Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Application of CFD and CSM Open Source Codes for Solving Multiscale Multiphysics Problems

Tomáš Karásek[a], David Horák[a], Václav Hapla[a], Alexandros Markopoulos[a], Lubomír Říha[a], Vít Vondrák[a], Tomáš Brzobohatý[a]

[a]*IT4Innovations, VŠB-Technical University of Ostrava (VSB)*

Abstract

Solution of multiscale and/or multiphysics problems is one of the domains which can most benefit from use of supercomputers. Those problems are often very complex and their accurate description and numerical solution requires use of several different solvers. For example problems of Fluid Structure Interaction (FSI) are usually solved using two different discretization schemes, Finite volumes to solve Computational Fluid Dynamics (CFD) part and Finite elements to solve the structural part of the problem. This paper summarizes different libraries and solvers used by the PRACE community that are able to deal with multiscale and/or multiphysic problems such as Elmer, Code_Saturne and Code_Aster, and OpenFOAM. The main bottlenecks in performance and scalability on the side of Computational Structure Mechanics (CSM) codes are identified and their possible extension to fulfill needs of future exascale problems are shown. The numerical results of the strong and weak scalabilities of CSM solver implemented in our FLLOP library are presented.

1. Multiscale and muliphysics problem introduction

Let us start with the question "What is multiphysics?" and "What is multiscale?" The Wikipedia says in [1] and [2]:

"Multiphysics treat simulations that involve multiple physical models or multiple simultaneous physical phenomena. Multiphysics typically involve solving coupled systems of partial differential equations."

"In engineering, mathematics, physics, meteorology and computer science, **multiscale modelling** is the field of solving physical problems which have important features at multiple scales, particularly multiple spatial and(or) temporal scales. Multiscale modelling in physics is aimed to calculation of material properties or system behaviour on one level using information or models from different levels. On each level particular approaches are used for description of a system."

A typical example of multiscale modelling in CFD calculations is modelling of turbulent flow. Solving turbulent flow is quite a difficult task. There are an infinite number of degrees of freedom and velocities fluctuate in all directions. The flow is chaotic, three dimensional, diffusive, dissipative and intermittent. Governing equations describing it are elliptic, non-linear and coupled (pressure-velocity and temperature-velocity). The main obstacle in modelling turbulent flows is a quite wide range of length and time scales associated with turbulence which makes it a good example of multiscale problems. There are several approaches how to address this problem and each of them comes with its

own computational cost and accuracy. To solve turbulent flow with high accuracy Direct numerical simulation (DNS) has to be employed. In this case turbulence is numerically solved without any turbulence model and all spatial and temporal scales of turbulence must be resolved. To resolve even smallest turbulences a very fine grid must be created which makes DNS very computationally expensive even at low Reynolds numbers.

Numerical solution of multiphysics problems needs a fundamentally different approach to solution than multiscale problems. While multiscale problems combine different levels of scale, multiphysics combines various physical models. Currently two main approaches are used for solution of such problems: monolithic and partitioned one.

In case of the monolithic approach governing equations for all physics problems cast in terms of the same primitive variables and single discretization scheme is applied to the entire domain. This leads to solving of a single coupled matrix equation system. The monolithic approach is suitable for multiphysics problems with very strong interaction between individual physical phenomena. The advantage of the monolithic approach is its good stability and convergence properties. The disadvantage is that in some cases it may lead to ill-conditioned matrices due to non-optimal discretization and solution procedure of the unified spatial domain. Another disadvantage is that the monolithic approach requires a code developed for this particular combination of physical problems.

The partitioned approach on the other hand uses separate governing equation for each individual physical phenomenon. This allows us to separate the discretization scheme of each domain where exchange of data between domains is done via the domain interface. This leads to solving of separate matrix equations with distinct solvers for each domain. The advantage of the partitioned approach is that it preserves software modularity because existing solvers could be employed. Moreover, the partitioned approach facilitates solution with different, possibly more efficient techniques which have been developed specifically for solving particular physical phenomena. On the other hand, the development of a stable and accurate coupling algorithm is required in partitioned simulations.

Fluid structure interaction (FSI) is one of the multi-physics domains where the partitioned approach is widely used. The main reason is that usually the Finite Volume Method (FVM) is used for solving CFD problems and the Finite Element Method (FEM) is used for Computational Structure Mechanics (CSM) calculations. There are several CFD and CSM software packages developed, for example codes OpenFoam and Code_Saturne for CFD simulations or Code_Aster and Elmer for CSM calculations.

Unfortunately, those codes have limited capability in term of FSI simulations. In OpenFOAM icoFsiFoam solver could be used. The limitation of this approach is that it could be used only for linear models with small displacements. Other options such as the combination of Code_Saturne with Code_Aster or using Elmer for FSI simulation have similar limitations. In case of Elmer the main bottleneck is the limitation in terms of problem size which could be solved.

This paper is not intended to deal with the problem of coupling CFD and CSM codes. The partitioned approach is very well known and many research groups around the world are working on coupling of different codes as a part of their own research activities.

In this paper another issue arising with FSI calculations is addressed instead. This issue could be found in different domain sizes. It is very common that to solve accurately CFD problems meshes consisting of hundred millions or even billions of cells are required. On the other hand it's quite rare if CSM numerical model consists of meshes bigger than tens of millions of cells. Historically solving of CSM problems by numerical methods led to simplification and using of specially developed element types

such as shell elements for discretization of thin structures etc. This results in approaches where complex problems could be solved by relatively small numerical models.

The disproportion in term of domain sizes is even more evident in case of exascale computing. While solvers for CFD simulations are already capable of a parallel solution of very large problems with a very good scalability, the same cannot be said about CSM codes. In this paper highly scalable algorithms' implementations were tested on problems with 380M DOFS and 14,000 subdomains. These problem sizes are beyond the limits reachable by most of meshing tools, this is the reason, why we have implemented for this purpose our own generator PermonCube.

2. PRACE CFD and CSM codes overview

The **OpenFOAM**® [11] (Open Field Operation and Manipulation) CFD Toolbox is an open source CFD software package with an extensive range of features to solve anything from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics. It includes meshing tools for parallel mesh generation using complex CAD geometries. Almost everything in OpenFOAM (including meshing, and pre- and post-processing) runs in parallel by default, enabling users to take full advantage of computer hardware at their disposal. OpenFOAM could be customized and its capability and functionality can be extended by users to solve their specific problems. OpenFOAM includes over 80 solver applications that simulate specific problems in engineering mechanics and over 170 utility applications that perform pre- and post-processing tasks, e.g. meshing, data visualization, etc. This makes this code quite popular among users who created a large community of engineers and scientists exchanging their knowledge and expertise in various fields.

Code_Saturne [10] is a general purpose CFD software package based on FVM. It solves the Navier-Stokes equations for 2D, 2D-axisymmetric and 3D flows, steady or unsteady, laminar or turbulent, incompressible or weakly dilatable, isothermal or not, with scalars transport if required. Several turbulence models are implemented, from Reynolds-Averaged models (a. k. a. RANS models) to Large-Eddy Simulation models. In addition, a number of specific physical models are also available as "modules": gas, coal and heavy-fuel oil combustion, semi-transparent radiative transfer, particle-tracking with Lagrangian modelling, Joule effect, electrics arcs, weakly compressible flows, atmospheric flows, rotor/stator interaction for hydraulic machines. Code_Saturne can handle any type of cells tetrahedral, hexahedral, prismatic, pyramidal, polyhedral etc. or any type of grid such as unstructured, block structured, hybrid, conforming or with hanging nodes etc.

Code_Aster [9] is a free and open source software package for civil and structural engineering based on FEM. Most of the fields of the software have been validated by independent comparisons with analytical or experimental results, benchmarks towards other codes. It is provided with about 2,000 tests and examples. Code_Aster is mainly a solver for mechanics. Its capability covers a large range of applications: 3D thermal analyses and mechanical analyses in linear and non-linear statics and dynamics, for machines, pressure vessels and civil engineering structures. Beyond the standard functionalities of FEM software for solid mechanics, Code_Aster compiles specific research in various fields: fatigue, damage, fracture, contact, geomaterials, porous media, and multi-physics coupling. It is widely used at EDF for the expertise and the maintenance of power plants and electrical networks.

Code_Saturne can be coupled to thermal software SYRTHES for conjugate heat transfer. It can also be used jointly with structural analysis software *Code_Aster*, in particular in the Salomé platform.

All codes i.e. Code_Saturne, Code_Aster and SYRTHES are developed by EDF and distributed under the GNU GPL license.

Elmer [8] is open-source multiphysics simulation software developed by CSC - IT Center for Science in Helsinki, Finland. Elmer includes physical models of fluid dynamics, structural mechanics, electromagnetics, heat transfer, acoustics, etc. These are described by PDEs, which Elmer solves by the FEM. Currently Elmer has more than 5000 worldwide users. Elmer has shown excellent scaling on appropriate problems up to thousands of cores. Elmer developers focused on the implementation of a more robust solver, which enables further scaling of Elmer. The standalone tool ElmerSolver has implemented several types of solvers: time integration schemes for the first and second order equations, solution methods for eigenvalue problems, direct linear system solvers (Lapack & Umfpack), iterative Krylov subspace solvers for linear systems (GMRES, CG), multigrid solvers (GMG and AMG) for some basic equations, ILU preconditioning of linear systems, the discontinuous Galerkin method. Recently Elmer was extended by new FETI1 and TFETI domain decomposition methods implemented also via the FLLOP interface. For fluid flow ElmerSolver uses a stabilized (SUPG or residual free bubbles) Navier-Stokes equation. Elmer is suited for incompressible and compressible low Mach number flows. For turbulent flows Elmer includes some RANS models. Currently, the development of VMS LES models and segregated solvers is under way. The fluid flow may have coupling to thermal, electrostatic, magnetostatic or structural phenomena. For FSI-problems an ALE formulation is available also many different kinds of free surface models (Lagrangian and Eulerian) may be used. There is also a dimensionally reduced flow model, i.e. the Reynolds equation. Elmer may be used also in parallel and good scaling for CFD problems has been achieved up to hundreds of processors.

FLLOP (Finite Element Tearing and Interconnect (FETI) Light Layer On top of PETSc) [4] is a novel software package developed at IT4Innovations, VSB-Technical University of Ostrava, Czech Republic, for solution of quadratic programming problems (QP). It is an extension of PETSc, which is a suite of data structures and routines for the parallel solution of scientific applications modelled by PDE. FLLOP is designed to be modular and easy-to-use but at the same time efficient and targeted to HPC. The typical workflow looks like this: natural specification of the QP by the user, a user-specified series of QP transforms automatic or manual selection of a suitable solver, solution of the most derived QP by the solver, a series of reconstruction functions to get a solution of the original QP. Additionally, any combination of these constraints can be prescribed: equality, inequality and box constraints. A QP transform derives from the given original QP a new QP which is simpler or has some better properties. Currently, the implemented transforms are dualization, homogenization of the equality constraints, enforcing of equality using a penalty or a projector onto the kernel of the linear equality constraint matrix. Current concrete solvers are CG, DCG, MPGP, and SMALSE. The algebraic part of the TFETI domain decomposition method is considered a special QP transform. The overall scheme of FLLOP components is shown in Figure 1.

FllopAIF is a general pure C array based FLLOP interface. It is intended to be used within codes that do not use PETSc but can take advantage of the FLLOP solvers. The user which is going to apply FLLOP FETI solvers has to convert his/her matrices and vectors into standard arrays, include `fllopaif.h`, and then FLLOP solvers can be called from the (C or Fortran) code as shown below:

```
FllopAIFInitialize(comm,argc,args);
...
FllopAIFSetArrayBase(1);
FllopAIFSetFETIOperator(nb, i, j, A);
FllopAIFSetRhs(nb, b);
FllopAIFSetSolutionVector(nb,x);
FllopAIFSolve();
...
FllopAIFFinalize();
```

The final contribution is not only the scalability improvement of the code, but also functionality extension of the code enabling an efficient parallel solution QPs resulting from contact problems and other equality, inequality and box constrained QPs.

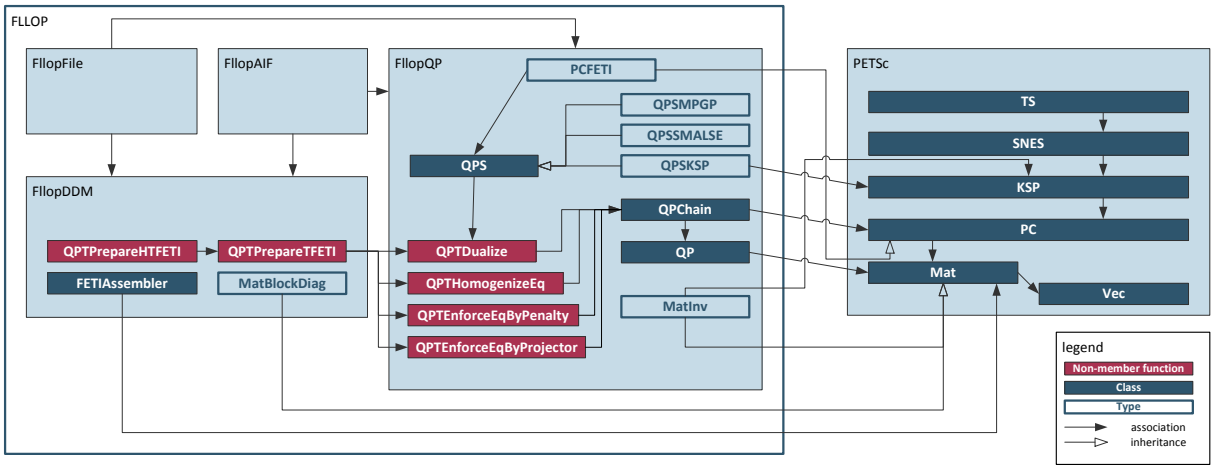

Figure 1: FLLOP components and PETSc integration

3. Numerical experiments and link to exascaling

We will demonstrate the capabilities of existing CFD and CSM solvers in term of solving real life multiphysics and multiscale problems on the example of a wind turbine – see Figure 2. Solving this very complex problem includes turbulence modelling, hence multiscale and deflection of propeller blades, an FSI problem. In our work we concentrated on the CFD (OpenFOAM) and CSM (FLLOP) simulation. We also performed very primitive one way FSI calculation where pressures on a blade in one particular time step were through I/O files passed to FLLOP as a loading for CSM calculation. As we mentioned earlier it was not the intention of our work to develop a coupler between two codes but this simple exchange of necessary data allowed us to use a real world example for our numerical experiments.

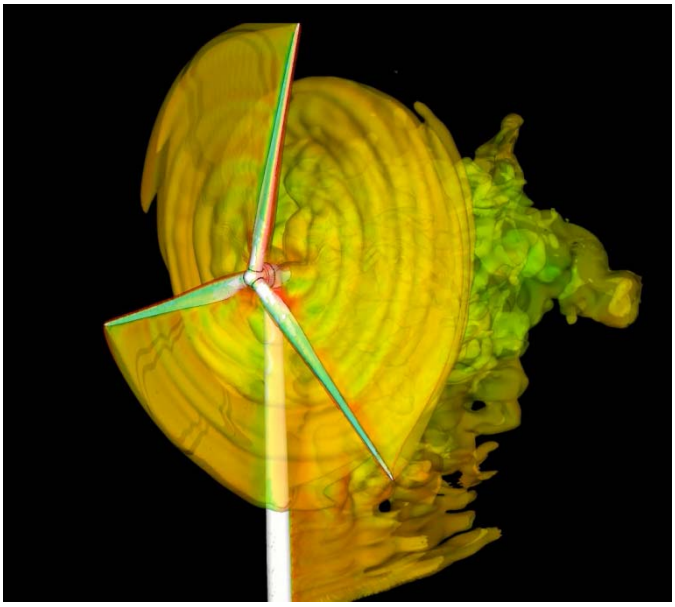

Figure 2: Wind turbine benchmark

Since our main interest is in identification of problems and solution methods for future exascale systems we focused on the strong scalability of CFD and CSM solvers. The reason is that in real engineering applications mesh size is usually fixed and computational time is of interest. Another typical feature mentioned already in this paper is that CSM meshes are much smaller than CFD meshes. In our case we modelled only one turbine blade for which approx. 10,000,000 of DOFS was needed to capture its physical behaviour. In contrast we needed more than 220,000,000 of cells

(1,300,000,000 of unknowns) on the CFD side to obtain results with requested accuracy. Figure 3 and Figure 4 show scalability of CFD (OpenFOAM) and CSM (FLLOP) solvers.

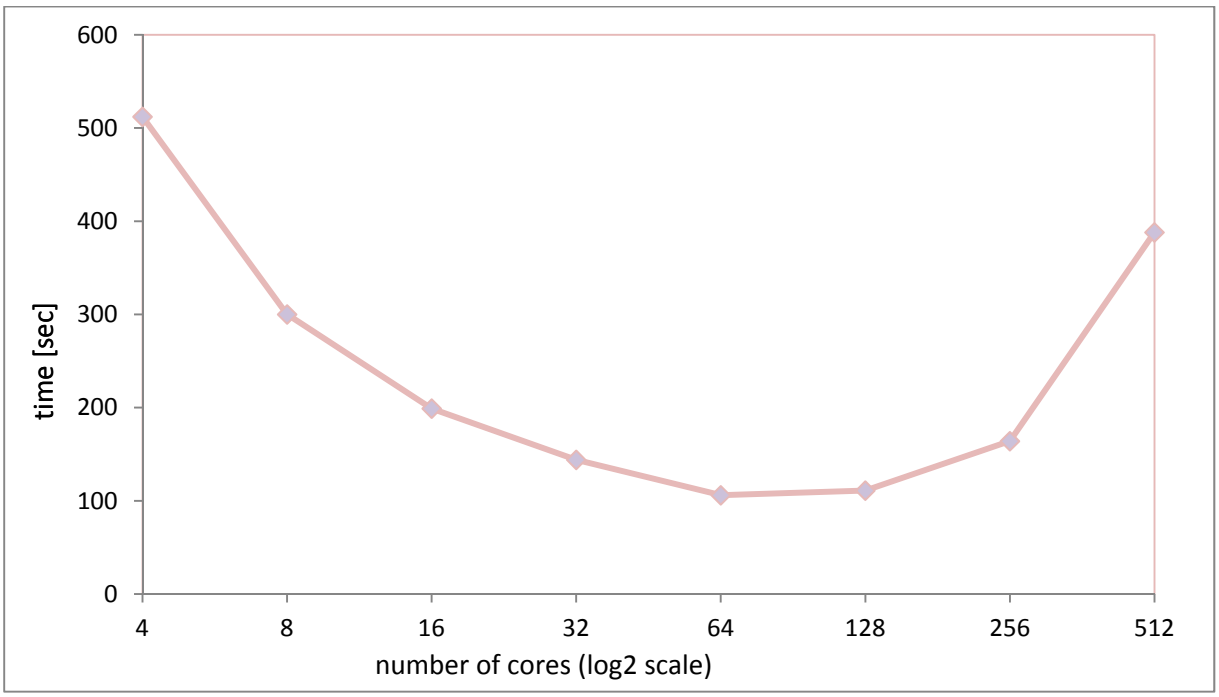

Figure 3: Strong scalability of the CSM solver for the wind turbine

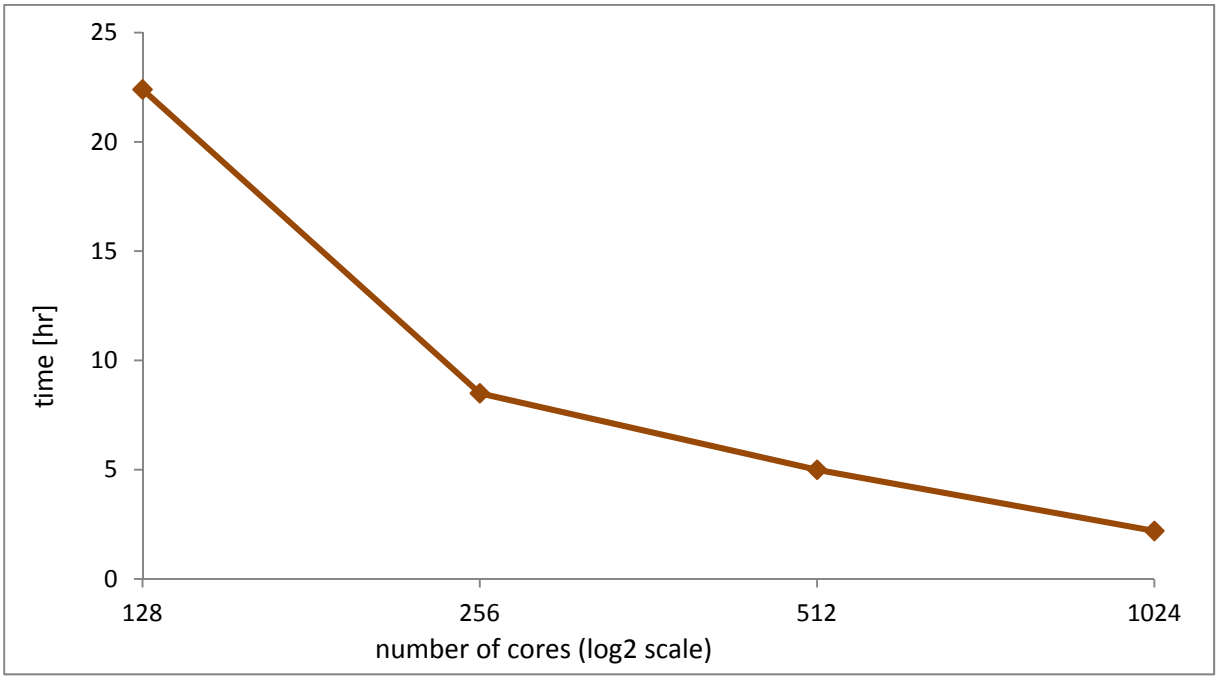

Figure 4: Strong scalability of the CFD solver for the wind turbine

From our numerical experiments we could observe almost ideal scalability of the CFD solver up to 1024 cores. The CSM solver on the other hand shows good scalability only up to 16 cores. This phenomenon could be explained by fact that the CSM mesh is too small and more time is spent on communication instead of calculation. Since this is not the first case where we have observed such a

behaviour we have decided to investigate whether this is really due to a problem size or whether it comes from the nature of the problem i.e. whether all CSM simulations regardless of the problem size would exhibit the same behaviour.

The team of researchers in Research Programme no. 3 at IT4Innovations is working in the field of efficient CSM algorithms development for many years. One of the successful classes of methods is based on domain decomposition into subdomains, definition of boundary problems, and an iterative process building the global solution from the local ones. **FETI** (Finite Element Tearing and Interconnecting) methods use the Lagrange multipliers and the original primal constrained problem is transformed into a significantly smaller and better conditioned dual constrained problem. For the dual problem solved by the conjugate gradient (CG) method the spectral condition number can be bounded by $const\frac{H}{h}$, with H denoting decomposition parameter and h the discretization parameter. This makes FETI methods the most efficient tool for the numerical solution of PDEs preserving both, high parallel and numerical scalabilities. The natural effort using the massively parallel computers is to maximize the number of subdomains (decrease H) so that sizes of subdomain stiffness matrices are reduced which accelerates not only their factorization and subsequent pseudoinverse application but also improves conditioning and reduces the number of iterations. The negative effect of that is an increase of a dual and null space (kernel) dimension, which decelerates the coarse problem (CP) solution being so the bottleneck of the FETI method.

To test the scalability of CSM solvers a numerical model of an **elastic cube** was used. There were two reasons for this decision. The elastic cube is the numerical model which could be fully controlled and obtained results won't be affected by the complexity of its geometry. Another reason is that it's very difficult or even impossible to create very large meshes on complex geometries using existing meshing tools. With our mesh generator PermonCube we were able to prepare large scale problems decomposed into thousands of subdomains. In our case, the loading was f_z = 77.0085 N/mm^3, Young's modulus E = 2.00e5 MPa, Poisson's ratio μ = 0.33. The model benchmark was decomposed into up to 13,824 subdomains and discretized up to 380M DOFs, the benchmark, its decomposition and total displacements are depicted in Figure 5.
The **weak scalability** for 13,824; 8,000 and 4,096 elements per subdomain and the **numerical scalability** for these configurations (corresponding to the fixed ratios $\frac{H}{h}$ =24, 20, 16) are then shown in Tables 1-3 and illustrated in Figure 6 and Figure 7 – the colour of each table corresponds with the line colour in the graphs. To investigate the **strong scalability** of CSM solvers we selected two different discretizations with 7,077,888 elements (approx. 22,000,000 unknowns) and 32,768,000 elements (approx. 100,000,000 unknowns), respectively, reported in Table 4, Table 5, Figure 8, and Figure 9. These results demonstrate the fact that the significant scaling for tens/hundreds cores can be reached if the local problems are sufficiently large.

The numerical experiments were run on **supercomputer HECToR** at EPCC, the Phase 3 system is contained in 30 cabinets and comprises of a total of 704 compute blades. Each blade contains four compute nodes giving a total of 2816 compute nodes, each with two 16-core AMD Opteron 2.3GHz Interlagos processors. This amounts to a total of 90,112 cores. Each 16-core socket is coupled with a Cray Gemini routing and a communications chip. Each 16-core processor shares 16 GB of memory. The theoretical peak performance of the Phase 3 system is over 800 Tflops.

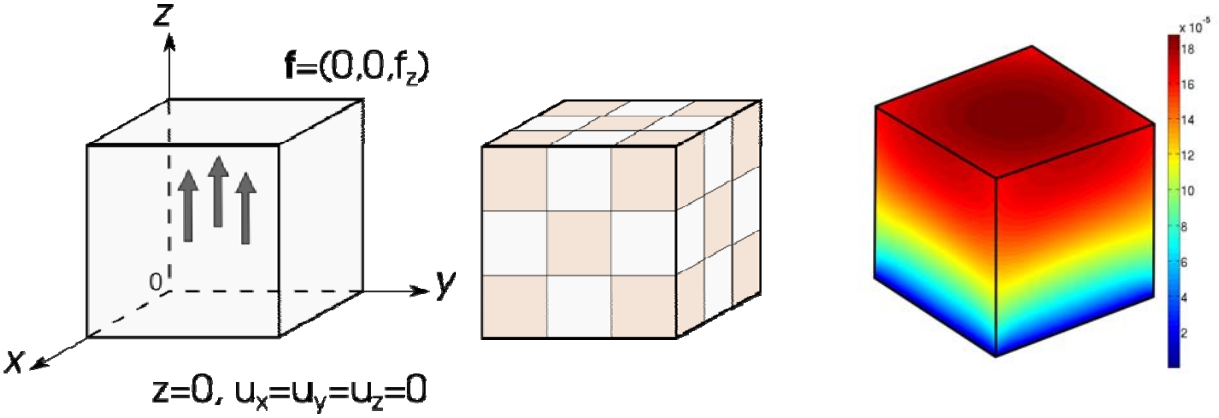

Figure 5: Cube benchmark, its regular decomposition and total displacements

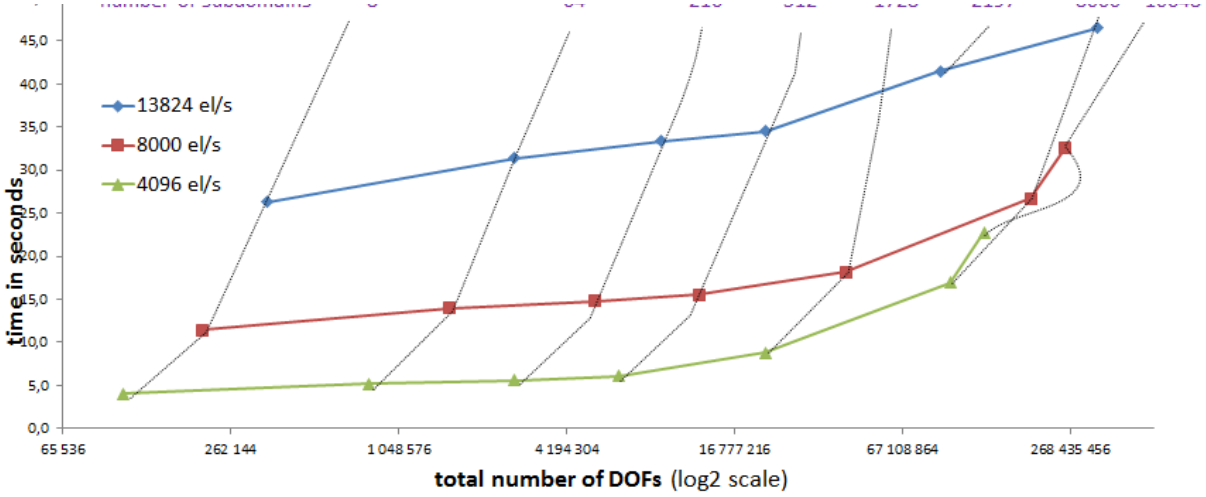

Figure 6: Weak scalability for 13,824; 8,000; 4,096 elements per subdomain

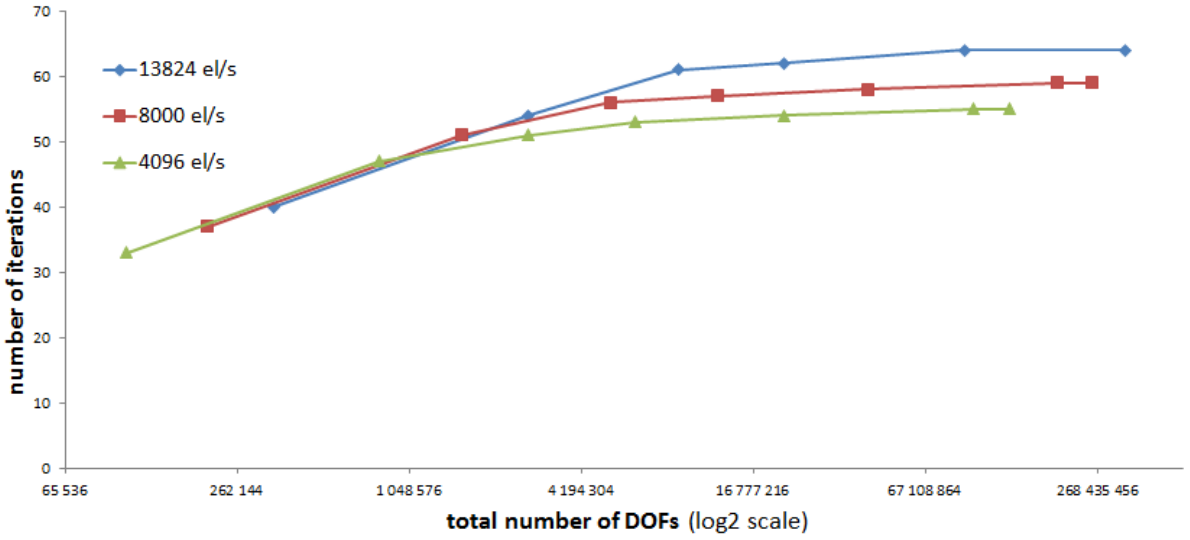

Figure 7: Numerical scalability for 13,824; 8,000; 4,096 elements per subdomain

NS1	NS	Ne all	prim.dim. #dofs	dual dim.	ker. dim.	CP red	CP proc /subc	K fact.	CP prep.	total prep.	all K+ act.	CP sol.	iter. sol.	#iter.	total fllop	CP fact+sol
2	8	110 592	375 000	30 912	48	1	8	12.8	0.0	13.8	12.4	0.0	12.5	40	26.3	0.01
4	64	884 736	3 000 000	316 524	384	1	64	13.4	0.0	14.8	16.4	0.4	16.6	54	31.4	0.43
6	216	2 985 984	10 125 000	1 149 000	1 296	1	216	13.2	0.2	14.6	18.5	0.3	18.8	61	33.4	0.50
8	512	7 077 888	24 000 000	2 820 516	3 072	28	18	13.5	0.3	15.2	18.8	0.5	19.3	62	34.5	0.81
13	2197	30 371 328	102 984 375	12 587 511	13 182	28	78	13.4	5.7	20.5	19.4	1.6	21.0	64	41.5	7.28
20	8000	110 592 000	375 000 000	46 831 308	48 000	28	286	13.4	8.7	23.8	19.6	3.0	22.7	64	46.5	11.74

Table 1: Weak scalability for 13,824 elements per subdomain

NS1	NS	Ne all	prim.dim. #dofs	dual dim.	ker. dim.	CP red.	CP proc/ subc.	K fact.	CP prep.	total prep.	all K+ act.	CP sol.	iter. sol.	#iter.	total fllop	CP fact+sol
2	8	64 000	222 264	21 936	48	1	8	5.3	0.0	5.8	5.6	0.0	5.6	37	11.4	0.01
4	64	512 000	1 778 112	225 612	384	1	64	5.5	0.0	6.3	7.6	0.2	7.7	51	14.0	0.28
6	216	1 728 000	6 001 128	820 248	1 296	1	216	5.4	0.2	6.3	8.3	0.2	8.5	56	14.8	0.41
8	512	4 096 000	14 224 896	2 015 076	3 072	28	18	5.5	0.3	6.6	8.5	0.3	8.8	57	15.5	0.68
12	1728	13 824 000	48 009 024	7 042 236	10 368	28	62	5.5	2.5	8.7	8.6	0.7	9.4	58	18.1	3.19
20	8000	64 000 000	222 264 000	33 505 068	48 000	28	286	5.5	8.4	14.9	8.8	2.8	11.8	59	26.7	11.17

Table 2: Weak scalability for 8,000 elements per subdomain

NS1	NS	Ne all	prim.dim. #dofs	dual dim.	ker. dim.	CP red.	CP proc/ subc.	K fact.	CP prep.	total prep.	all K+ act.t.	CP sol.	iter. sol.	#iter.	total fllop	CP fact+sol
2	8	32 768	117 912	14 496	48	1	8	1.8	0.0	2.0	2.0	0.0	2.0	33	4.1	0.01
4	64	262 144	943 296	150 060	384	1	64	1.9	0.0	2.3	2.8	0.1	2.9	47	5.2	0.15
6	216	884 736	3 183 624	546 792	1 296	1	216	1.8	0.2	2.4	3.0	0.2	3.2	51	5.6	0.40
8	512	2 097 152	7 546 368	1 344 804	3 072	28	18	1.9	0.4	2.6	3.2	0.3	3.5	53	6.1	0.60
12	1728	7 077 888	25 468 992	4 705 116	10 368	28	62	1.8	2.6	4.8	3.2	0.7	4.0	54	8.8	3.29
20	8000	32 768 000	117 912 000	22 406 028	48 000	28	286	1.8	8.7	11.2	3.3	2.3	5.8	55	17.0	10.97
22	10648	43 614 208	156 940 872	29 936 136	63 888	20	532	1.8	14.4	16.9	3.3	2.3	5.8	55	22.7	16.70

Table 3: Weak scalability for 4,096 elements per subdomain

NS1	NS	Ne all	prim. pim. #dofs	dual dim.	ker. dim.	CP red.	CP proc/ subc.	K fact.	CP prep.	total prep.	all K+ act.t.	CP sol.	iter. sol..	#iter.	total fllop	CP fact+sol
8	512	7 077 888	24 000 000	2 820 516	3 072	28	18	13.5	0.3	15.2	18.8	0.5	19.3	62	34.5	0.81
12	1728	7 077 888	25 468 992	4 705 116	10 368	28	62	1.8	2.6	4.8	3.2	0.7	4.0	54	8.8	3.29
16	4096	7 077 888	26 996 736	6 853 332	24 576	28	146	0.6	10.8	11.8	1.2	1.0	2.2	50	14.0	11.85

Table 4: Strong scalability for 7.7M elements

NS1	NS	Ne all	prim. pim. #dofs	dual dim.	ker. dim.	CP red.	CP proc/ subc.	K fact.	CP prep.	total prep.	all K+ act.	CP sol.	iter. sol.	#iter.	total fllop	CP fact+sol
13	2197	32 768 000	102 984 375	12 587 511	13 182	28	78	13.4	5.7	20.5	19.4	1.6	21.0	64	41.5	7.28
16	4096	32 768 000	113 799 168	16 980 948	24 576	16	256	5.4	3.6	10.0	8.8	1.1	9.8	59	19.8	4.68
20	8000	32 768 000	117 912 000	22 406 028	48 000	28	286	1.8	8.2	10.7	3.3	1.5	5.0	55	15.7	9.70

Table 5: Strong scalability for 33M elements

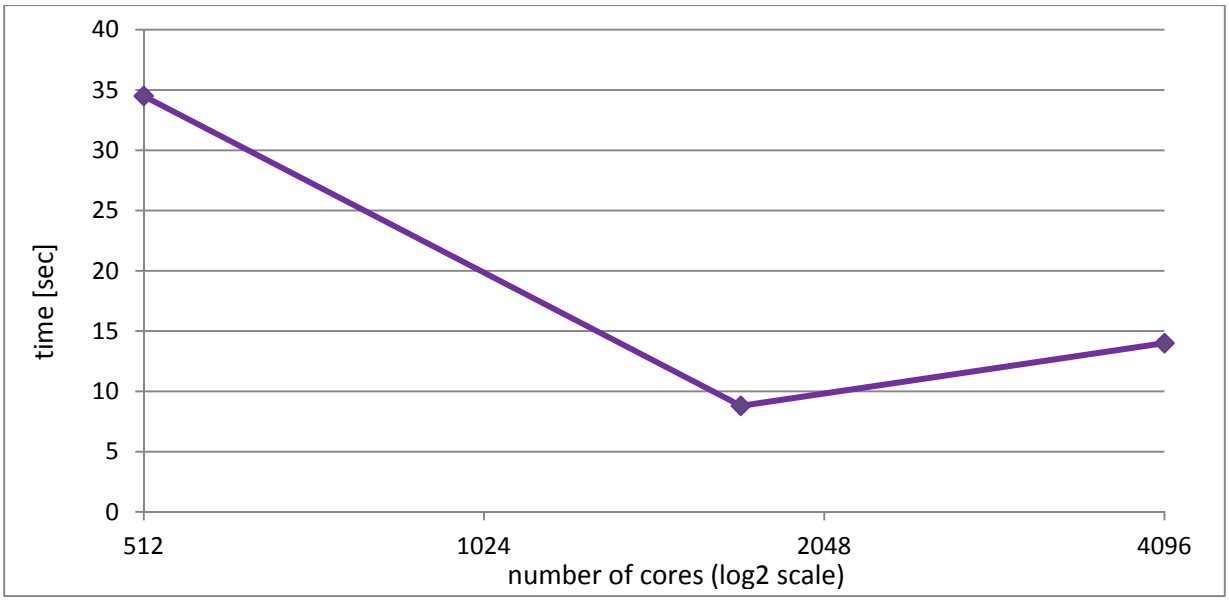

Figure 8: Graph of the strong scalability for 7.7M elements – total time [sec]

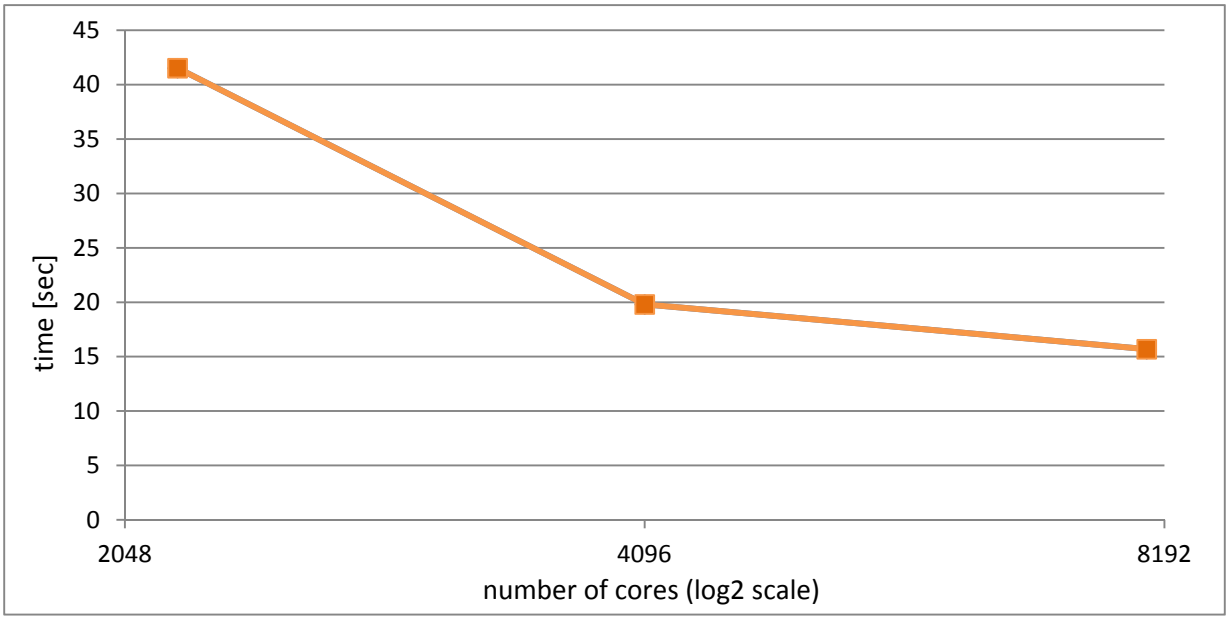

Figure 9: Graph of the strong scalability for 33M elements – total time [sec]

4. Conclusions

In this paper several codes capable of solving multiphysics multiscale problems were discussed. Their limitations in term of solving really large problems together with limitations in term of a mesh generation for complex geometries creates a bottleneck for multiphysics multiscale simulations of real world problems. We have proven that we are able to solve efficiently large-scale CSM problems with billions of unknowns using FETI methods and algorithms implemented in our FLLOP library. These solvers unlike those contained in other CSM packages preserve high parallel and numerical scalabilities. The FLLOP library provides the general FLLOP_AIF interface to other libraries such as Elmer and OpenFOAM, so that the employed solvers can enjoy scaling up to ten thousands cores for CSM problems, especially if local problems are large enough. This is very promising for the scaling of

FSI applications. To solve FSI problems the CFD-CSM coupler has to be at a disposal. It will also require an efficient mesh generator to prepare large meshes.

References

[1] http://en.wikipedia.org/wiki/Multiphysics
[2] http://en.wikipedia.org/wiki/Multiscale_modeling
[3] http://www.mcs.anl.gov/petsc/
[4] http://industry.it4i.cz/produkty/fllop/
[5] http://icl.cs.utk.edu/magma/software/
[6] http://crd-legacy.lbl.gov/~xiaoye/SuperLU/
[7] https://hpcforge.org/plugins/mediawiki/wiki/elmer/index.php/Docs
[8] http://www.csc.fi/english/pages/elmer
[9] http://www.code-aster.org/
[10] http://www.code-saturne.org/
[11] http://www.openfoam.org/

Acknowledgements

This work was financially supported by the PRACE-3IP project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-312763.
This publication was also supported by the Projects of major infrastructures for research, development and innovation of Ministry of Education, Youth and Sports (LM2011033), the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).