A Simple Framework for Energy Efficiency Evaluation and Hardware Parameter Tuning with Modular Support for Different HPC Platforms

Ondrej Vysocky, Jan Zapletal and Lubomir Riha
IT4Innovations, VSB – Technical University of Ostrava
Ostrava-Poruba, Czech Republic
{ondrej.vysocky|jan.zapletal|lubomir.riha}@vsb.cz

***Abstract*—High Performance Computing (HPC) faces the problem of the potentially excessive energy consumption requirements of the upcoming exascal machines. One of the proposed approaches to reduce energy consumption coming from the software side is dynamic tuning of hardware parameters during the application runtime. In this paper, we tune CPU core and uncore frequencies using Dynamic Voltage and Frequency Scaling (DVFS), and number of active CPU cores by means of OpenMP threads. For our research it is also essential that the HPC cluster contains infrastructure that provides energy consumption measurements. In this paper, we evaluate the energy consumption of an ARM-based platform with lower performance and even lower energy consumption, and two traditional HPC architectures based on x86 CPU architecture - Intel Xeon E5-26xx v3 (codename Haswell) and Intel Xeon Phi (codename KNL). To improve the efficiency and quality of such research we have developed a MERIC library. It enables both resource (time, energy, performance counters) usage monitoring and dynamic tuning of any HPC application that is properly instrumented. This library is designed to contribute minimal overhead to application runtime, and is suitable for analysis and tuning of both simple kernels and complex applications. This paper presents an extension of the library to support new architectures, (i) the low power ARMv8 based Jetson TX1 and (ii) the HPC centric Intel Xeon Phi (KNL) many-core CPU. The evaluation is carried out using a Lattice Boltzmann based benchmark, which shows energy savings on all presented platforms, in particular 20 % on Haswell processors.**

Keywords*—*Energy Efficient Computing; MERIC; HDEEM; RAPL; DVFS.

I. INTRODUCTION

High Performance Computing (HPC) is progressing to more and more powerful machines that, with current technology, would consume huge amounts of energy. This becomes one of the most significant constraints to building upcoming machines. For instance, an exascale machine based on Piz Daint (the most powerful European cluster) hardware would consume approximately 90 MW, based on a multiplication of the power of the current system and number of systems we would need to reach the exascale performance.

To reduce the power consumption of modern clusters, from the runtime system point of view it is possible to control selected hardware knobs to fit the needs of running applications. Memory bound (high data bandwidth) kernels have different requirements than compute bound (high CPU throughput) kernels. Within memory bound regions, it is possible to reduce the frequency of the CPU cores to reduce the power without extending the application runtime. In the same fashion, the frequency of the DDR memory controllers, the last level caches, and the DDR memory itself can be similarly reduced for compute bound regions.

For basic applications that generate a similar workload through the entire execution, one can use simple static tuning. For this approach one tunes the selected knobs at the application start, and they remain the same for the entire application runtime. However, more complex applications usually contains several regions with different workloads, and therefore with different optimal settings. These must be dynamically adjusted during the application runtime.

To find optimal settings in terms of energy consumption for particular HPC hardware, it is necessary to measure the consumed energy on different granularity levels. Almost all currently deployed HPC clusters based on Intel Xeon CPUs starting from the Sandy Bridge generation are equipped with Intel Running Average Power Limit (RAPL) counters [1]. The advantage of RAPL is fast access to energy counters with a very low overhead and quite a high sampling frequency of 1kHz. The main disadvantage is that it is able to measure only the energy consumption of the CPU and DDR memory but ignores the energy consumed by the rest of the compute node (main board, fans, storage, network card, etc.). This, called baseline energy consumption, must be accounted for by different means. The most straight forward way is to use a linear model, which predicts its power consumption based on the power consumption of the CPUs and memory.

Other more advanced measurement systems, such as High Definition Energy Efficiency Monitoring (HDEEM) [2], are based on additional hardware attached to the compute node, and provide out of bound energy measurements, which do interfere with running applications and do not introduce any additional overhead. These systems, in addition to the CPU and DDR memory, also measure the energy consumption of the entire compute node and thus provide all the necessary information for finding optimal settings.

Energy efficient high performance computing is an area of interest for several research activities, most commonly applying DVFS or power capping to the whole application run [3][4]. In this case only separate code kernels are extracted from the application and tuned. Complex application tuning is a goal of the Adagio project [5], presenting a scheduling system which changes the hardware configuration with a negligi-

 ISBN: 978-1-61208-655-2

ble time penalty based on previous application runs. Dynamic application tuning is the goal of the Horizon 2020 READEX (Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing) project [6] [7], which develops tools for application dynamic behavior detection, automatic instrumentation, and analysis of available system parameters configuration to attain the minimum energy consumption for the production runs.

Our tool called MERIC uses the same approach, with a focus on manual tuning, and is therefore more flexible for certain tasks. MERIC is a library for efficient manual evaluation of HPC applications' dynamic behavior and manual tuning from the energy savings perspective, applying the idea of dynamic tuning.

The goal of this paper is to present energy measurement and hardware parameters tuning using our MERIC tool on several different hardware platforms (two Intel Xeon E5-26xx v3 (code name Haswell - HSW) processors with different energy measurement systems (RAPL and HDEEM), Intel Xeon Phi (code name Knights Landing - KNL), and an experimental ARM platform). The approach is presented using the Lattice Boltzmann application benchmark.

This paper is organized as follows. Section II describes the MERIC library that was used for the application behavior analysis and runtime tuning. Section III describes the used hardware platforms and their energy measurement interfaces. Following on, Section IV presents experiments we performed on each hardware platform.

II. MERIC

MERIC [8][9] is a C++ dynamic library with a Fortran interface designed to measure resource consumption and runtime of analyzed MPI+OpenMP applications. In addition it can also tune selected hardware (HW) parameters during the application runtime.

MERIC automates hardware performance counters reading, time measurements, and energy consumption measurements for all user annotated regions of the evaluated application. These are called significant regions, and in general the different regions should have different workloads. The main idea of MERIC is that by running the code with different settings of tuning parameters multiple times, one can identify both optimal settings and possible energy savings for each significant region.

The supported system parameters in MERIC are:

- CPU core frequency,
- CPU uncore frequency,
- number of active CPU cores by means of number of OpenMP threads and thread placement, and
- selected application parameters (not used in this paper).

CPU uncore frequency refers to frequency of subsystems in the physical processor package that are shared by multiple processor cores e.g., L3 cache or on-chip ring interconnect. This parameter is not supported on Intel Xeon Phi processors.

The measurement results are analyzed using our second tool called RADAR [10]. This tool generates a detailed LaTeXreport of application behavior for different settings, and generates a final tuning model. The tuning model contains optimal settings for each significant region and it is used by MERIC for final runs of an application to perform dynamic tuning.

III. HPC PLATFORMS

Several current, and potentially future, HPC platforms are able to be tuned and analyzed for energy consumption by the MERIC library. Four of them are used to present the approach of dynamic tuning ofparallel applications.

The Technische Universität Dresden Taurus machine has nodes with Intel Haswell processors (2x Intel Xeon CPU E5-2680v3, 12 cores) [11] from the Bull company, which contain an energy measurement system called HDEEM [2] that has capability to measure energy consumption of the entire compute node with a sampling rate of 1kHz (the measurement error is approximately 2%).

HDEEM provides energy measurements in two different ways. In the first mode, HDEEM works as an energy counter (similar to RAPL), and by reading this counter we measure energy consumed from when HDEEM is initialized. Access to HDEEM measurements is through the C/C++ API. In this mode we read the counter at the start and end of each region. This solution is straightforward, however there is a delay of approximately 4 ms associated with every read from HDEEM. To avoid this delay, we take advantage of the fact that during the measurement HDEEM stores the power samples in its internal memory. The samples are stored without causing any additional overhead to the application runtime because all samples are transferred from HDEEM memory at the end of the application runtime. The energy is subsequently calculated from these samples based on the timestamps that MERIC records during the application runtime. The timestamps are associated with every start and end of significant regions.

Intel RAPL counters [1] are used to extrapolate the energy consumption. Both the RAPL and HDEEM energy measurement systems provide the same sampling frequency of 1 kHz, but the RAPL counters only approximate the energy consumption of the CPUs and RAM DIMMs, and do not take into account the energy consumption of the mainboard and other parts of the compute node. This fact may have major effect on the code analysis. If we were to measure only CPUs and RAM DIMMs without considering consumption of the rest of the node it would result in lower CPU frequencies and a longer runtime. however, the longer runtime leads to higher energy consumption of the entire node due to its baseline. Based on measurements made on the Taurus system, where the same type of hardware is present, the Haswell node baseline (the power consumption of the entire node without the power consumed by the CPUs and memory DIMMs) has been measured as 70 W. We add this constant to each measurement taken by RAPL to calibrate energy measurements.

Both energy measurement systems were compared on the Intel Haswell processor, which allows the CPU uncore frequency to be set in the range of 1.2–3.0 GHz and the CPU core frequency in range of 1.2–2.5 GHz, which we used for UFS

 ISBN: 978-1-61208-655-2

and DVFS in our Experiments section. Haswell experiments using RAPL counters were performed on IT4Innovations' Salomon cluster [12].

As a modern Intel platform, Intel Xeon Phi 7210 (KNL) supports energy consumption measurement using RAPL counters. Similarly to the case of Haswell nodes, we had to evaluate a node power baseline for more precise power consumption results. According to our measurements from Intelligent Platform Management Interface (IPMI), when not under a load the node consumes *75* W.

Xeon Phi platforms host GPU cards, and many smaller, less complex and low frequency cores. This is the reason why these cards provides a better FLOPs per Watt ratio than the usual x86 processors [13]. KNL nodes can also be tuned during the application runtime due to the changing number of active OpenMP threads (64 cores each with up to 4 hyper-threads) and a CPU core frequency which can scale from 1.0 GHz to 1.4 GHz. Uncore frequency tuning is not supported on KNL.

For KNL nodes it is possible to setup in a different memory mode, where MCDRAM works as a last-level cache (Cache mode), as an extension of RAM (Flat mode), or partially as a cache and partially as a RAM extension (Hybrid mode). Due to DVFS and energy measurement requirements, we had access to nodes in Cache mode only.

Another tested platform is Jetson/TX1, which is an ARM system (ARM Cortex-A57, 4 cores, 1.3 GHz), which is not a very powerful system, but which can set much lower frequencies than standard HPC systems, consequently allowing ARM systems to consume less energy. This fact makes such platforms interesting for further investigation. In the case of this system, it is not possible to set the uncore frequency, however, the user may change the frequency of the RAM. The minimum CPU core frequency is 0.5 GHz and the maximum is 1.3 GHz. The minimum and maximum RAM frequencies are 40 MHz and 1.6 GHz respectively.

TABLE I. JETSON/TX1 ENERGY MEASUREMENT INTERFACE AND ITS EFFECT ON THE CPU LOAD

frequency [Hz]	CPU load
10	2%
50	4%
100	8%
200	14%
500	23%
1000	30%

This system was selected from the available Barcelona Supercomputing Center ARM prototype systems under the Mont-Blanc project [14] because it is the only one that allows DVFS and supports energy measurements. To gather the power data from the board, the Texas Instruments INA3221 is featured on the board [15]. It measures the per node energy consumption and stores sample values in a file. It is possible to take approximately hundreds of samples per second, however the measurement runs on the CPU. Table I shows how the CPU is effected due to different energy measurement sampling frequencies, measured via the htop process-manager.

IV. EXPERIMENTS

In the following section we compare presented hardware platforms using the Lattice Boltzmann benchmark, which is a computational fluid dynamics application that describes flows in two or three dimensions.

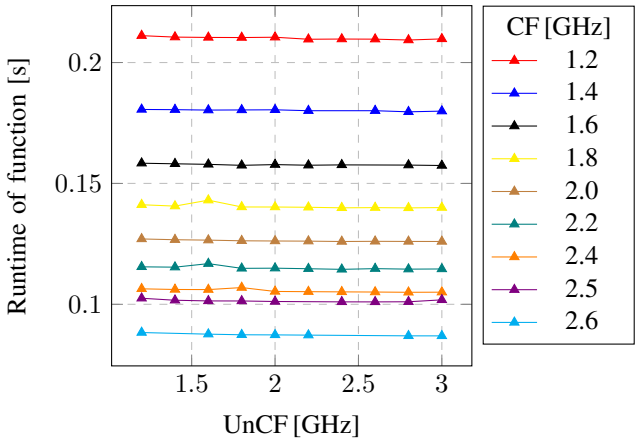

Figure 1: Collide region runtime when different CPU core and uncore frequencies are applied on a Haswell node.

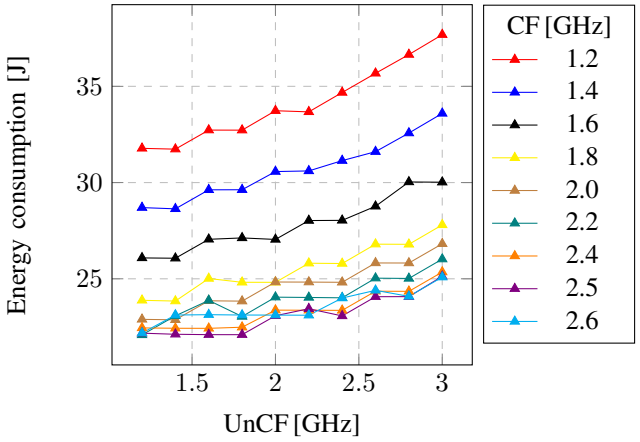

Figure 2: Collide region energy consumption when different CPU core and uncore frequencies are applied on a Haswell node.

The most significant parts of the code are functions called Collide and Propagate. Figure 1 shows the Collide region runtime for various CPU core and uncore frequencies on an Intel Xeon CPU (Haswell). The Collide region performs all the mathematical steps, so it is a typical compute-bound region, and its runtime is not effected by the uncore frequency. Figure 2 shows that on the other hand, the energy consumption is affected by increasing the uncore frequency, and that by reducing it to its minimum we save energy.

conversely, the Propagate region demonstrates very different behavior. It consists of a large number of sparse memory accesses, so it is highly affected by the uncore frequency, while core frequency has minimal impact on its runtime, as show in Figure 3. Also Figure 4 shows that the CPU core frequency can

 ISBN: 978-1-61208-655-2

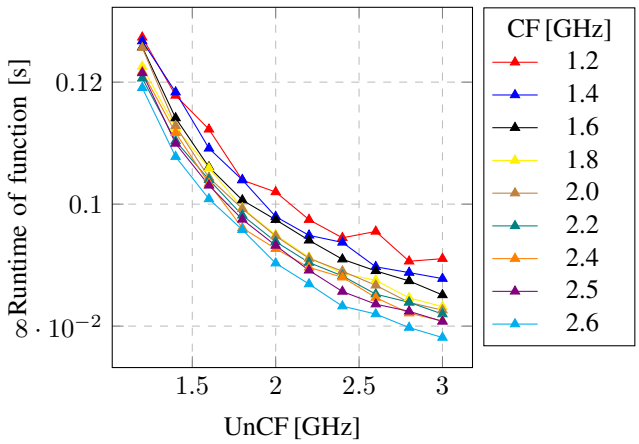

Figure 3: Propagate region runtime when different CPU core and uncore frequencies are applied on a Haswell node.

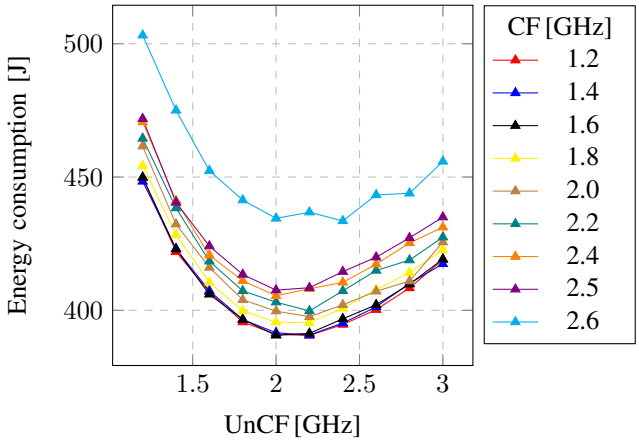

Figure 4: Propagate region energy consumption when different CPU core and uncore frequencies are applied on a Haswell node.

be reduced, but only to a specific minimum, since afterwards the energy consumption starts to grow again.

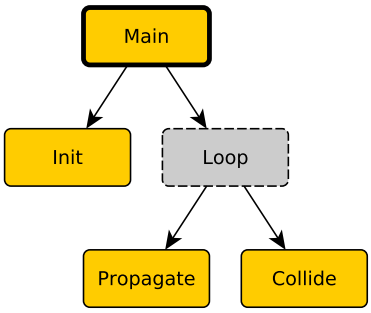

Figure 5: Diagram of significant regions in the LBM benchmark.

These two regions take most of the time of the main loop, however, an initialization of the application itself also takes several seconds, depending on the problem size, and we should not ignore this region. In this simple benchmark, we inserted

TABLE II. LBM APPLICATION REGIONS' OPTIMAL CONFIGURATIONS FOR THE ANALYZED PLATFORMS

Region	Parameter	HSW HDEEM	HSW RAPL	KNL	JTX
Main	threads [-]	24	24	256	4
	CF [GHz]	2.5	2.5	1.4	1.33
	UCF [GHz]	2.2	2.2	–	–
	RAMF [GHz]	–	–	–	1.1
Init	threads [-]	–	–	–	–
	CF [GHz]	2.6	2.6	1.4	1.33
	UCF [GHz]	1.4	1.6	–	–
	RAMF [GHz]	–	–	–	0.41
Propagate	threads [-]	6	6	128	4
	CF [GHz]	1.6	1.6	1.0	1.22
	UCF [GHz]	2.2	2.4	–	–
	RAMF [GHz]	–	–	–	1.6
Collide	threads [-]	24	24	256	4
	CF [GHz]	2.5	2.5	1.2	1.33
	UCF [GHz]	1.6	1.4	–	–
	RAMF [GHz]	–	–	–	0.41

four regions as illustrated in Figure 5. Please note that Loop region is not evaluated as it only repeatedly calls the Propagate and Collide regions.

A. Application Analysis

The application analysis runs the benchmark in the following configurations:

- Intel Xeon Haswell CPU (HSW) - CPU core frequency (1.2 – 2.6 GHz, step 0.2 GHz), CPU uncore frequency (1.2 to 3.0 GHZ, step 0.2 GHz), number of OpenMP threads (2 – 24 threads, step 2 threads);
- Intel Xeon Phi (KNL) - CPU core frequency (1.0 – 1.4 GHz, step 0.1 GHz), number of OpenMP threads (16, 32, 64, 128,192, 256 threads);
- Jetson TX1 - CPU core frequency (0.5 – 1.3 GHz, variable step), RAM frequency (0.04 to 1.6 GHZ, variable step), number of OpenMP threads (1 – 4 threads, step 1 thread).

First, we explore the analysis done on Intel Xeon Haswell processors, using different energy measurement systems. Table II presents the optimal configuration for the inserted regions. Despite the inaccuracy of the RAPL counters, the optimal configuration of the regions is very similar (the optimal configurations differs in one step of the analysis). The differences are caused by several factors: (i) small differences might be caused due to running the analysis on a different node; (ii) the proximity of the measured values; (iii) the baseline estimation for RAPL counters.

By evaluating the optimum configuration for each significant region while running the application with a domain size of *512x4096* for one hundred iterations, the tuned application has an approximately 1.5 % longer runtime, but consumes 19.8 % less energy.

Figure 6 shows the application behavior (energy consumption of the Main region) when running on Jetson/TX1 using different CPU core and RAM frequencies. The smallest possible RAM frequencies have a massive impact on application

 ISBN: 978-1-61208-655-2

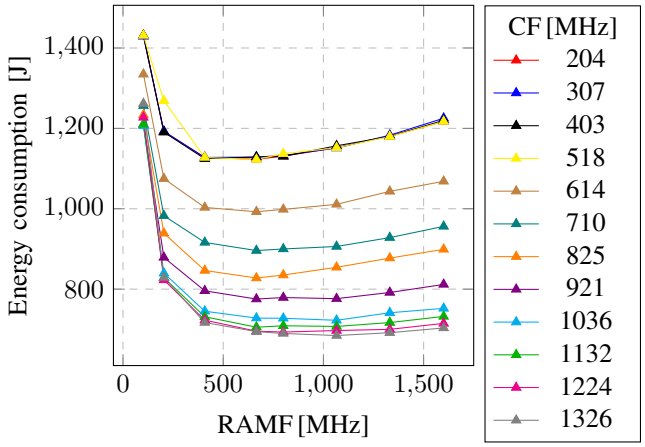

Figure 6: Jetson/TX1 application analysis comparing the energy consumption when applying various available CPU core and RAM frequencies.

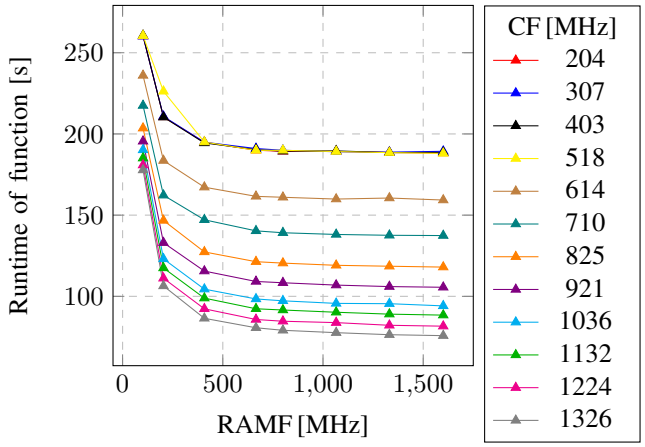

Figure 7: Jetson/TX1 application analysis comparing application runtime when applying various available CPU core and RAM frequencies.

runtime, as can be seen in Figure 7, which results in significantly higher energy consumption. In this case the graph of the application energy consumption follows the graph of application runtime. The minimum consumed energy was in the configuration 1065.6 MHz RAMF and 1326.00 MHz CF, when the application consumed 684.53 J.

From the analysis, see Table II, we can see the different behavior of th Propagate (optimum at 1600 MHz RAMF and 1224 MHz CF) and Collide (optimum at 408 MHz RAMF and 1326 MHz CF) regions. However the difference in RAM frequency does not translate into significant energy savings because the CPU core frequencies are quite similar for both regions.

When comparing tuned and non-tuned runs on Jetson the time remains the same while energy consumption drops about 4 %. When comparing to the same test case executed on Haswell CPUs where the tuned application finishes in 234 s and consumes 1992 J, the Haswell solved the same problem 10 times faster but consumed 2.3 times more energy than Jetson.

We evaluated the same test case on the Intel Xeon Phi 7210 (KNL), but since the test size was selected to run on Jetson/TX1, the test was too small for KNL node. In the *512x4096* domain size configuration, the Propagate and Collide regions take about 50 milliseconds only, which results very frequent frequency switching.

The overhead of switching CPU core frequencies on KNL limits its usage for such short regions. The Haswell processor can change the CPU core frequencies in about10 microseconds, while KNL nodes requires 20 milliseconds if it is done in the OMP parallel region and each thread is switching the frequency of its core. In case the master thread does the switching for all cores, it takes more than 50 milliseconds when using cpufreq library [16]. There are libraries providing slightly faster DVFS/UFS than cpufreq, such as the x86_adapt library [17], but these unfortunately currently do not support KNL.

Dealing with such overheads forces us to specify much higher restrictions for minimum region size. We have scaled the Lattice domain size to find the minimum problem size, which will provide some energy savings. When running the LBM simulation on a *4096x8192* domain, each Propagate and Collide region took 274.47 ms and 358.46 ms respectively. From this configuration we can see only a three percent longer runtime and one percent overall energy savings when running 100 iterations. Extending the region's size would continue the reduction of the DVFS overhead and increase the energy savings.

On KNL the sequential Init region becomes much more important because of the low single core performance of KNL. When running the application for 100 iterations on a *512x4096* domain on Haswell, this region takes less than 5 % of the application runtime. When running the same test case on KNL the initialization takes over 40 % of the application runtime. For a domain size of *4096x8192* elements, the initialization part takes longer than the Loop region with 100 iterations. Of course this ratio will differ as the number of iterations increases.

TABLE III. TABLE OF PRESENTED RESULTS COMPARING ENERGY SAVINGS FOR EACH PLATFORM WHEN RUNNING THE LBM BENCHMARK FOR 100 ITERATIONS

platform	default		static savings		dynamic savings	
domain	time	energy	time	energy	time	energy
HSW - S	24 s	5.7 kJ	-11.6%	7.8%	-1.5%	19.8%
JTX - S	233 s	2.1 kJ	-2.4%	2.6 %	-0.1%	4.0%
KNL - S	9 s	1.9 kJ	0%	0%	-2.8%	-3.6%
KNL - L	152 s	32.2 kJ	0%	0%	-3.0%	1.4%

Table III shows how much energy it is possible to save if one hardware configuration is set for the whole application run (static savings). This table shows the results for two different domain sizes - large (L) domain size represents a domain of 4096x8192 elements, the small (S) domain has 512x4096 elements. Despite having two very different regions in the application, on Haswell and Jetson/TX1 it is possible to save 7.8 % and 2.6 % energy respectively. This savings are reached

 ISBN: 978-1-61208-655-2

due to CPU uncore frequency (RAM frequency in the case of Jetson/TX1) reduction (HSW: 2.6 GHz; JTX: 1065 MHz), which explains why there are no static savings for KNL. Selected optimal static configuration is friendly for the Collide region, but extends the runtime of the Propagate region. When applying optimal configuration dynamically for each region (dynamic tuning) the application runtime, compared to non-tuned run, extends slightly, but energy savings further improves from static tuning.

V. Conclusion and Future Work

The MERIC library is a lightweight tool for the evaluation of resource consumption and dynamic hardware parameters tuning. The library is focused on the tuning of complex applications without rewriting the application itself. It is continually extended to support different and experimental platforms. Two Intel Xeon CPU E5-26xx v3 (codename Haswell) based machines with RAPL and HDEEM energy measurement systems, one Intel Xeon Phi (codename KNL) system with RAPL counters, and finally a Jetson/TX1 with an INA3221 system have been presented and compared using the Lattice Boltzmann benchmark.

Tuning achieved up to 20 % energy savings with a 1 % longer runtime for Haswell nodes. On the Jetson and KNL nodes there are several restrictions that limit reachable gains. We attained about 4 % energy savings without any runtime penalty in case of the Jetson TX1 system. For KNL it was possible to reach savings only if the problem size had been scaled, and regions' sizes extended sufficiently to overcome the problem of system slow frequency switching.

The ARM platforms become more and more interesting for future HPC systems builders, because of their low energy consumption. The possibility to tune within a wide range of frequencies seems interesting in the case of Jetson/TX1. Despite the limited performance of its CPU cores, it was able to provide the simulation result, and consumed a significantly smaller amount of energy than a usual HPC node powered with two Intel Xeon CPUs.

Many more energy efficient platforms are coming to the market, especially new ARM and IBM systems, as well as GPU cards, and we would like to extend the MERIC library and provide support for their hardware tuning.

Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project IT4Innovations National Supercomputing Center LM2015070.

This work was supported by the READEX project - the European Union's Horizon 2020 research and innovation programme under grant agreement No. 671657.

This work was partially supported by the SGC grant No. SP2018/134 "Development of tools for energy-efficient HPC applications", VSB - Technical University of Ostrava, Czech Republic.

This work was supported by Barcelona Supercomputing Center under the grants 288777, 610402 and 671697.

References

[1] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, "Measuring energy consumption for short code paths using rapl," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 3, pp. 13–17, Jan. 2012. [Online]. Available: http://doi.acm.org/10.1145/2425248.2425252

[2] D. Hackenberg, T. Ilsche, J. Schuchart, R. Schne, W. E. Nagel, M. Simon, and Y. Georgiou, "HDEEM: High definition energy efficiency monitoring," in *2014 Energy Efficient Supercomputing Workshop*, Nov 2014, pp. 1–10.

[3] A. Haidar, H. Jagode, P. Vaccaro, A. YarKhan, S. Tomov, and J. Dongarra, "Investigating power capping toward energy-efficient scientific applications," *Concurrency and Computation: Practice and Experience*, p. e4485. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4485

[4] J. Eastep, S. Sylvester, C. Cantalupo, B. Geltz, F. Ardanaz, A. Al-Rawi, K. Livingston, F. Keceli, M. Maiterth, and S. Jana, "Global extensible open power manager: A vehicle for hpc community collaboration on co-designed energy management solutions," in *ISC*. Cham: Springer International Publishing, 2017, pp. 394–412.

[5] B. Rountree, D. K. Lowenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. K. Bletsch, "Adagio: making dvs practical for complex hpc applications," ser. ICS '09. New York, NY, USA: ACM, 2009, pp. 460–469. [Online]. Available: http://doi.acm.org/10.1145/1542275.1542340

[6] Y. Oleynik, M. Gerndt, J. Schuchart, P. G. Kjeldsberg, and W. E. Nagel, "Run-time exploitation of application dynamism for energy-efficient exascale computing (READEX)," in *Computational Science and Engineering (CSE), 2015 IEEE 18th International Conference on*, C. Plessl, D. El Baz, G. Cong, J. M. P. Cardoso, L. Veiga, and T. Rauber, Eds. Piscataway: IEEE, Oct 2015, pp. 347–350.

[7] J. Schuchart, M. Gerndt, P. G. Kjeldsberg, M. Lysaght, D. Horák, L. Říha, A. Gocht, M. Sourouri, M. Kumaraswamy, A. Chowdhury, M. Jahre, K. Diethelm, O. Bouizi, U. S. Mian, J. Kružík, R. Sojka, M. Beseda, V. Kannan, Z. Bendifallah, D. Hackenberg, and W. E. Nagel, "The READEX formalism for automatic tuning for energy efficiency," *Computing*, vol. 99, no. 8, pp. 727–745, 2017. [Online]. Available: https://doi.org/10.1007/s00607-016-0532-7

[8] IT4Innovations. MERIC library. URL: https://code.it4i.cz/vys0053/meric [accessed: 2018-06-25].

[9] O. Vysocky, M. Beseda, L. Riha, J. Zapletal, V. Nikl, M. Lysaght, and V. Kannan, "Evaluation of the HPC applications dynamic behavior in terms of energy consumption," in *Proceedings of the Fifth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, pp. 1–19, paper 3, 2017. doi:10.4203/ccp.111.3.

[10] IT4Innovations. READEX RADAR library. URL: https://code.it4i.cz/bes0030/readex-radar [accessed: 2018-06-25].

[11] Technische Universitat Dresden. System Taurus. URL: https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/SystemTaurus [accessed: 2018-06-25].

[12] IT4Innovations National Supercomputing Centre, IT4I. Salomon supercomputer. URL: https://docs.it4i.cz/salomon/introduction/ [accessed: 2018-06-25].

[13] T. Dong, V. Dobrev, T. Kolev, R. Rieben, S. Tomov, and J. Dongarra, "A step towards energy efficient computing: Redesigning a hydrodynamic application on cpu-gpu," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, May 2014, pp. 972–981.

[14] Mont-Blanc project. Mont-Blanc project mini-clusters. URL: http://montblanc-project.eu/prototypes [accessed: 2018-06-25].

[15] Barcelona Supercomputing Center. Power Monitoring on mini-clusters. URL: https://wiki.hca.bsc.es/dokuwiki/wiki/prototype/power_monitor [accessed: 2018-06-25].

[16] E. Calore, A. Gabbana, S. F. Schifano, and R. Tripiccione, "Software and dvfs tuning for performance and energy-efficiency on intel knl processors," *Journal of Low Power Electronics and Applications*, vol. 8, no. 2, pp. 1–11, 2018. [Online]. Available: http://www.mdpi.com/2079-9268/8/2/18

[17] R. Schoene. x86_adapt. Techniche Universitat Dresden. URL: https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/X86Adapt [accessed: 2018-06-25].

 ISBN: 978-1-61208-655-2