Proceedings of the First International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2014)
Porto, Portugal

Jesus Carretero, Javier Garcia Blas
Jorge Barbosa, Ricardo Morla
(Editors)

August 27-28, 2014

On Efficiency of the OpenFOAM-based Parallel Solver for the Heat Transfer in Electrical Power Cables

Raimondas Čiegis, Vadimas Starikovičius, Andrej Bugajev

Vilnius Gediminas Technical University, Lithuania
raimondas.ciegis@vgtu.lt, vadimas.starikovicius@vgtu.lt, andrej.bugajev@vgtu.lt

Abstract

In this work, we study the efficiency of the OpenFOAM-based parallel solver for the heat conduction in electrical power cables. The 2D benchmark problem with three cables is used for our numerical tests. We study and compare the efficiency of conjugate gradient solver with diagonal incomplete Cholesky (DIC) preconditioner and generalized geometric-algebraic multigrid solver (GAMG), which is available in OpenFOAM. The convergence and parallel scalability of the solvers are presented and analyzed. Parallel numerical tests are performed on the cluster of multicore computers.

Keywords OpenFOAM, parallel algorithms, domain decomposition, preconditioner, multigrid

I. Introduction

The knowledge of heat generation and distribution in and around the high-voltage electrical cables is necessary to optimize the design and exploitation of electricity transferring infrastructure. Engineers are interested in maximum allowable current in different conditions, optimal cable parameters, cable life expectancy estimations and many other engineering factors.

Presently applicable IEC standards for the design and installation of electrical power cables are often based on the analytical and heuristic formulas. Obviously, these formulas cannot accurately account for the various conditions under which the cables are actually installed and used. They estimate the cable's current-carrying capacity (so-called *ampacity*) with significant margins to stay on the safe side [1]. The safety margins can be quite large and result in 50-70% usage of actual resources. A more accurate mathematical modelling is needed to meet the latest technical and economical requirements and to elaborate new, improved, cost-effective design rules and standards.

When we need to deal with mathematical models for the heat transfer in various media (metals, insulators, soil, water, air) and non-trivial geometries, only the means of parallel computing technologies can allow us to get results in an adequate time. To solve numerically selected models, we develop our numerical solvers using the OpenFOAM package [2]. OpenFOAM is a free, open source CFD software package. It has an extensive set of standard solvers for popular CFD applications. It also allows us to implement our own models, numerical schemes and algorithms, utilizing the rich set of OpenFOAM capabilities [3]. However, application of OpenFOAM libraries for solving specific problems still requires an appropriate theoretical and empirical analysis and a nontrivial selection of optimal algorithms. Examples of such problems are described in [4] and [5].

The important consequence of this software development approach is that obtained application solvers can automatically exploit the parallel computing capabilities already available in the OpenFOAM package. Parallelization of OpenFOAM is based on MPI (Message Passing Interface) standard. However, the modular structure of this package allows the development of parallel applications for modern hybrid and heterogeneous high-performance computing (HPC) platforms. See, for example, [6], for CUDA applications on GPU.

In recent years, scalability and performance of parallel OpenFOAM solvers are actively studied for various applications and HPC platforms. In [7] it is noted that the scalability of parallel OpenFOAM solvers is not very well understood for many applications when executed on massively parallel systems.

We note that an extensive experimental scalability analysis of selected OpenFOAM applications is one of the tasks solved in PRACE (Partnership for Advanced Computing in Europe) project, see [8], [9]. In [8] are presented results on IBM BlueGene/Q (Fermi) and Hewlett Packard C7000 (Lagrange) parallel supercomputers for a few CFD applications with different multi-physics models. The presented experimental results are showing a good scaling and efficiency with up to 2048–4096 cores. It is noted that such results are expected when balancing between computation, message passing and I/O work is good. Obviously, the next generation of ultrascale computing systems will cause additional challenges due to their complexity and heterogeneity.

In this work, we study and analyze the performance of OpenFOAM-based parallel solver for the heat conduction in electrical power cables. Two linear system solvers are considered and compared, namely, the conjugate gradient solver with diagonal incomplete Cholesky (DIC) preconditioner and generalized geometric-algebraic multigrid solver (GAMG), which is available in OpenFOAM. We study the convergence and parallel scalability of the linear solvers, the sensitivity of parallel preconditioners with respect to the grid size and the number of processes. Another goal is to study the ability of OpenFOAM to efficiently deal with the heterogeneity of the computer cluster.

In Section II, we shortly describe the problem, mathematical model and benchmark problem used for our parallel numerical tests. In Section III, we describe our OpenFOAM-based solver and discuss the parallelization approach employed in the OpenFOAM package. In Section IV, we present and analyze the obtained results on convergence and scalability of the considered linear solvers for our application. Finally, some conclusions are drawn in Section V.

II. Benchmark problem

As a benchmark problem in this research we solve the heat conduction problem for electrical power cables directly buried in the soil. It is also assumed that the thermo-physical properties of the soil remain constant, i.e. the moisture transfer in the soil is not considered. Such a simplified problem is described by the following well-known mathematical model:

$$\begin{cases} c\rho\dfrac{\partial T}{\partial t} = \nabla\cdot(\lambda\nabla T) + q, & t\in[0,t_{max}], \vec{x}\in\Omega, \\ T(\vec{x},0) = T_b, & \vec{x}\in\Omega, \\ T(\vec{x},t) = T_b, & \vec{x}\in\partial\Omega, \\ T, \lambda\nabla T \ \text{ are continuous}, & \vec{x}\in\Omega, \end{cases} \tag{1}$$

where $T(\vec{x},t)$ is the temperature, $c(\vec{x}) > 0$ is the specific heat capacity, $\rho(\vec{x}) > 0$ is the mass density, $\lambda(\vec{x}) > 0$ is the heat conductivity coefficient, $q(\vec{x},t,T)$ is the heat source function due to power losses, T_b is the initial and boundary temperature. Coefficients $\lambda(\vec{x})$, $c(\vec{x})$, $\rho(\vec{x})$ are discontinuous functions. Their real values can vary between metallic conductor, insulators and soil by several orders of magnitude [1].

In this work, we have used 2D geometry for our benchmark problem. Three cables are buried in the soil as shown in figure 1. The red area is metallic conductor, the blue area is an insulator and the gray area marks the soil.

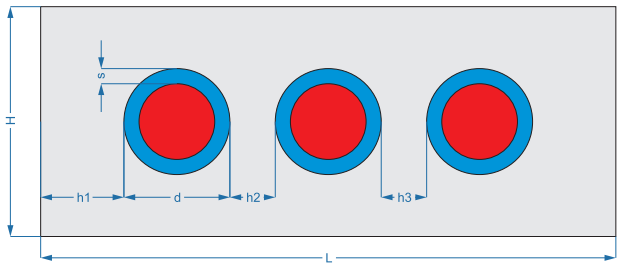

Figure 1: 2D geometry of benchmark problem: three cables in the soil.

III. Parallel OpenFOAM-based solver

OpenFOAM (Open source Field Operation And Manipulation) [2] is a C++ toolbox (library) for the development of customized numerical solvers for partial differential equations (PDEs). For numerical solution of PDEs OpenFOAM uses the Finite Volume Method (FVM) with co-located arrangement of unknowns [3].

We obtain a numerical solver for our benchmark problem (1) by the slight modification of the standard *laplacianFoam* solver: adding variable problem coefficients $c(\vec{x})$, $\rho(\vec{x})$, $\lambda(\vec{x})$ and a nonlinear source term $q(\vec{x},t,T)$ dependent on temperature T. Note that a proper interpolation should be used for calculation of discontinuous coefficient $\lambda(\vec{x})$ on conductor-insulator-soil interface walls of according finite volumes. We employ the *harmonic* interpolation to ensure the continuity of heat flux $\lambda\nabla T$.

We generate the finite volume mesh in problem domain Ω using OpenFOAM preprocessing tool and obtain FVM discretization with the four point stencil for the 2D problem. Resulting systems of linear equations with symmetric matrices can be solved by preconditioned conjugate gradient method with various preconditioners and multigrid method.

Parallelization in OpenFOAM is robust and implemented at a low level using the MPI library. Solvers are built using high level objects and, in general, don't require any parallel-specific coding. They will run in parallel automatically. Thus there is no need for users to implement standard steps of any parallel code: decomposition of the problem into subproblems, distribution of these tasks among different processes, implementation of data communication methods. A drawback of such automatic tools is that the user has very limited possibilities to modify the generated parallel algorithm if the efficiency of the OpenFOAM parallel code is not sufficient.

OpenFOAM employs a common approach for parallelization of numerical algorithms – domain decomposition. The mesh and its associated fields are partitioned into sub-domains, which are allocated to different processes. Parallel computation of the proposed finite FVM algorithm requires two types of communication. First type is the local communication between neighboring processes for approximation of the Laplacian term on the given stencil and matrix-vector multiplication. Second type is the global communication between all processes for computation of scalar products in iterative linear solvers.

OpenFOAM employs a zero-halo layer approach [6], which considers cell edges on sub-domain boundaries as boundary and applies a special kind of boundary condition.

OpenFOAM supports four methods of domain decomposition, which decompose the data into non-overlapping sub-domains: simple, hierarchical, scotch and manual [2]. For our parallel tests the mesh is partitioned by using Scotch library [10]. Scotch is a library for graph and mesh partitioning, similar to well-known Metis library [11]. It requires no geometric input from the user and attempts to minimize the number of boundary edges between subdomains. The user can specify the weights of the sub-domains, what can be useful on heterogeneous clusters of parallel computers with different performance of processors. We will use this feature solving our benchmark problem on heterogeneous cluster.

IV. Parallel performance tests and analysis

In our previous work [12], we have investigated the theoretical complexity and scalability of our OpenFOAM based parallel solver with preconditioned conjugate gradient method. To validate the theoretical model in numerical tests, we have fixed the number of iterations for solving systems of linear equations – 1000. In this way, we have ensured that the same amount of work was done in all parallel tests (10 time steps with 1000 iterations), despite the possible differences in convergence due to parallel preconditioning and different round-off errors, due to data communication.

In this study we want to investigate the convergence of conjugate gradient linear solver with diagonal incomplete Cholesky preconditioner (DIC/CG) and compare to the convergence of generalized geometric-algebraic multigrid solver (GAMG). We want to study the scalability of both linear solvers and the influence of the mesh partitioning on parallel preconditioners. So, now the tolerance of linear solvers is fixed and set to 10^{-6}. In each test, we do 10 time steps with different number of iterations, which is dependent on convergence.

	p	1×1	2×1	4×1	8×1	8×2
Mesh size - 1018488						
DIC/CG	N_p^{av}	721.9	839.3	865.4	942.4	1009.7
	T_p	265.1	161.7	77.0	41.1	28.5
	T_p^{av}	0.0367	0.0192	0.0089	0.0044	0.0028
GAMG	N_p^{av}	20.4	22.6	25.2	37.0	59.2
	T_p	49.7	30.7	20.3	25.3	44.8
	T_p^{av}	0.2436	0.1356	0.0807	0.0684	0.0757
Mesh size - 2048000						
DIC/CG	N_p^{av}	1015.6	1140.1	1142.6	1448.2	1401.6
	T_p	799.3	437.1	237.0	133.4	87.2
	T_p^{av}	0.0787	0.0383	0.0207	0.0092	0.0062
GAMG	N_p^{av}	19.4	27.0	27.7	39.0	39.3
	T_p	94.6	70.8	41.5	39.7	39.2
	T_p^{av}	0.4874	0.2622	0.1497	0.1017	0.0997
Mesh size - 4102264						
DIC/CG	N_p^{av}	1427.0	1452.0	1928.4	1939.4	1717.8
	T_p	2119.4	1114.9	872.2	414.6	265.8
	T_p^{av}	0.1485	0.0768	0.0452	0,0214	0,0155
GAMG	N_p^{av}	24.6	25.9	41.0	40.3	47.0
	T_p	243.9	137.0	120.2	68.9	69.7
	T_p^{av}	0.9913	0.5291	0.2932	0.1708	0.1482
Mesh size - 8192000						
DIC/CG	N_p^{av}	-	2468.9	2463.9	2535.1	2635.1
	T_p	-	3933.5	2165.9	1126.8	829.6
	T_p^{av}	-	0.1593	0.0879	0,0445	0,0315
GAMG	N_p^{av}	-	38.2	33.8	39.0	41.3
	T_p	-	402.5	193.3	123.67	100.6
	T_p^{av}	-	1.0536	0.5720	0.3171	0.2436

Table 1: The average number of iterations N_p^{av} with p processes, the total wall time T_p of 10 time steps with p processes, the average time of one iteration - $T_p^{av} = T_p/N_p^{av}/10$ with p processes for DIC/CG and GAMG linear solvers with tolerance 10^{-6}. Here $p = n_d \times n_c$ is the number of parallel processes using n_d nodes with n_c cores per node.

Parallel numerical tests were performed on the Vilkas cluster of Vilnius Gediminas technical university. We have used eight nodes with Intel Core i7-860 processors, 4 cores (2.80 GHz) per node. Computational nodes are interconnected via Gigabit Smart Switch.

In table 1, we present the average number of iterations N_p^{av}, the total wall time T_p of 10 time steps, the average time of one iteration - $T_p^{av} = T_p/N_p^{av}/10$ obtained with p processes. Results are presented for both linear solvers and increasing finite volume mesh. Size of the mesh (the number of finite volumes) was almost doubled for each next test.

Note that the case $p = 1\times1$ provides data on convergence and elapsed wall time for the sequential algorithms. There are no data for the mesh size 8192000, because this case does not fit into memory available on the single node. As we can see, the number of iterations of DIC/CG solver is increasing as $\sqrt{2}$ when the mesh size is doubled. This is in accordance with the theory. The number of iterations of GAMG solver is also growing with the mesh size, but not as much as in the case of DIC/CG, compare $24.6/20.4 \approx 1.21$ to $1427/721.9 \approx 1.99$ (2 expected).

Comparing the elapsed times T_p we see that the GAMG solver is a faster alternative that DIC/CG solver. The GAMG solver achieves the biggest advantage over the DIC/CG solver at sequential tests. For parallel tests, the advantage is constantly decreasing with increasing number of processes - p. It seems that there are two reasons for this. First reason is the degradation in performance of parallel preconditioner. For parallel DIC preconditioner with conjugate gradients, the number of iterations is quite gradually increasing up to 40% going from 1 to 16 processes. At the same time the number of GAMG iterations exibits significant jumps, altogether up to 2-3 times going from 1 to 16 processes.

Another reason for the significant degradation of parallel performance of GAMG solver is algorithmic scalability of its parallel algorithm. Comparing the average times of one iteration with p processes - T_p^{av}, we see that the parallel algorithm of DIC/CG solver scales much better than the parallel algorithm of GAMG solver.

Next, we want to study the ability of our OpenFOAM-based parallel solver to utilize the full power of heterogeneous cluster made of computational nodes of different speeds. For these numerical tests we have used eight additional nodes with Intel Quad Q6600 processors, 4 cores (2.4 GHz) per node.

Computing nodes with Intel Core i7-860 processors are up to 1.6-1.7 times faster than the Intel Quad Q6600 processors solving our benchmark problem (see [12]). To achieve the load balancing between $i7$ and q nodes, we employ the weights in the mesh partitioning algorithm from the Scotch library [10]. The performance results of the tests on heterogeneous cluster are presented in table 2.

Solving our biggest case in this work with 8192000 finite volumes, we obtain a real speedup only with DIC/CG linear solver. The performance of GAMG linear solver is further degrading. It is interesting to note that slight changes in the weighting factor w, can cause significant changes in the convergence of GAMG: from 43.2 to 51.7 iterations per time step with 16×1 processes.

V. Conclusions

We have tested the parallel performance of the conjugate gradient solver with diagonal incomplete Cholesky preconditioner (DIC/CG) and generalized geometric-algebraic multigrid (GAMG) solver in OpenFOAM-based application for the heat conduction in

Mesh size - 8192000							
	p	16×1	16×1	16×2	16×2	16×4	16×4
	$i7(w)$	1.6	1.7	1.6	1.7	1.6	1.7
DIC/CG	N_p^{av}	2340.7	2589.5	2790.7	2659.9	2630.2	2643.9
	T_p	647.2	728.6	574.6	536.6	540.7	524.7
	T_p^{av}	0.0277	0.0281	0.0206	0.0202	0.0206	0,0199
GAMG	N_p^{av}	43.2	51.7	53.8	55.2	56.2	60.7
	T_p	144.8	170.7	135.4	135.6	229.0	256.3
	T_p^{av}	0.3351	0.3301	0.2517	0.2457	0.4075	0.4223

Table 2: The average number of iterations N_p^{av} with p processes, the total wall time T_p of 10 time steps with p processes, the average time of one iteration - $T_p^{av} = T_p/N_p^{av}/10$ with p processes for DIC/CG and GAMG linear solvers with tolerance 10^{-6}. Here $p = n_d \times n_c$ is the number of parallel processes using n_d nodes with n_c cores per node, $i7(w)$ is the weighting factor used in mesh partitioning algorithm for faster i7 nodes.

electrical power cables. The best running times in our tests were obtained with GAMG solver. However, DIC/CG linear solver has shown to be less sensitive to the parallel preconditioning degradation. It has also shown better algorithmic scalability.

DIC/CG linear solver is to be recommended for the parallel computations on parallel computing systems with large number of processors and cores. Additional options of GAMG solver need to be investigated to improve its scalability. Parallel performance of the conjugate gradient solver with GAMG preconditioner (instead of DIC) is to be studied in future work.

The weighting factors in mesh partitioning algorithm allow efficient utilization of heterogeneous computing nodes for our parallel application. More research is needed on the influence of different graph partitioning algorithms.

Acknowledgment

The work of authors was supported by Eureka project E!6799 POWEROPT "Mathematical modelling and optimization of electrical power cables for an improvement of their design rules".

The work presented in this paper has been partially supported by EU under the COST programme Action IC1305, 'Network for Sustainable Ultrascale Computing (NESUS)'.

References

[1] I. Makhkamova, *Numerical Investigations of the Thermal State of Overhead Lines and Underground Cables in Distribution Networks*, Ph.D. thesis, Durham University, Durham, United Kingdom, 2011.

[2] OpenFOAM, *Open source Field Operation And Manipulation*, CFD toolbox, http://www.openfoam.org.

[3] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, "A Tensorial Approach to Computational Continuum Mechanics Using Object-oriented Techniques," *Journal of Computational Physics*, vol. 12, no. 6, pp. 620-631, 1998.

[4] P. Higuera, J. Lara, and I. Losada, "Realistic wave generation and active wave absorption for Navier–Stokes models: Application to OpenFOAM," *Coastal Engineering*, vol. 71, no. 1, pp. 102-118, 2013.

[5] O. Petit, A. Bosioc, H. Nilsson, S. Muniean, and R. Susan-Resigo, "Unsteady simulations of the flow in a swirl generator using OpenFOAM," *International Journal of Fluid Machinery and Systems*, vol. 4, no. 1, pp. 199-208, 2011.

[6] A. AlOnazi, *Design and Optimization of OpenFOAM-based CFD Applications for Modern Hybrid and Heterogeneous HPC Platforms*, Master thesis, University College Dublin, Ireland, 2013.

[7] O. Rivera, K. Furlinger, and D. Kranzimuller, "Investigating the scalability of OpenFOAM for the solution of transport equations and large eddy simulations," *Lecture Notes in Computer Science*, vol. 7017, pp. 121-130, 2011.

[8] P. Dagna, "OpenFOAM on BG/Q porting and performance", PRACE Report, CINECA, Bologna, Italy, 2012.

[9] M. Culpo, "Current Bottlenecks in the Scalability of OpenFOAM on Massively Parallel Clusters", PRACE White Papers, CINECA, Bologna, Italy, 2012.

[10] C. Chevalier and F. Pellegrini, "PT-Scotch: A Tool for Efficient Parallel Graph Ordering," *Parallel Computing*, vol. 34, no. 6-8, pp. 318-331, 2008.

[11] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359-392, 1999.

[12] R. Čiegis, V. Starikovičius, and A. Bugajev, "On Parallelization of the OpenFOAM-based Solver for the Heat Transfer in Electrical Power Cables," in *Euro-Par 2014: Parallel Processing Workshop*, Lecture Notes in Computer Science, vol. 8805, Porto, Portugal, August 25-29, 2014.