a linear algebra problem

$$\begin{bmatrix} \mathbf{A}_{RR} & \mathbf{0} & \mathbf{B}_R^T \\ \mathbf{0} & \mathbf{A}_{SS} & \mathbf{B}_S^T \\ \mathbf{B}_R & \mathbf{B}_S & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U}_R \\ \mathbf{U}_S \\ \mathbf{Q} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_R \\ \mathbf{F}_S \\ \mathbf{0} \end{bmatrix}. \tag{1}$$

Here the discrete equations

$$\mathbf{B}_R\mathbf{U}_R + \mathbf{B}_S\mathbf{U}_S = \mathbf{0}$$

arise from the weak formulation of the continuity constraint on the common interface Γ of the two bodies where the approximation of the primary unknown is associated with either the vector $\mathbf{U}_R$ or $\mathbf{U}_S$. In addition, the vector $\mathbf{Q}$ is associated with the approximation of the Lagrange multiplier, which typically provides an approximation to a flux or a circulation variable on Γ.

Standard iterative linear solvers are usually inefficient when they are applied to saddle point systems such as (1). In Elmer the systems arising from the mortar discretization may be preconditioned in connection with the iterative solution by utilizing the inexact version of the Uzawa algorithm in order to obtain better convergence.

2.3. Applying the continuity constraint in parallel

To construct the continuity constraint, we typically need to go through Gaussian integration points associated with the non-mortar side (that is, the computational mesh on the common interface Γ where the Lagrange multipliers are also defined) and express them in terms of local coordinates associated with an element on the mortar side. This process involves a geometric search process that is at least $O(N \log N)$ in complexity. Creating the continuity constraint will be greatly complicated if the elements on the mortar side containing the master degrees of freedom are not in the same partition. This would also mean that the application of the continuity constraint would require heavy communication.

The complete parallelization of the mortar projector turned out rather cumbersome and was not implemented within the current project. Unfortunately having all the mortar elements in one partition only would severely limit the maximum scalability. In 2D the fraction of mortar elements (or boundary elements in general) is proportional to $N^{-1/2}$ and in 3D to $N^{-1/3}$, respectively. Therefore at some stage a further increase in the number of partitions would not give any improvement as the load balancing becomes suboptimal.

In order to improve the chances for good scalability a special hybrid partitioning scheme was implemented. It the scheme the elements associated with the mortar projector are treated separately and the other elements

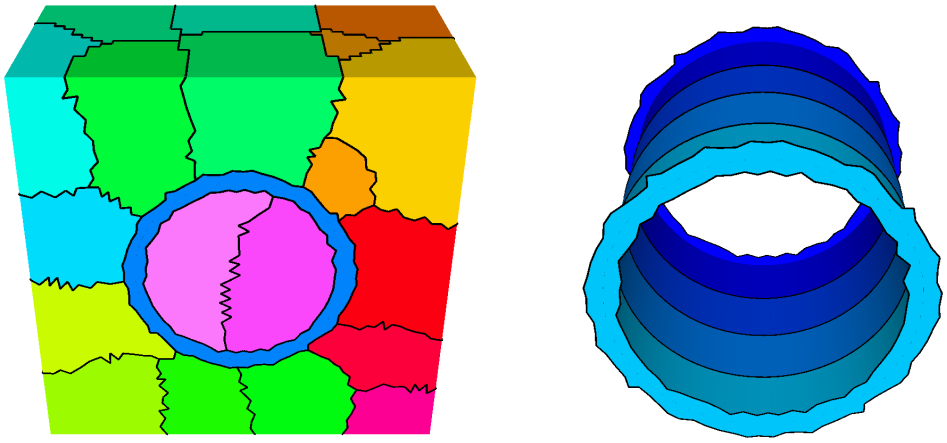

Fig. 1. An example of partitioning utilizing a hybrid partitioning scheme. On the left the whole partitioning into 32 partitions and on the right the rotational part divided into 4 partitions.

Table 1. Scalability of a realistic 2D induction machine case

procs	Time(s)	Speedup
1	225.1	-
3	93.6	2.4
4	63.5	3.5
8	34.1	6.6
16	29.4	7.7

are thereafter partitioned using standard partitioning algorithms, such as Metis. In 2D rotational problems the mortar elements must all lie within the same partition to ensure that the no parallel communication is needed for the projector. However, in 3D rotational problems that have been created by extrusion the elements may be grouped into sets along the rotational axis that have only interaction within themselves. This enables simple partitioning along the direction of the rotational axis and thereby better load balancing, and consequently improved parallel performance. An example of the resulting partitioning may be seen in figure 1.

2.4. Parallel performance results for the rotating boundaries

The mortar projector was tested in the case of a realistic 2D induction machine. This is a relatively small case where the total time consumption comes from the need to resolve the time-discretized equations for thousands of times. The number of linear triangular elements was 24 481. Of these elements 1 788 were located at the mortar boundary which is roughly 1/13.6. In this case all the mortar boundaries had to be located in one partition. Considering that the computational cost associated to the mortar elements is much higher than elsewhere the scalability is severely limited. The case is also so small that even without the mortar projector the scaling would be limited to rather small number of partitions. The results presented in table 1 still show that the scalability is quite good up to 8 partitions but going further to 16 partitions has only little benefit.

Now the true parallel benchmark should of course be 3D. Unfortunately, we were not able to test this in a realistic case. The computation of 3D electrical engines is based on the utilization of Whitney elements (edge elements) and for these elements the mortar projector still needs some further improvement. Instead we tested the 3D mortar projector with an synthetic extruded case using a transient heat equation. The computational mesh consisted of 459 269 nodes and 879 240 wedge elements. The mesh was obtained by extruding a triangular mesh for 60 layers. Of the elements 29 880 were associated with the mortar projector which is about 1/34.

First we fixed the total number of partitions to 64 and tested the effect of the number of mortar partitions. In table 2 it may be seen that there is an optimal number of partitions for the mortar projector. In this case choosing 8 cores for the mortar projector resulted to smallest computation time. For that value each mortar partition included 3 375 element in average and non-mortar partitions 15 167 respectively (ratio 4.5).

After establishing a favourable ratio between non-mortar and mortar partitions to 8 we repeated the computations with different number of cores, see table 3. The scaling of this case is reasonable considering the small size of the case. As long as the mortar projector can be further divided into independent sectors we don't expect it to have significant effect on the scalability. Larger 3D tests on realistic geometries are needed to confirm this.

Table 2. Computation of simple 3D extruded geometry with 64 partitions varying the number of mortar partitions

mortar procs	Time(s)
1	9.5
2	5.5
4	4.3
8	4.0
16	4.6
32	7.2

Table 3. Computation of simple 3D extruded geometry with fixed ratio of mortar to non-mortar partitions

procs	Time(s)	Speedup
8	25.4	-
16	13.3	1.9
32	9.1	2.8
64	4.0	6.4
128	2.4	10.6

2.5. Mortar projectors in cases with symmetry

Sometimes the periodic objects have additional symmetries. For example, it could be that it suffices to model only 1/6 of the object. This introduces an additional complexity. To overcome this the nodes belonging to the rotating body are mapped to an angle that is covered by the steady master body. By considering the symmetry the size of the problem, and also the size of the mortar projector, is greatly reduced. Figure 2 shows the periodic mortar projectors in action.

Considering the symmetry of the problem enables the solution of large problems. This is particularly important in 3D where geometrically accurate models may become extremely large. The added effect of the symmetry is that for problems of similar size the fraction of mortar elements decreases and therefore it is possible to obtain better scalability.

3. Hybridization of the Elmer code

3.1. Thread safety

ElmerSolver has not been originally implemented to use multithreading and being a legacy code using FORTRAN 90 and C/C++, static and module variables are commonly used for storing the data. When enabling the parallelization of the finite element assembly loops, OpenMP `THREADPRIVATE` directives were inserted to make some of the global variables reserved in DefUtils, Integration, InterpolateMeshToMesh and Lists modules private to each thread. To enable all the threads to update the global matrices simultaneously, `ATOMIC` directives were added to the routines handling the local matrix gluing in CRSMatrix and SolverUtils modules.

ElmerSolver uses a library called matc both internally and for user defined subroutine input. These functions may be then used to compute boundary conditions or body forces within a parallelized finite element assembly loop. Therefore we went through the whole code of matc-library and converted it to be thread safe. Within the ElmerSolver, initialization calls to matc were modified in order to have all the threads have their private copy

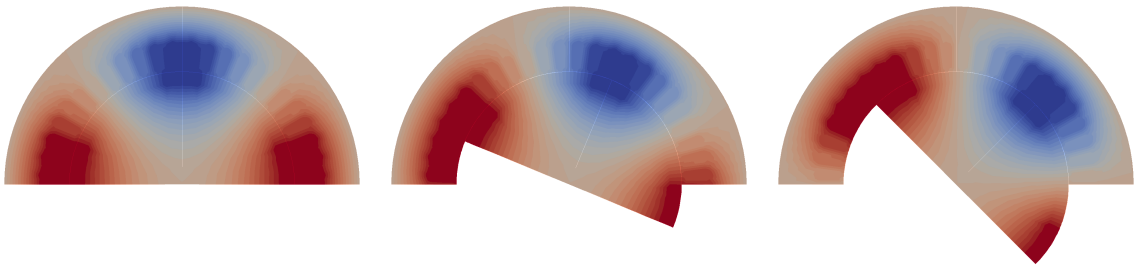

Fig. 2. An electrical induction machine with two-fold symmetry. Continuity of the solution is ensured by three mortar projectors. The first one is applied to the rotating boundary, the second to obtain the rotational symmetry of the stator, and the third to create the rotational symmetry of the rotor. The figure shows the vector potential in three different positions.

Table 4. Scalability of FE assembly for a steady state 3D heat equation

threads	CPU time	Speedup
1	5.5	1
2	2.87	1.9
4	1.56	3.5
8	1.24	4.4
16	0.84	6.5

Table 5. Scalability of CG solve for a steady state 3D heat equation

threads	CPU time	Speedup
1	3.40	1
2	2.00	1.7
4	1.52	2.2
8	1.41	2.4
16	0.84	4.04

of the variables within the library.

3.2. Multithreading improvements

For the linear solvers, an important part of the computations are sparse matrix-vector multiplications, commonly done within Krylov subspace methods. This operation was parallelized by adding OpenMP `PARALLEL DO` directives. In addition, we added an interface to highly optimized sparse matrix-vector subroutines from Intel MKL library.

3.3. Multithreaded performance

We investigate the multithreaded performance of ElmerSolver by solving a simple model problem, i.e., the heat equation in a steady state. To discretize the problem, we use 64000 3D hexahedrons with a finite element basis function degree $p = 2$. As a solver for the linear system, we use Conjugate Gradient iteration without preconditioning.

The results for finite element assembly and the solution of the linear system are presented in Tables 4 and 5. The timings are in seconds and they were performed on a dual socket machine with two Intel Xeon E5-2670 processors. The performance improvements are quite modest in both cases, especially the conjugate gradient solve exhibits quite poor scaling.

In order to improve the multithreaded performance of the solver in the future, we studied how the performance of the routines used to solve sparse linear systems could be improved. Our main purpose was to try to understand, which factors caused the solvers to have quite modest speedups in a multithreaded environment.

To test the performance of the solvers, we chose two test problems, a steady state heat equation (heat) and a linear elasticity problem (elas), both of which are relatively easy to solve with the conjugate gradient method even without any preconditioning. As a test domain for the problems, we used two intersecting square pipes in 3D. We use three different mesh sizes for the problems, small, medium and large. The test problems with their sizes, number of nonzeroes and relative densities are described in Table 6.

Since a CG solver without preconditioning is readily parallelizable, the poor performance of the solvers must be explained by threading overhead and NUMA effects.

We first considered the key computational kernel of the CG method, i.e., sparse matrix-vector product $y = Ax$, where A is an $n \times n$ matrix which is large and sparse. In Elmer, the matrix A is commonly stored in compressed row storage (CRS) format, which is known to have flops to byte ratio of approximately $O(\frac{1}{6})$. This means that the kernel is almost completely memory bound and prone to suffer from any NUMA effects.

To make this kernel NUMA aware, we redistribute the original matrix rowwise among threads. We then have the operation $y = Ax$ as

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{bmatrix}, \quad y = Ax \Leftrightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} A_1 x \\ A_2 x \\ \vdots \\ A_k x \end{bmatrix},$$

where k denotes the maximum number of threads, A_j denotes a thread local submatrix held by thread j and y_j the local part of the result vector. To avoid false sharing, we distributed the rows of matrix A in blocks of size 64 bytes.

Table 6. Test problems and their properties for multithreading tests

Test problem	n	nz(A)	$\rho = \text{nz}(A)/n$
3D_heat_small	35721	896761	25.1
3D_heat_med	105300	2702656	25.6
3D_heat_large	291060	7580368	26.0
3D_elas_small	24843	1786545	71.9
3D_elas_med	107163	8070849	75.3
3D_elas_large	315900	24323904	77.0

To avoid threading overhead, we implemented the whole CG method as a single parallel loop and let each thread to operate only on predefined sections of the work vectors. Again, to avoid false sharing, all work vectors were distributed in blocks of size 64 bytes. Local variables and OpenMP barriers were used to implement reduction operations and to do thread synchronization. With these modifications, we have the (preconditioned) CG method as described in Algorithm 1.

Algorithm 1 NUMA aware preconditioned conjugate gradient algorithm

for all threads in parallel **do**
  reduce bnrm2 $\leftarrow_+ ||b_{l:t}||_2^2$
  $r = b - Ax$
  barrier
  reduce $||r||_2^2 \leftarrow_+ ||r_{l:t}||_2^2$
  $\text{resnorm}_L = \sqrt{||r||_2^2/\text{bnrm2}}$
  if $\text{resnorm}_L < tol$ **stop**
  for $t = 1, \ldots, \text{maxit}$ **do**
    Solve $Mz = r$
    barrier
    reduce $\rho \leftarrow_+ r_{l:t}^T z_{l:t}$
    if $t > 1$ **then**
      $\beta_L = \rho/\rho_L$
      $p_{l:t} = z_{l:t} + \beta_L p_{l:t}$
    else
      $p_{l:t} = z_{l:t}$
    end if
    barrier
    $q = Ap$
    barrier
    reduce pdotq $\leftarrow_+ p_{l:t}^T q_{l:t}$
    $\alpha_L = \rho/\text{pdotq}$
    $x_{l:t} = x_{l:t} + \alpha_L p_{l:t}$
    $r_{l:t} = r_{l:t} - \alpha_L q_{l:t}$
    reduce $||r||_2^2 \leftarrow_+ ||r_{l:t}||_2^2$
    $\text{resnorm}_L = \sqrt{||r||_2^2/\text{bnrm2}}$
    if $\text{resnorm}_L < tol$ **stop**
    $\rho_L = \rho$
  end for
end for

We then tested Algorithm 1 against a textbook implementation of CG, taken from the HutIter-library of Elmer. As a method to implement the local and global matrix-vector products, we used a high-performance implementation CRS matrix-vector product from Intel MKL. The timings on test system with two Xeon E5-2670 processors are given in Table 7. As seen from the results, the NUMA aware conjugate gradient iteration performs signicifantly better than the original one. Especially when the matrix is small enough to fit into the local caches, the performance and scaling achieved is impressive.

4. Summary

In this paper the developments to enable the parallel use of Elmer for industrial application was reviewed. The rotational boundary conditions have been implemented as a generic feature and will allow simulation of new class of problems, such as electrical engines. The applicability of the chosen strategies has already been demonstrated. The mortar projector has not been fully parallelized. However, for extruded geometries a new partitioning scheme was implemented which gives reasonable parallel performance nevertheless. Further work

Table 7. Timings for original and NUMA aware CG methods

	Time (s)			Speedup, T_{nt}/T_1	
Test problem	nt=1	nt=16	nt=32	nt=16	nt=32
Original					
3D_heat_small	0.17	0.03	0.06	5.27	2.95
3D_heat_med	0.85	0.20	0.24	4.17	3.50
3D_heat_large	3.36	0.87	0.94	3.84	3.55
3D_elas_small	1.13	0.25	0.33	4.50	3.43
3D_elas_med	8.59	2.51	2.67	3.43	3.22
3D_elas_large	36.34	10.39	10.72	3.50	3.39
NUMA CRS					
3D_heat_small	0.17	0.02	0.01	7.75	11.59
3D_heat_med	0.87	0.20	0.05	4.43	16.48
3D_heat_large	3.45	0.86	0.46	4.00	7.48
3D_elas_small	1.15	0.20	0.07	5.81	16.44
3D_elas_med	8.83	2.57	1.28	3.44	6.90
3D_elas_large	36.49	10.76	5.69	3.39	6.41

would be needed to make the mortar projector fully parallel. The current implementation sets an upper limit to the number of partitions in 2D and in 3D it assumes an extruded nature for the mortar boundary.

In hybridization of the code an important milestone was to make the whole Elmer code thread-safe. This is a first requirement if multiple threads are to be run. The hybridization was carried out with Open MP pragmas to some critical pieces of the code, such as matrix-vector multiplication. As there are many places of the code where hybridization has not been carried out the overall performance improvements remain modest and are best suited for current standard multicore processors. Still choosing pure MPI parallelization over the Open MP one gives better scalability. The reason for the observations lie in the fact that many of the finite element routines are not optimally suited for the accelerator architectures.

To take full use of future accelerators, such as Inter MICs, more thorough changes in the code are required. For this reason we wrote a conjugate gradient solver from the scratch and optimized it for the multi-core architectures. The results showed that rather impressive parallel performance may be achieved also for sparse problems. Such drastic rewriting of the code was not possible within this work. However, as Elmer is now threadsafe all the developments in the external linear algebra libraries may be directly taken into use.

The test cases that were demonstrated in this case do not show the full potential of the developments for parallel computation. When the rotating boundary conditions are used for more complex problems with more degrees of freedom the number of partitions may be increased. The developments in multithreading, on the other hand, are intended to be used in conjunction with MPI parallelization. Unfortunately in this regard the developments were not finalized.

All-in-all the reported work improves the parallel performance and industrial applicability of the code. Given that Elmer is used by a large number of researchers the developments help to provide a smooth transitions into the world of parallel computing.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-283493.

References

1. `http://www.csc.fi/elmer`, Elmer finite element software homepage, 2013.
2. V. Savolainen et al., Simulation of large-scale silicon melt flow in magnetic Czochralski growth, J. Crystal Growth 243 (2002), 243-260.
3. A. Kärkkäinen et al., FEM Simulations of Thermal and Viscous Boundary Effects in Acoustics, In proceeding of: ICA (Internartional Conference on Acoustics) Kyoto 2004.
4. A. Pursula et al., Coupled FEM simulations of accelerometers including nonlinear gas damping with comparison to measurements, J. Micromech. Microeng. 16 (2006), 2345-2354.
5. F. Ben Belgacem, The mortar finite element method with Lagrange multipliers, Numer. Math., 84 (1999) 173-197.