Balancing the Communications and Computations in Parallel FEM Simulations on Unstructured Grids

Nikola Kosturski, Svetozar Margenov, and Yavor Vutov

Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences

Abstract. We consider large scale finite element modeling on 3D unstructured grids. Large scale problems imply the use of parallel hardware and software. In general, the computational process on unstructured grids includes: mesh generation, mesh partitioning, optional mesh refinement, discretization, and the solution. The impact of the domain partitioning strategy on the performance of the discretization and solution stages is studied.

Our investigations are focused on the Blue Gene/P massively parallel computer. The mapping of the communications to the underlying 3D tours interconnect topology is considered as well.

As a sample problem, we consider the simulation of the thermal and electrical processes, involved in the radio-frequency (RF) ablation procedure. RF ablation is a low invasive technique for the treatment of hepatic tumors, utilizing AC current to destroy the tumor cells by heating.

1 Introduction

Finite element method (FEM) on unstructured grids has proven to be an indispensable tool in computer modelling. Large scale simulations and complex models require parallel computing. This work is focused on optimizing the performance of parallel simulations.

We use state of the art parallel computer – IBM Blue Gene/P with a state of the art linear solver – BoomerAMG multigrid method[4] from the Hypre library[13]. Our intention was to study the influence of the mesh partitioning on the performance of the entire computational process. We compare two different partitioning libraries: ParMETIS and PT-Scotch. We also, to improve performance, try to map the communication pattern of our programs to the underlying 3D torus interconnect.

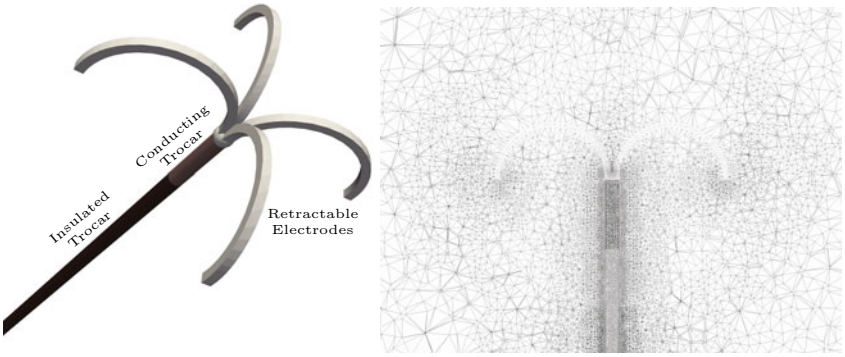
Our investigations are done while performing parallel simulation of the radio-frequency (RF) ablation. This is a hepatic tumor treatment technique which uses AC current from an electrode with a complex shape (see Figure 1. a).

The paper is organized as follows: In Section 2, we describe the problem we solve. In Section 3 we discuss and compare the used partitioners. Section 4 contains times from the parallel experiments with and without the mapping of the communications and we finish with a conclusion.

R. Wyrzykowski et al. (Eds.): PPAM 2011, Part II, LNCS 7204, pp. 211–220, 2012.

2 Radio-Frequency Tumor Ablation

Our test problem is numerical simulation of radio-frequency (RF) tumor ablation. RF ablation is an alternative, low invasive technique for the treatment of hepatic tumors, utilizing AC current to destroy the tumor cells by heating ([7,8]). The destruction of the cells occurs at temperatures of 45°C–50°C. The procedure is relatively safe, as it does not require open surgery.



a) b)

Fig. 1. a) The structure of a fully deployed RF probe; b) Sample mesh: cross-section, different gray levels are used for different materials

The considered RF probe is illustrated on Fig. 1. a) It consists of a stainless steel trocar with four nickel-titanium retractable electrodes. Polyurethane is used to insulate the trocar. The RF ablation procedure starts by placing the straight RF probe inside the tumor. The surgeon performs this under computer tomography (CT) or ultrasound guidance. Once the probe is in place, the electrodes are deployed and RF current is initiated. Both the surfaces areas of the uninsulated part of the trocar and the electrodes conduct RF current.

The human liver has a very complex structure, composed of materials with unique thermal and electrical properties. There are three types of blood vessels with different sizes and flow velocities. Here, we consider a simplified test problem, where the liver consists of homogeneous hepatic tissue and blood vessels.

The RF ablation procedure destroys the unwanted tissue by heating. The heat is dissipated by the electric current flowing through a conductor. The bio-heat time-dependent partial differential equation [7,8]

$$\rho c \frac{\partial T}{\partial t} = \nabla \cdot k \nabla T + J \cdot E - h_{bl}(T - T_{bl}) \tag{1}$$

is used to model the heating process during the RF ablation. The term $J \cdot E$ in (1) represents the thermal energy arising from the current flow and the term $h_{bl}(T - T_{bl})$ accounts for the heat loss due to blood perfusion.

The following initial and boundary conditions are applied

$$\begin{aligned} T &= 37°\text{C when } t = 0 \text{ at } \Omega, \\ T &= 37°\text{C when } t \geq 0 \text{ at } \partial\Omega. \end{aligned} \tag{2}$$

The following notations are used in (1) and (2):

- Ω – the entire domain of the model;
- $\partial\Omega$ – the boundary of the domain;
- ρ – density (kg/m^3);
- c – specific heat (J/kg K);
- k – thermal conductivity (W/m K);
- J – current density (A/m);
- E – electric field intensity (V/m);
- T_{bl} – blood temperature (37°C);
- w_{bl} – blood perfusion (1/s);
- $h_{bl} = \rho_{bl} c_{bl} w_{bl}$ – convective heat transfer coefficient accounting for the blood perfusion in the model.

The bio-heat problem is solved in two steps. The first step is finding the potential distribution V of the current flow. With the considered RF probe design, the current is flowing from the conducting electrodes to a dispersive electrode on the patient's body. The electrical flow is modeled by the Laplace equation

$$\nabla \cdot \sigma \nabla V = 0, \tag{3}$$

with boundary conditions

$$\begin{aligned} V &= 0 \text{ at } \partial\Omega, \\ V &= V_0 \text{ at } \partial\Omega_{el}. \end{aligned}$$

The following notations are used in the above equations:

- V – potential distribution in Ω;
- σ – electric conductivity (S/m);
- V_0 – applied RF voltage;
- $\partial\Omega_{el}$ – surface of the conducting part of the RF probe.

After determining the potential distribution, the electric field intensity can be computed from

$$E = -\nabla V,$$

and the current density from

$$J = \sigma E.$$

The second step is to solve the heat transfer equation (1) using the heat source $J \cdot E$ obtained in the first step.

For the numerical solution of both of the above discussed steps of the simulation the Finite Element Method (FEM) in space is used [10]. Linear conforming elements are used. To solve the bio-heat equation, after the space discretization,

backward Euler scheme is used [11]. There, the time derivative is discretized via finite differences. For a description of the discretization of the problem (3) see [12].

Let us focus on the discrete formulation of the bio-heat equation. Let us denote with K and M the stiffness and mass matrices from the finite element discretization of (1). They can be written as

$$K = \left[\int_\Omega k \nabla \Phi_i \cdot \nabla \Phi_j d\mathbf{x} \right]_{i,j=1}^{N},$$

$$M = \left[\int_\Omega \rho c \Phi_i \Phi_j d\mathbf{x} \right]_{i,j=1}^{N}.$$

Let us also denote with Ω_{bl} the subdomain of Ω occupied by blood vessels and with M_{bl} the matrix

$$M_{bl} = \left[\int_\Omega \delta_{bl} h_{bl} \Phi_i \Phi_j d\mathbf{x} \right]_{i,j=1}^{N},$$

where

$$\delta_{bl}(x) = \begin{cases} 1 \text{ for } x \in \Omega_{bl}, \\ 0 \text{ for } x \in \Omega \setminus \Omega_{bl}. \end{cases}$$

Than, the parabolic equation (1) can be written in matrix form as:

$$M \frac{\partial T}{\partial t} + (K + M_{bl})T = F + M_{bl} T_{bl}. \tag{4}$$

If we denote with τ the time-step, with T^{n+1} the solution at the current time level, and with T^n the solution at the previous time level and approximate the time derivative in (4) we obtain the following system of linear algebraic equations for the nodal values of T^{n+1}

$$(M + \tau(K + M_{bl}))T^{n+1} = MT^n + \tau(F + M_{bl}T_{bl}). \tag{5}$$

In table 1 are given the material properties, taken from [7]. The blood perfusion coefficient $w_{bl} = 6.4 \times 10^{-3}$ 1/s. For the test simulations, a RF voltage of 15 V is applied for a duration of 8 minutes. A time step of $\tau = 10$ s is used.

Table 1. Thermal and Electrical Properties of the Materials

Material	ρ (kg/m^3)	c (J/kg K)	k (W/m K)	σ (S/m)
Ni-Ti	6 450	840	18	1×10^8
Stainless steel	21 500	132	71	4×10^8
Liver	1 060	3 600	0.512	0.333
Blood	1 000	4 180	0.543	0.667
Polyurethane	70	1 045	0.026	10^{-5}

3 Partitioning Methods

Our software tools are written in C++, using MPI for the parallelization. Although we use external libraries for graph partitioning, and for the solution, plenty of nontrivial gluing code was required.

The first stage in the computational process is the mesh generation. Here Netgen package is used [14]. Currently, we use computer generated model of the liver and blood vessels. The RF ablation probe is inserted in the model. Then the geometric data is fed to the generator. In Fig. 1. b) is depicted a cross-section from a sample mesh. The generator output consist in three parts: list of coordinates of the vertices, list of tetrahedrons with assigned materials, and a list of boundaries with assigned boundary conditions.

The next stage is to partition the computational domain among processors. Our intent was to investigate the applicability of two graph partitioning libraries: ParMETIS [2,3] and PT-Scotch [5,6]. To use the graph partitioning routines in both cases we first calculate the dual graph of the mesh using the routine `ParMETIS_V3_Mesh2Dual`. Both libraries require for performance and scalability reasons the initial data to be distributed (fairly) among the processors. This is done by our toolchain as the mesh is read.

The routine `ParMETIS_V3_PartMeshKway` is used for the graph partitioning with ParMETIS. This call computes graph partitioning minimizing the number of cut edges. The result is a part (processor) number, assigned to each tetrahedron. ParMETIS uses parallel version of multilevel k-way partitioning algorithm described in [1].

PT-Scotch library computes graph partitioning via recursive bipartitioning algorithm. PT-Scotch contains ParMETIS compatibility layer, so we use the very same `ParMETIS_V3_PartMeshKway` function from it.

After the partitioning of the elements some postprocessing is required. This includes: distribution of the elements to their processors, determining the partitioning of vertices which are on the interfaces between the processors, distribution of vertex data, distribution of boundary condition data, and node renumbering, which is required for the parallel solver. We assign a shared vertex to a processor with lower number of previously assigned vertices.

Before giving experimental results, let us describe the Blue Gene/P parallel computer. It consist of racks with 1024 PowerPC 450 based compute nodes each. Each node has four computing cores and 2 GB RAM. The nodes are interconnected with several networks. The important ones from the computational point of view are: Tree network for global broadcast, gather, and reduce operations with a bandwidth of 1.7GB/s, 3D torus network for point to point communications, and a separate global interrupt network, used for barriers. In the torus network each node is connected to six other nodes with bidirectional links, each with bandwidth of 0.85GB/s. Torus network is available only when using multiplies of 512 nodes. For smaller number of nodes only 3D mesh interconnect is possible. We were using the machine in so call virtual node mode(VN), in which different MPI rank is assign to each of the computing cores. This is the mode

Table 2. Mesh partitioning

		Elements			Vertices		
mesh	N_{proc}	avg	$\max^{ParMetis}$	$\max^{PT-Scotch}$	avg	$\max^{ParMetis}$	$\max^{PT-Scotch}$
M0	128	110 453	114 606	110 885	18 561	21 672	20 937
M0	256	55 226	57 494	55 507	9 280	11 363	10 812
M0	512	27 613	28 736	27 765	4 640	5 857	5618
M0	1 024	13 806	14 364	13 902	2 320	3 073	2 968
M1	128	883 627	922 323	888 266	147 654	157 272	151 698
M1	256	441 813	460 387	444 112	73 827	80 836	77 136
M1	512	220 906	230 192	222 074	36 913	41 944	39 366
M1	1 024	110 453	115 055	111 102	18 456	21 796	21 118
M2	1 024	883 627	920 329	888 327	147 405	158 601	154 454
M2	2 048	441 813	461 092	444 934	73 702	81 441	78 815
M2	4 096	220 906	230 500	222 279	36 851	42 419	40 492

to use the entire power of the system, for pure MPI programs (without shared memory parallelism). We were allowed to use up to 1024 computing nodes – 4096 cores, further called processors.

The computation volume for the discretization is proportional to the number of elements in each processor. The computation volume for the solver is proportional to the number of vertices in each processor. Because of the global synchronizations which present in both processes, parallel times will be governed by the maximum number of elements and vertices per processor. In the table 2 these numbers are shown, compared to the averages. Three meshes had been partitioned: M0, M1, and M2. Meshes M1 and M2 are obtained from uniform refinement of mesh M0 once and twice. Let us note that the finest mesh M2 has about 9.0×10^8 elements and about 1.5×10^8 vertices. We expect from a good partitioner per processor maximums to be as close to the averages as possible. Results from partitions on different number of processors are shown.

We can clearly see that PT-Scotch produces better partitions. Both the maximum number of elements per processor and the maximum number of vertices per processor are closer to the optimal values.

To give an idea about the communication pattern resulting from the partitionings, we have shown some info about the connections in table 3. We call two processors connected if they share at least one vertex in their elements. Connected processors must exchange data during the discretization and the solution processes. The minimum – C_{min}, average – C_{avg}, and maximum – C_{min} number of connection for an processor are shown. The other column N_{max} is defined as follows:

$$\begin{aligned} N_{max} &= \max_{0<i<N_{proc}} \sum_{j=1}^{N_{proc}} N_{i,j}, \\ N_{i,j} &= \max(S_{i,j}, R_{i,j}), \end{aligned} \tag{6}$$

where N_{proc} is the number of processors, $S_{i,j}$ is the number of values sent from processor i to processor j and $R_{i,j}$ is the number of received values. In other

Table 3. Communication volume

		ParMETIS					PT-Scotch				
mesh	N_{proc}	C_{min}	C_{avg}	C_{max}	N_{max}	t[s]	C_{min}	C_{avg}	C_{max}	N_{max}	t[s]
M0	128	4	12	28	73 108	2.96	5	12	22	63 728	39.0
M0	256	4	13	26	47 392	1.59	3	13	25	39 020	35.9
M0	512	5	14	26	30 596	1.03	4	14	27	25 714	33.9
M0	1 024	5	14	27	19 376	0.87	4	14	26	16 652	32.8
M1	128	5	12	25	225 788	23.0	5	13	23	172 096	125
M1	256	7	14	29	155 912	11.5	3	13	27	118 948	92.7
M1	512	5	14	29	103 508	5.94	5	14	26	78 774	72.5
M1	1 024	5	15	32	80 360	3.46	5	14	25	67 392	61.11
M2	1 024	7	15	30	271 951	25.1	4	15	27	206 942	192
M2	2 048	5	16	30	162 946	13.4	5	15	29	137 587	138
M2	4 096	4	16	30	106 734	8.01	4	15	28	97 510	111

words N_{max} is the maximum amount of data a processor communicates. We also give the time t for the partitioning in seconds. We see in both cases similar number of connections. The communication volume N_{max} is lower for PT-Scotch. Although PT-Scotch produces better partitions, we see that it does not scale. Its run times are several times longer than those of ParMETIS.

The matrices of the linear systems are ill-conditioned and large. Since they are symmetric and positive definite, we use the PCG [9] method, with a BoomerAMG as a preconditioner. The settings for the BoomerAMG preconditioner were carefully tuned for maximum scalability. The selected coarsening algorithm is *Falgout-CLJP. Modified classical interpolation* is applied. The selected relaxation method is *hybrid symmetric Gauss-Seidel or SSOR.* To decrease the operator and grid complexities two levels of aggressive coarsening are used. Smaller operator and grid complexities are lead to faster iterations and reduced memory requirements. This is essential on BlueGene/P, as each processor has only 512MB of RAM. The downside is that this affect the convergence rate of the solver.

In table 4 are shown parallel times for the discretization – T_{disc} and the solution of the linear system – T_{solve} for the bioheat problem 1. The matrix $A = M + \tau(K + M_{bl})$ from (5) is assembled only once on the first time step and not varied after that. The corresponding AMG preconditioner is also constructed only on the first time step. We see that the discretization is faster and the linear solver performs better for the partitions produced by PT-Scotch. The discretization part scales nicely. We see that the linear solver fails to scale for (relatively) small problems, and beyond 2 048 processors.

We asked ourselves the question "can we do better?" We tried to use the static graph mapping capabilities of the Scotch library. A mapping is called static if it is computed prior to the execution of the program. Static mapping is NP-complete in the general case [5]. Scotch library uses suboptimal method of Dual recursive bipartitioning. The parallel program to be mapped onto the target architecture

Table 4. Computation times

		ParMETIS		PT-Scotch	
mesh	N_{proc}	$T_{disc}[s]$	$T_{solve}[s]$	$T_{disc}[s]$	$T_{solve}[s]$
M0	128	515	54.6	471	51.2
M0	256	247	34.6	249	29.7
M0	512	138	23.8	133	23.4
M0	1 024	72.8	22.7	69.9	21.3
M1	128	4 225	332	4 007	319
M1	256	2 160	173	2 101	167
M1	512	1 196	101	1 131	97.9
M1	1 024	608	75.2	574	69.0
M2	1 024	5 381	356	4 856	337
M2	2 048	2 742	224	2 364	211
M2	4 096	1 322	204	1 215	207

is modeled by a weighted unoriented graph S called source graph or process graph, the vertices of which represent the processes of the parallel program, and the edges of which are the connections between communicating processes. The target machine onto which is mapped the parallel program is also modeled by a weighted unoriented graph T called target graph or architecture graph. The algorithm starts with the set of processors in T, also called domain, which is associated the set of all the processes in S to map. At each step, the algorithm bipartitions a yet unprocessed domain into two disjoint subdomains, and calls a graph bipartitioning algorithm to split the subset of processes.

In our case the we use $N_{i,j}$ from (6) as edge weights for S. We set the weights on the edges of T to be the number of hops between a pair of nodes plus one (there is a non-zero weight requirement in Scotch). We take into account that the mesh topology is mesh up to 1024 processors and torus in other two cases. The mapping is computed with the tool *gmap*, with an option which does not allow to assign two processes to one processor. In our solver, we construct an MPI communicator with reordered ranks to match the mapping. We use that communicator for the computations. We performed the same tests, bur with mapped communications. The results are shown in table 5. To ease the comparison only the ratios $T^{mapped}/T^{non-mapped}$ are shown in percents, where $T^{non-mapped}$ are the corresponding values in table 4.

In all cases the performance of the linear solver is improved. Although the improvements are small – under 2%, we can say that that the mapping has a positive effect. Things differ for the discretization times. For the ParMETIS partitionings, there is a small improvement in most cases, expect three. But for the PT-Scotch partitions things got worse in all cases. We expected that because of the algorithm PT-Scotch uses and because of the nature of the interconnect, its partitions were already properly mapped. But the slower run times can only mean that there is an overhead in using communicator different from `MPI_COMM_WORLD`.

Table 5. Computation mapping

		ParMETIS		PT-Scotch	
mesh	N_{proc}	T_{disc}[%]	T_{solve}[%]	T_{disc}[%]	T_{solve}[%]
M0	128	96.1	99.7	101.4	99.4
M0	256	98.1	100.0	102.2	99.6
M0	512	99.8	98.9	103.6	99.2
M0	1 024	100.1	99.7	103.6	99.7
M1	128	99.6	99.4	102.5	99.7
M1	256	99.1	99.8	101.6	99.2
M1	512	99.5	99.1	101.8	98.2
M1	1 024	101.9	99.1	104.1	98.1
M2	1 024	100.7	98.6	104.3	98.7
M2	2 048	98.7	99.7	104.2	99.7
M2	4 096	99.5	98.9	107.2	99.0

4 Conclusion and Future Work

In this work we compared the impact two graph partitioning libraries on the performance of parallel FEM simulations. We saw that partitioning quality has direct and non-negligible influence on the performance.

We used a tool for static graph mapping to optimize the communications on massively parallel computer. The results show that we could gain by this kind of optimizations, especially on problems with heavy communication loads.

Our future plans include tuning the parameters. Scotch library offers many possibilities to influence its partitioning results. This is also true for the static mapping. We also expect a parallel version of the routine `METIS_PartGraphVKway` from the Metis library, which directly minimizes total communication volume.

We intent to use process remapping method, different from currently used MPI communicator with reordered ranks.

Acknowledgments. This work is partly supported by the Bulgarian NSF Grants DCVP 02/1 and DPRP7RP02/13. We also kindly acknowledge the support of the Bulgarian Supercomputing Center for the access to the IBM Blue Gene/P supercomputer.

References

1. Karypis, G., Kumar, V.: Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and DistributedComputing 48(1) (1998)
2. Karypis, G., Kumar, V.: A coarse-grain parallel multilevel k-way partitioning algorithm. In: Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing (1997)
3. ParMETIS - Parallel Graph Partitioning and Fill-reducing Matrix Ordering, `http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview`

4. Henson, V.E., Yang, U.M.: BoomerAMG: A parallel algebraic multigrid solver and preconditioner. Applied Numerical Mathematics 41(1), 155–177 (2002)
5. Pellegrini, F., Roman, J.: Experimental analysis of the dual recursive bipartitioning algorithm for static mapping. Research Report, LaBRI, Universit e Bordeaux I (August 1996), http://www.labri.fr/~pelegrin/papers/scotch_expanalysis.ps.gz
6. Scotch and PT-Scotch: Software package and libraries for sequential and parallel graph partitioning, static mapping, and sparse matrix block ordering, and sequential mesh and hypergraph partitioning, http://www.labri.fr/perso/pelegrin/scotch/
7. Tungjitkusolmun, S., Staelin, S.T., Haemmerich, D., Tsai, J.Z., Cao, H., Webster, J.G., Lee, F.T., Mahvi, D.M., Vorperian, V.R.: Three-dimensional finite-element analyses for radio-frequency hepatic tumor ablation. IEEE Transactions on Biomedical Engineering 49(1), 3–9 (2002)
8. Tungjitkusolmun, S., Woo, E.J., Cao, H., Tsai, J.Z., Vorperian, V.R., Webster, J.G.: Thermal-electrical finite element modelling for radio frequency cardiac ablation: Effects of changes in myocardial properties. Medical and Biological Engineering and Computing 38(5), 562–568 (2000)
9. Axelsson, O.: Iterative Solution Methods. Cambridge University Press (1996)
10. Brenner, S., Scott, L.: The mathematical theory of finite element methods. Texts in Applied Mathematics, vol. 15. Springer, Heidelberg (1994)
11. Hairer, E., Norsett, S.P., Wanner, G.: Solving ordinary differential equations I, II. Springer Series in Comp. Math. (2002)
12. Kosturski, N., Margenov, S.: Supercomputer Simulation of Radio-Frequency Hepatic Tumor Ablation, AMiTaNS 2010 Proceedings. AIP CP, vol. 1301, pp. 486–493 (2010)
13. Lawrence Livermore National Laboratory, Scalable Linear Solvers Project, http://www.llnl.gov/CASC/linear_solvers/
14. Netgen Mesh Generator, http://sourceforge.net/apps/mediawiki/netgen-mesher/