Energy Efficient High Performance Computing due to Application Dynamism

Lubomír Říha, Ondřej Vysocký, Martin Beseda, Jan Zapletal, Vojtěch Nikl and Tomáš Kozubek
IT4Innovations National Supercomputing Center, VSB Technical University of Ostrava,
Ostrava, Czech Republic

READEX
Runtime Exploitation of Application Dynamism
for Energy-efficient eXascale computing

Abstract

The energy consumption of supercomputers is one of the critical problems for the upcoming exascale supercomputing era. The awareness of energy consumption is required on both software and hardware side.

This poster presents the evaluation of selected applications and the potential effect of static and dynamic tuning of the *CPU core frequency (CF)*, *CPU uncore frequency (UCF)* and the *number of active cores (threads)* on energy consumption of HPC cluster with nodes equipped with two Haswell Xeon 12-core CPUs and 64 or 128 GB of RAM.

We have evaluated basic kernels with various computational intensity, parallel I/O and two full fledge applications: ESPRESO FEM library with FETI based solvers and well known open-source CFD package OpenFOAM.

H2020 READEX Project

Several measures that influence the energy consumed when running a software application on an HPC system are available to application developers, including hardware settings, system software parameters, and application characteristics. However, developers typically focus on implementing and optimizing algorithms for accuracy and performance and neglect possible improvements to the energy-efficiency of the application running on the HPC system.

The objective of the READEX project is to deliver the first stand-alone auto-tuning framework that has the capability to automatically and dynamically tune a large number of HPC applications at design- and run-time as we progress from deep-Petascale to Exascale computing. In developing such a tools-aided auto-tuning methodology the project aims to enable developers to achieve significant improvements in the energy-efficiency of current and future applications on extreme-scale systems, while at the same time significantly increasing productivity relative to manual tuning.

Tools for Manual Evaluation

Presently the MERIC tool is being developed and used in the READEX project to measure the dynamism metrics and energy consumption to evaluate applications. The measurements collected by this tools for an application are logged into a READEX Application Dynamism Analysis Report (RADAR).

Tools for Energy Measurements

- Running Average Power Limit (RAPL) interface – only CPU and RAM
- High Definition Energy Efficiency Monitoring (HDEEM) system – CPU, RAM and blade/compute node

Application Dynamism

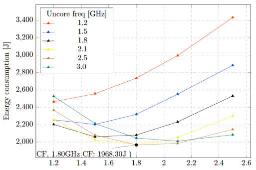
The dynamism observed in an application can be caused by a variation of the following factors:

- Floating point computations – variation in computational intensity
- Memory read/write access patterns - variation in the sparsity of matrices in sparse linear algebra
- Inter-process communication patterns
- I/O operations performed during the application's execution
- Different inputs to regions in the application.

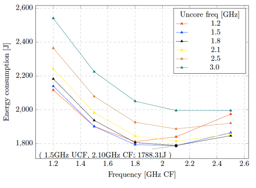
Computational Intensity Investigation

Workload type	Default settings	Default values	Best static configuration	Static Savings
DGEMV	12 threads, 3.0 GHz UCF, 2.5 GHz CF	2085.47 J	12 threads, 2.5 GHz UCF, 1.8 GHz CF	117.17 J (5.62%)
DGEMM	12 threads, 3.0 GHz UCF, 2.5 GHz CF	1995.29 J	12 threads, 1.5 GHz UCF, 2.1 GHz CF	206.98 J (10.37%)
Compute only	12 threads, 3.0 GHz UCF, 2.5 GHz CF	1666.32 J	12 threads, 1.2 GHz UCF, 2.5 GHz CF	212.51 J (12.75%)

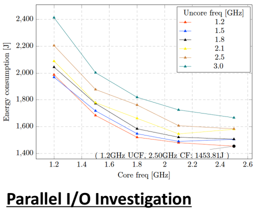
Low CI - DGEMV - 12 OpenMP threads



Higher CI - DGEMM - 12 OpenMP threads



High CI - compute only - 12 OpenMP threads



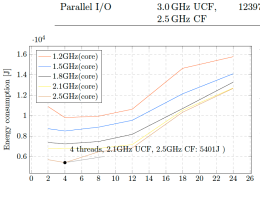
The computational intensity (CI) is one of the key metrics to evaluate the dynamism. If an application has a low CI, the application is memory bound (such as Matrix-Vector Multiplication - GEMV) and high CPU frequency cannot be utilized as the data in caches cannot be reused. On the other hand, for high arithmetical intensity (such as Matrix-Matrix vector multiplication - GEMM) the memory traffic is significantly lower and a CPU running at high frequency can be fully utilized.

One can observe that with increasing CI the effect of the uncore frequency becomes less important and the optimal setting is decreased from 2.5 GHz to 1.2 GHz. On the other hand the optimal core frequency should be high (2.5 GHz) for applications with high CI and the core frequency should be lower with low CI. It can be also observed that core frequency tuning is most efficient for kernels with high CI.

The highest static energy savings, 12.5%, have been achieved by compute bound codes while memory bounded code achieved only 5.6%.

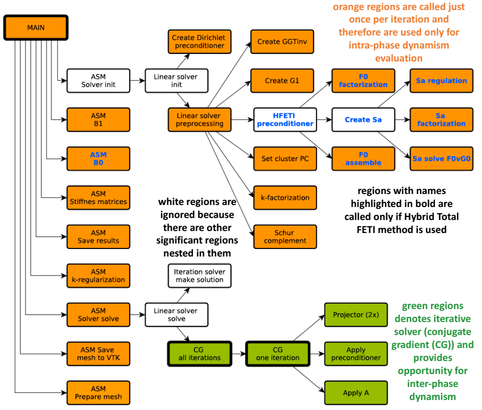
Parallel I/O Investigation

Workload type	Default settings	Default values	Best static configuration	Static Savings
Parallel I/O	12 threads, 3.0 GHz UCF, 2.5 GHz CF	12397 J	4 threads, 2.1 GHz UCF, 2.5 GHz CF	6996 J (56.43%)



To evaluate the parallel I/O we have developed a benchmark which reads a sparse matrix from a file.
The parallelization is done using OpenMP. The following results show the optimal setup for reading large amount of data from network file system on Taurus machine.

ESPRESO FEM Library Analysis



orange regions are called just once per iteration and therefore are used only for intra-phase dynamism evaluation

white regions are ignored because there are other significant regions nested in them

regions with names highlighted in bold are called only if Hybrid Total FETI method is used

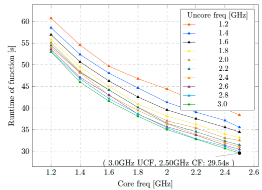
green regions denotes iterative solver (conjugate gradient (CG)) and provides opportunity for inter-phase dynamism

Diagram of the significant regions in the ESPRESO library

Static tuning

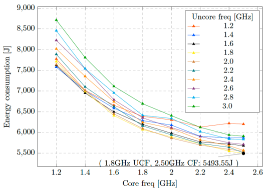
	Default settings	Default values	Best static configuration	Static Savings	Dynamic Savings
Energy consumption [J] , Blade summary	24 threads, 3.0 GHz UCF, 2.5 GHz CF	6265.18 J	18 threads, 1.8 GHz UCF, 2.5 GHz CF	771.63 J (12.32%)	499.2 J of 5493.6 J (9.09 %)
Runtime of function [s], Job info - hdeem	24 threads, 3.0 GHz UCF, 2.5 GHz CF	29.55 s	22 threads, 3.0 GHz UCF, 2.5 GHz CF	0.01 s (0.04%)	0.82 s of 29.54 s (2.76 %)

22 threads



Uncore freq [GHz UCF] / Core freq [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
1.2	60.727	58.527	56.92	55.813	55.095	54.454	53.975	53.517	53.023	52.953
1.4	54.553	52.357	50.638	49.499	48.44	48.232	48.128	47.066	46.746	45.975
1.6	49.634	48.031	46.226	44.984	44.163	44.134	42.875	42.999	42.153	41.551
1.8	46.755	44.614	42.516	40.89	40.118	40.101	39.405	38.555	38.611	38.067
2	44.359	41.281	39.536	38.114	37.025	36.431	36.352	35.646	35.404	35.005
2.2	41.544	39.015	37.503	36.214	35.329	34.505	33.865	33.714	32.878	32.779
2.4	39.773	37.102	35.515	33.968	33.053	32.318	31.931	31.36	31.106	30.634
2.5	38.339	35.518	34.437	33.109	32.481	31.425	30.983	30.433	29.862	29.536

18 threads



Uncore freq [GHz UCF] / Core freq [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
1.2	7,774.33	7,577.13	7,620.86	7,712.41	7,638.19	7,887.51	8,017.52	8,224.55	8,457.63	8,713.34
1.4	7,014.96	7,006.61	6,951.7	6,989.9	7,013.88	7,100.78	7,353.77	7,538.7	7,540.17	7,808.45
1.6	6,657.43	6,585.3	6,497.84	6,405.66	6,448.15	6,626.3	6,742.37	6,790.9	6,955.32	7,114.6
1.8	6,387.41	6,286.4	6,195.08	6,068.22	6,093.49	6,158.65	6,244.49	6,354.23	6,412.18	6,693.56
2	6,303.9	6,177.23	5,979.14	5,892.41	5,862.35	5,941.4	6,094.83	6,116.72	6,337.78	6,405.45
2.2	6,130.89	5,908.28	5,771.2	5,729.32	5,695.97	5,732.87	5,822.58	5,901.66	6,020.53	6,124.2
2.4	6,219.49	5,866.82	5,718.77	5,548.09	5,590.74	5,644.12	5,679.25	5,750.53	5,840.8	5,940.3
2.5	6,201.34	5,870.99	5,678.56	5,493.55	5,544.76	5,507.07	5,567.86	5,711.89	5,834.93	5,909.88

ESPRESO library is a combination of Finite Element (FEM) and Boundary Element (BEM) tools and TFETI/HTFETI solvers. It supports FEM and BEM (uses BEM4I library – http://bem4i.it4i.cz) discretization for advection-diffusion equation, atokes flow and structural mechanics.

The ESPRESO library contains both FEM preprocessing tools and sparse iterative solvers based on FETI method.

We have annotated more than 20 regions, which includes all types of operations including:

- I/O,
- MPI communication,
- sparse BLAS and
- dense BLAS.

The results show that static savings are 12.3% and dynamic savings 9.1%. The highest total savings for ESPRESO are 21.4%.

Dynamic tuning

Region	% of 1 phase	Best static configuration	Value	Best dynamic configuration	Value	Dynamic savings
Assembler–AssembleStiffMat	14.32	18 threads, 1.8 GHz UCF, 2.5 GHz CF	733.73 J	20 threads, 2.0 GHz UCF, 2.5 GHz CF	731.22 J	2.51 J (0.34%)
Assembler–Assemble-B1	2.23	18 threads, 1.8 GHz UCF, 2.5 GHz CF	114.30 J	2 threads, 2.2 GHz UCF, 2.5 GHz CF	94.15 J	20.15 J (17.63%)
Cluster–CreateF0-FactF0	0.17	18 threads, 1.8 GHz UCF, 2.5 GHz CF	8.71 J	6 threads, 1.6 GHz UCF, 2.5 GHz CF	6.90 J	1.80 J (20.73%)
Assembler–SaveResults	3.10	18 threads, 1.8 GHz UCF, 2.5 GHz CF	158.81 J	2 threads, 1.2 GHz UCF, 2.5 GHz CF	147.66 J	11.16 J (7.03%)
Assembler-K_Regularization	5.43	18 threads, 1.8 GHz UCF, 2.5 GHz CF	278.39 J	2 threads, 1.8 GHz UCF, 2.5 GHz CF	231.38 J	47.01 J (16.89%)
Cluster–CreateSa-SolveF0vG0	2.22	18 threads, 1.8 GHz UCF, 2.5 GHz CF	113.87 J	6 threads, 2.0 GHz UCF, 2.5 GHz CF	97.46 J	16.41 J (14.41%)
Create_-GGT_Inv	0.28	18 threads, 1.8 GHz UCF, 2.5 GHz CF	14.23 J	2 threads, 1.2 GHz UCF, 2.5 GHz CF	8.92 J	5.31 J (37.34%)
Cluster–Kfactorization	12.84	18 threads, 1.8 GHz UCF, 2.5 GHz CF	658.07 J	24 threads, 2.0 GHz UCF, 2.4 GHz CF	629.62 J	28.45 J (4.32%)
Assembler–SaveMeshtoVTK	6.36	18 threads, 1.8 GHz UCF, 2.5 GHz CF	325.69 J	2 threads, 1.2 GHz UCF, 2.5 GHz CF	296.66 J	29.03 J (8.91%)
Cluster–CreateSa-SaFactorization	1.95	18 threads, 1.8 GHz UCF, 2.5 GHz CF	99.93 J	4 threads, 2.2 GHz UCF, 2.5 GHz CF	80.85 J	19.08 J (19.09%)
Cluster–SetClusterPC	1.46	18 threads, 1.8 GHz UCF, 2.5 GHz CF	74.70 J	20 threads, 2.0 GHz UCF, 2.5 GHz CF	74.54 J	0.16 J (0.22%)
Assembler–PrepareMesh	12.53	18 threads, 1.8 GHz UCF, 2.5 GHz CF	641.88 J	22 threads, 1.8 GHz UCF, 2.5 GHz CF	639.39 J	2.49 J (0.39%)
Assembler–SolverSolve	30.79	18 threads, 1.8 GHz UCF, 2.5 GHz CF	1578.06 J	10 threads, 2.2 GHz UCF, 2.5 GHz CF	1289.85 J	288.21 J (18.26%)
Assembler–Assemble-B0	0.26	18 threads, 1.8 GHz UCF, 2.5 GHz CF	13.28 J	24 threads, 2.0 GHz UCF, 2.5 GHz CF	12.51 J	0.77 J (5.81%)
Cluster–CreateG1-perCluster	0.47	18 threads, 1.8 GHz UCF, 2.5 GHz CF	24.20 J	14 threads, 2.2 GHz UCF, 2.5 GHz CF	22.32 J	1.88 J (7.76%)
Cluster–CreateF0-AssembleF0	5.43	18 threads, 1.8 GHz UCF, 2.5 GHz CF	278.22 J	24 threads, 2.2 GHz UCF, 2.2 GHz CF	254.98 J	23.24 J (8.35%)
Cluster–CreateSa-SaReg	0.17	18 threads, 1.8 GHz UCF, 2.5 GHz CF	8.59 J	8 threads, 2.0 GHz UCF, 2.5 GHz CF	7.03 J	1.56 J (18.15%)
Total value for static tuning for significant regions			733.73 + 114.30 + 8.71 + 158.81 + 278.39 + 113.87 + 14.23 + 658.07 + 325.69 + 99.93 + 74.70 + 641.88 + 1578.06 + 13.28 + 24.20 + 278.22 + 8.59 = 5124.66 J			
Total savings for dynamic tuning for significant regions			2.51 + 20.15 + 1.80 + 11.16 + 47.01 + 16.41 + 5.31 + 28.45 + 29.03 + 19.08 + 0.16 + 2.49 + 288.21 + 0.77 + 1.88 + 23.24 + 1.56 = 499.22 J of 5124.66 J (9.74 %)			
Dynamic savings for application runtime			499.22 J of 5493.55 J (9.09 %)			
Total value after savings			4994.33 J (79.72 % of 6265.18 J)			

OpenFOAM Analysis

OpenFOAM is an abbreviation for Open source Field Operation And Manipulation. It is an open source C++ toolbox for computational fluid dynamics (CFD).

For the OpenFOAM investigation we have selected the simpleFoam application, the steady-state solver for incompressible flows with turbulence modeling.

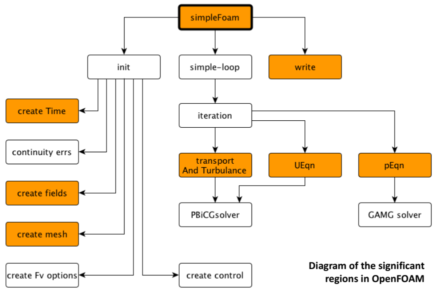
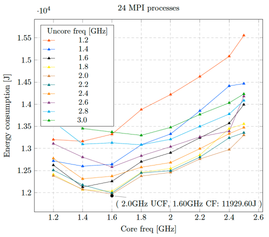


Diagram of the significant regions in OpenFOAM

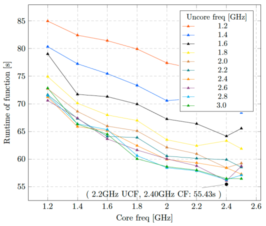
Static tuning

	Default settings	Default values	Best static configuration	Static Savings	Dynamic Savings
Energy consumption [J], Blade summary	3.0 GHz UCF, 2.5 GHz CF	14231.30 J	2.0 GHz UCF, 1.6 GHz CF	2264.94 J (15.92%)	207.54 J of 11966.36 J (1.73 %)
Runtime of function [s], Job info - hdeem	3.0 GHz UCF, 2.5 GHz CF	56.45 s	2.6 GHz UCF, 2.4 GHz CF	0.37 s (0.66%)	2.36 s of 56.08 s (4.20 %)

24 MPI processes



Uncore freq [GHz] / Core freq [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
1.2	84.976	80.339	79.028	74.925	72.716	71.654	71.226	70.598	71.485	72.839
1.4	82.421	77.248	71.714	70.108	68.613	67.333	65.866	67.414	66.378	66.305
1.6	81.434	75.465	71.304	67.975	65.964	64.097	65.142	63.646	65.33	64.499
1.8	79.927	73.333	69.946	66.98	65.096	63.895	62.421	61.626	60.613	60.035
2	77.398	70.564	67.241	63.494	62.111	60.541	59.925	60.035	58.424	58.617
2.2	76.263	71.037	66.391	62.403	60.939	60.121	59.311	58.759	57.863	57.992
2.4	74.449	71.009	64.141	63.308	58.418	59.89	58.406	56.076	56.27	56.461
2.5	75.975	68.4	65.549	61.872	59.241	58.514	57.273	58.729	57.091	56.447



Uncore freq [GHz] / Core freq [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
1.2	13,200.02	12,717.1	12,621.78	12,410.62	12,380.68	12,507.38	12,774.16	13,108.6	13,604.2	14,040.8
1.4	13,161.9	12,597.78	12,125.18	12,065.52	12,074.54	12,173.36	12,312.24	12,802.26	13,095.84	13,450.8
1.6	13,320.66	12,640.76	12,256.22	12,033.62	11,966.36	11,992.7	12,372.04	12,579.22	13,126.44	13,370.24
1.8	13,878.04	13,082.66	12,700.92	12,457.08	12,373.86	12,445.98	12,574.6	12,831.82	13,081.62	13,296.04
2	14,218.58	13,327.12	12,902.62	12,544.82	12,456.82	12,494.8	12,680.32	13,038.86	13,207.38	13,474.8
2.2	14,625.62	13,849.58	13,240.14	12,851	12,760.98	12,802.24	12,993.44	13,260.38	13,497.6	13,767.62
2.4	15,083.2	14,412.62	13,568.68	13,447.18	12,973.38	13,238.6	13,332.7	13,388.7	13,777.68	14,030.66
2.5	15,554.96	14,465.2	13,991	13,553.84	13,300.24	13,354.46	13,472.36	14,179.16	14,083.06	14,231.3

In case of the OpenFOAM application a simpleFoam solver was used on the motorBike benchmark, that is part of the OpenFOAM repository. The experiments were performed on a single node with 24 MPI processes.

The simpleFoam was set to use GAMG solver and PBiCG solvers. The results were written twice during the runtime into a binary uncompressed format file.

Since the most time consuming regions, the GAMG and PBiCG solvers, perform similar sparse BLAS operations, the optimal configuration for these regions is either identical or very similar.

Due to this reason the most of the savings, 15.9%, can be achieved by static tuning while only the remaining regions provide some potential for dynamic savings. Since the runtime of remaining regions is only 14.5% the overall dynamic savings are only 1.7%.

Dynamic tuning

Region	% of 1 phase	Best static configuration	Value	Best dynamic configuration	Value	Dynamic savings
init-createTime	0.03	2.0 GHz UCF, 1.6 GHz CF	3.35 J	1.4 GHz UCF, 1.4 GHz CF	2.64 J	0.71 J (21.06%)
init-createFields	4.28	2.0 GHz UCF, 1.6 GHz CF	506.91 J	2.4 GHz UCF, 2.0 GHz CF	474.80 J	32.11 J (6.33%)
init-createMesh	2.26	2.0 GHz UCF, 1.6 GHz CF	267.33 J	1.4 GHz UCF, 1.4 GHz CF	194.38 J	72.96 J (27.29%)
UEqn	40.71	2.0 GHz UCF, 1.6 GHz CF	4820.82 J	2.2 GHz UCF, 1.6 GHz CF	4810.03 J	10.80 J (0.22%)
pEqn	19.15	2.0 GHz UCF, 1.6 GHz CF	2268.19 J	2.0 GHz UCF, 1.6 GHz CF	2268.19 J	0.00 J (0.00%)
trans-portAnd-Turbulence	25.70	2.0 GHz UCF, 1.6 GHz CF	3042.91 J	2.0 GHz UCF, 1.6 GHz CF	3042.91 J	0.00 J (0.00%)
write	7.88	2.0 GHz UCF, 1.6 GHz CF	932.59 J	1.2 GHz UCF, 1.4 GHz CF	841.62 J	90.97 J (9.75%)
Total value for static tuning for significant regions			3.35 + 506.91 + 267.33 + 4820.82 + 2268.19 + 3042.91 + 932.59 = 11842.12 J			
Total savings for dynamic tuning for significant regions			0.71 + 32.11 + 72.96 + 10.80 + 0.00 + 0.00 + 90.97 = 207.54 J of 11842.12 J (1.75 %)			
Dynamic savings for application runtime			207.54 J of 11966.36 J (1.73 %)			
Total value after savings			11758.82 J (82.63 % of 14231.30 J)			

Conclusions

This poster introduces the READEX project and its main idea of dynamic application behavior. The main attention is paid to the manual applications evaluation from the energy consumption optimization point of view. This is the key step in exploring the possible gains of the runtime dynamic tuning.
The evaluation of two real-world applications ESPRESO and OpenFOAM shows that energy savings are 21.3% and 17.7%, respectively, as combination of static and dynamic tuning.

Acknowledgement

The research leading to these results has received funding from the European Union’s Horizon 2020 Programme under grant agreement number 671657.

References

[1] J. Schuchart et al., “The READEX formalism for auto- matic tuning for energy efficiency”, *Computing*, 2017, ISSN 1436- 5057.

[2] Y. Oleynik, M. Gerndt, J. Schuchart, P.G. Kjeldsberg, W.E. Nagel, “Run-Time Exploitation of Application Dynamism for Energy-Efficient Exascale Comput- ing (READEX)”, *Computational Science and Engineering (CSE), 2015 IEEE 18th International Conference on*, 2015.

[3] D. Hackenberg, T. Ilsche, J. Schuchart, R. Schone, W. Nagel, M. Simon, Y. Georgiou, “HDEEM: High Definition Energy Efficiency Monitoring”, in *Energy Efficient Supercomputing Workshop (E2SC)*, 2014.

[4] L. Riha, T. Brzobohaty, A. Markopoulos, O. Meca, T. Kozubek, “Massively Parallel Hybrid Total FETI (HTFETI) Solver”, in *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC ’16. ACM, 2016.

[5] Jasak, Hrvoje, Aleksandar Jemcov, and Zeljko Tukovic. "OpenFOAM: A C++ library for complex physics simulations." *International workshop on coupled methods in numerical dynamics*. Vol. 1000. IUC Dubrovnik, Croatia, 2007.

European Commission
Horizon 2020
European Union funding
for Research & Innovation