Energy consumption optimization of the Total-FETI solver by changing the CPU frequency

David Horak[1], Lubomir Riha[1], Radim Sojka[1], Jakub Kruzik[1], Martin Beseda[1], Martin Cermak[1,a)] and Joseph Schuchart[2]

[1]*IT4Innovations, VSB-TU Ostrava, 17. listopadu 15, Ostrava 70833, Czech Republic*
[2]*Technical University Dresden, Dresden, Germany*

[a)]Corresponding author: martin.cermak@vsb.cz

Abstract. The energy consumption of supercomputers is one of the critical problems for the upcoming Exascale supercomputing era. The awareness of power and energy consumption is required on both software and hardware side. This paper deals with the energy consumption evaluation of the Finite Element Tearing and Interconnect (FETI) based solvers of linear systems, which is an established method for solving real-world engineering problems. We have evaluated the effect of the CPU frequency on the energy consumption of the FETI solver using a linear elasticity 3D cube synthetic benchmark. In this problem, we have evaluated the effect of frequency tuning on the energy consumption of the essential processing kernels of the FETI method. The paper provides results for two types of frequency tuning: (1) static tuning and (2) dynamic tuning. For static tuning experiments, the frequency is set before execution and kept constant during the runtime. For dynamic tuning, the frequency is changed during the program execution to adapt the system to the actual needs of the application. The paper shows that static tuning brings up 12% energy savings when compared to default CPU settings (the highest clock rate). The dynamic tuning improves this further by up to 3%.

INTRODUCTION

This work is performed in the scope of the READEX project (Run-time Exploitation of Application Dynamism for Energy-efficient eXascale computing) [6]. The main goal of the participating institutions in the project is to develop an auto-tuning tool, which tries to improve energy efficiency of extreme-scale applications by employing techniques for dynamically adapting hardware (the frequency of CPU), system-level software, and application parameters of the processing platform. A key component of the project is the evaluation and exploitation of dynamic behavior in current HPC applications. In this paper we present the manual tuning of the FETI [5] solvers, which belong to the class of non-overlapping Domain Decomposition Methods (DDM). Each FETI solver can be divided into a preprocessing stage and an iterative solver runtime stage. In the preprocessing stage, it is necessary to factorize the stiffness and coarse problem matrices. Both of these tasks belong to the most time and also energy-consuming operations. The iterative solver employs the Conjugate Gradient (CG) algorithm, which in general consists of sparse and dense matrix-vector multiplications, vector dot products, and AXPY functions. In case of FETI, we need to apply a direct solver twice: once for the pseudoinverse action and once for coarse problem solution. We can achieve energy savings by exploiting the fact that different kernels have different computational characteristics and thus achieve minimal energy consumption at different frequency settings. We call this *dynamic tuning* in the scope of this paper.

At IT4Innovations, we are developing two in-house software libraries that implement various FETI methods. The first library is PERMON [1] and is based on PETSc [3]. The second library is called ESPRESO [2], which relies on lower level libraries such as Intel MKL and MPI and contains its own communication layer. Both libraries focus on real-world engineering applications as well as algorithmic testing using synthetic benchmarks, such as the one used in this paper, see Fig. 1.

There are two main phases in FETI – preprocessing and solve. In preprocessing, it is necessary to (i) regularize the stiffness matrix K and factorize it and to (ii) assemble the coarse problem matrix GG^T which also needs to be factorized. Both operations belong to the most time and also energy consuming operations. The solve employs the Preconditioned CG (PCG) algorithm, which is shown in Fig. 1 for the solution of $PF\lambda = Pd$ or $MPF\lambda = MPd$ and

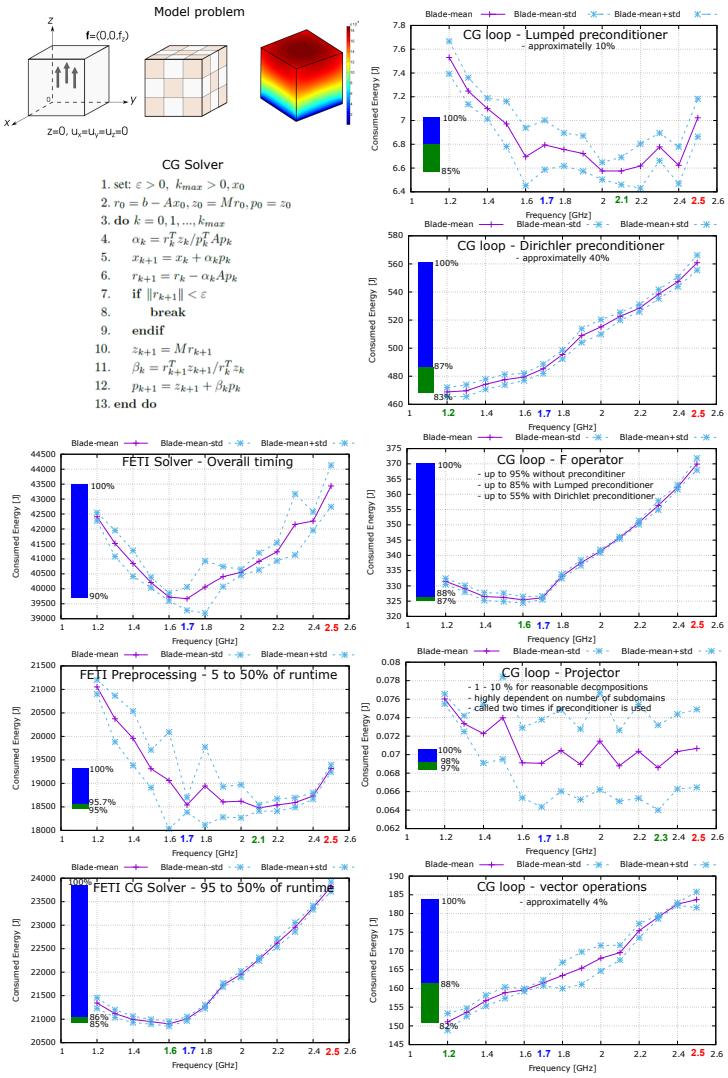

FIGURE 1. Model cube benchmark, PCG algorithm, energy savings of overall TFETI, preprocessing and PCG solve and its kernels obtained by static and dynamic frequency tuning.

consists of (i) sparse matrix-vector multiplications with the $F = BK^+B^T$ operator, the projector $P = I - G^T(GG^T)^{-1}G$, the lumped preconditioner - M_L, the Dirichlet preconditioner M_D and (ii) the vector dot products and AXPY functions. In each iteration, we need to apply the direct solver twice, i.e., for forward and backward solves for the pseudoinverse K^+ action in the F operator and for the coarse problem solution, the $(GG^T)^{-1}$ action in the projector P.

As a benchmark, the 3D linear elastic cube was used with the bottom face fixed, the top one loaded with a surface force, see Fig. 1 . For these computations, a mesh is generated and decomposed into subdomains by the PermonCube benchmark generator. The parallel mesh generator is controlled by two groups of parameters. The number of subdomains $N_S = XYZ$ is managed by parameters X, Y, Z, and similarly the number of elements per subdomain is given by x, y, z (each considered in the respective axis directions). Decomposition into large number of subdomains favorably affects the time of factorization of K and K^+ action. The negative effect is the increasing size of the coarse problem, whose solution becomes a critical part of this method for large number of subdomains. The model 3D elasticity problem was generated for our measurements by PermonCube with X=12, Y=9, Z=6, $x = y = z$=18, N_S=648, number of elements N_E=5,832N_S, primal dim. N_P=13,333,896, coarse problem dim. N_K=3,888, and the number of CG iterations as 56.

RESULTS

The experiments performed in this paper deal with CPU frequency. The measurements were performed on the Intel Xeon E5-2680 (Intel Haswell micro-architecture) based Taurus system installed at TU Dresden[1]. The system contains over 1400 nodes that have an FPGA-based power instrumentation called HDEEM (High Definition Energy Efficiency Monitoring), that allows for fine-grained and more accurate power and energy measurements [4]. The measurements can be accessed through the HDEEM library, allowing developers to take energy measurements before and after the region of interest. We have developed a library of measurement probes on top of the HDEEM library, that can: (i) start or stop the energy measurement and/or; (ii) change the CPU clock rate using the `cpufreq` library. With these probes we can measure the energy and thus average power consumption of any particular kernel or the entire solver, depending on their location.

We have used the HDEEM library version 2.1.5 for the measurements. The frequency is controlled using the `cpufreq` library and `cpufreq\_set\_frequency()` function. The dynamic frequency tuning can be performed by hard-coding the frequency values into every probe library call. For instance, if we know that the optimal frequency for a particular kernel 1 is f_1 we can setup the frequency manually. In case of a FETI solver evaluation, we need only a handful of probes, which can be easily inserted manually. The ultimate goal of the READEX project is an automatic tuning based on various input and runtime situations, which is beyond the scope of this paper. More details can be found in [7].

STATIC FREQUENCY TUNING: The relation of the CPU frequency and the consumed energy for the complete solution of the problem is depicted in Fig. 1 for (i) whole FETI (preprocessing and solving), (ii) preprocessing, and (iii) the CG solver. The graphs also contain standard deviations based on repeated measurements where the mean values were computed from 10 repetitions. Taking into consideration the energy savings using static tuning in Fig. 1 (left), we can see that the total saved energy in whole FETI – the difference between energy used with default frequency 2.5 GHz and minimum energy used with frequency 1.7 GHz – is $\approx 10\%$. This is in case the costs for the solve are comparable with preprocessing costs, which is the case for this rather small problem solved on one node only. If we need to solve the problem more precisely or if we need to solve time dependent problems, the preprocessing becomes negligible (it is performed only once at the start of the simulation). Hence, the solve will start to dominate. In the solve, we can see the difference between default frequency 2.5 GHz and the frequency with minimum energy 1.6 GHz is almost 13% saved energy. Comparing the best static tuning frequency 1.7 GHz to the dynamic frequency tuning with (i) 1.6 GHz for the solve phase and (ii) 2.1 GHz for the preprocessing phase, we can see that dynamic switching provides only a 0.6% and 0.4% reduction in energy consumption, respectively. To achieve better savings, we have to employ the dynamic tuning for the particular CG kernels.

DYNAMIC FREQUENCY TUNING: In order to further evaluate the dynamism in the FETI solvers we have performed energy consumption measurements of the main kernels of the solve phase, as shown in Fig. 1 (right). These kernels are: (i) Preconditioner action (lumped $M_L v$ or Dirichlet $M_D v$), (ii) Operator action Fv, (iii) Projector action Pv, and (iv) vector operations (AXPY). For the measured operations we have employed the following libraries: for $M_L v$ PETSc, for $M_D v$ MKL, for Fv PETSc + MUMPS Cholesky, for Pv PETSc + SuperLU_DIST, and for vector operations MKL. The optimal frequencies and the energy savings for the individual kernels compared to the best static configuration – 1.7 GHz – are provided in Tab. 1. In order to evaluate the overall potential of the dynamic tuning inside a single iteration of the FETI CG solver, we have to take into account the execution times of the respective kernels. For instance, even though the vector operations can save up to 13%, this saving is small since it takes less than 4% of a single iteration runtime. The most time-consuming regions are the matrix-vector multiplications by operator Fv and by the Dirichlet pre-conditioner $M_D v$ as they each account for 50% and 40% of the overall runtime, respectively. The projector runtime is highly problem-dependent and is mostly influenced by the number of subdomains. For the problem used in our tests, its runtime is small. In order to measure its effect, we would have to run it on hundreds of CPU cores. However, from our experience on large runs, the projector can take between 1-10% for reasonable configuration. The energy consumption characteristics of the projector operator will grow with the size of the coarse problem matrix. However, for this particular testcase performing dynamic tuning for this region does not yield any significant savings. We have summarized the overall savings achieved by the static and dynamic tuning in Tab. 1. This table shows that based on the best static tuning we can save further 2.68% by dynamic tuning of the CPU frequency as a single system parameter.

[1] `https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/SystemTaurus`

Action	Run-time	Static Tuning Savings	Overall Static Savings	Dynamic Tuning Savings	Optimal Frequency	Overall Dynamic Savings
Fv without M	90%	12%	**10.8%**	1%	1.6 GHz	**0.9%**
M_Lv	10%	3.2%	**0.32%**	3.1%	2.1 GHz	**0.31%**
Fv with M_L	80%	12%	**9.6%**	1%	1.6 GHz	**0.8%**
M_Dv	40%	13%	**5.2%**	4%	1.2 GHz	**1.6%**
Fv with M_D	50%	12%	**6%**	1%	1.6 GHz	**0.5%**
Pv	6%	2%	**0.12%**	1%	2.3 GHz	**0.06%**
AXPY	4%	13%	**0.52%**	6%	1.2 GHz	**0.52%**
Total savings:						
without M			**11.44%**			**1.48%**
with M_L			**10.56%**			**1.69%**
with M_D			**11.84%**			**2.68%**

TABLE 1. Efficiency of static and dynamic tuning for the FETI solve stage without and with both preconditioners.

CONCLUSION

We have presented our initial energy consumption measurements on one computational node with PERMON solver using synthetic 3D cube linear elasticity benchmark problems. We have reviewed the energy consumption of the whole FETI solver to find the optimal static tuning frequency. This step brought significant energy savings of 11.84%. The dynamic tuning has been performed on two levels. At coarse level, the preprocessing and solve phases have been evaluated dynamically. In this level there have been no energy savings achieved by dynamic tuning. On the fine-grained level, we have analyzed the behavior of the individual CG kernels. Based on our measurements, we have calculated the energy savings to be achieved by dynamic tuning as additional 2.68%. In total, the approach presented in this paper shows the potential to save up to 14.52% of energy for FETI based iterative solvers.

ACKNOWLEDGEMENTS

This work was supported by the READEX project - the European Union's Horizon 2020 research and innovation programme under grant agreement No 671657, The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602"; The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center LM2015070"; by the internal student grant competition project SP2016/178 "PERMON toolbox development II"; and by the Grant Agency of the Czech Republic (GACR) project no. 15-18274S.

REFERENCES

[1] PERMON webpages, http://industry.it4i.cz/en/products/permon/

[2] ESPRESO webpages, http://espreso.it4i.cz

[3] PETSc webpages, http://www.mcs.anl.gov/petsc

[4] Hackenberg, D., Ilsche, T., Schuchart, J., Schöne, R., Nagel, W., Simon, M., Georgiou, Y.: HDEEM: High Definition Energy Efficiency Monitoring. In: Energy Efficient Supercomputing Workshop (E2SC) (2014), `http://dx.doi.org/10.1109/E2SC.2014.13`

[5] Dostál, Z., Horák, D. and Kučera, R.: Total FETI – an easier implementable variant of the FETI method for numerical solution of elliptic PDE, Communications in Numer. Methods in Eng., Volume 22, number 12, pages 1155–1162, 2006

[6] Run-time Exploitation of Application Dynamism for Energy-efficient eXascale computing (READEX), `http://www.readex.eu/`

[7] Oleynik, Y., Gerndt, M., Schuchart, J., Kjeldsberg, P. G. and Nagel, W. E.: Run-Time Exploitation of Application Dynamism for Energy-Efficient Exascale Computing (READEX), Computational Science and Engineering (CSE), 2015 IEEE 18th International Conference on, Porto, 2015, pp. 347-350.