Multi-node advanced performance and power analysis with Paraver

Filippo MANTOVANI [a,1] and Enrico CALORE [b]
[a] *Barcelona Supercomputing Center, Spain*
[b] *INFN and University of Ferrara, Italy*

Abstract. Performance analysis tools allow application developers to identify and characterize the inefficiencies that cause performance degradation in their codes. Due to the increasing interest in the High Performance Computing (HPC) community towards energy-efficiency issues, it is of paramount importance to be able to correlate performance and power figures within the same profiling and analysis tools. For this reason, we present a preliminary performance and energy-efficiency study aimed at demonstrating how a single tool can be used to collect most of the relevant metrics. Moreover we show how the same analysis techniques are applicable on different architectures, analyzing the same HPC application running on two clusters, based respectively on Intel Haswell and Arm Cortex-A57 CPUs.

Keywords. Performance, Power, Energy, Analysis, Paraver, Cluster

1. Introduction and related works

Performance analysis tools allow application developers to identify and characterize the inefficiencies that cause performance degradation in their codes. Profiling and analysis is often the first step towards the optimization of an application. Also, being able to observe and measure the behavior of parallel applications *at scale*, or at least in a multi-node context, can show unexpected pitfalls and insightful information about performance bottlenecks and/or opportunities for performance and energy-efficiency improvements.

The problem of improving energy efficiency of HPC technology has become increasingly relevant in recent years [1] and it is now listed as one of the hardest challenges for exa-scale systems, e.g., in the report about top10 systems by the US – DoE [2]. The HPC community has therefore grown a strong interest towards integrating power and energy aspects into application analysis, allowing developers to not only optimize their codes for performance, but also to investigate their energy-efficiency.

Several tools were developed to target this need. Surveys studies like the one of S. Benedict [3] already provides an overview of the tools available on the market, but we want to complement it with the flexible and visual approach allowed by the Barcelona Supercomputing Center (BSC) tools[2]. The *Extrae* instrumentation library collects per-

[1]Corresponding Author: Filippo Mantovani, Barcelona Supercomputing Center; C. Jordi Girona, 29 – 08034 Barcelona (Spain); E-mail: filippo.mantovani@bsc.es
[2]https://tools.bsc.es

formance traces of parallel applications with minimal overhead, while *Paraver* [4], a postmortem advanced trace visualizer, allows to inspect Extrae traces, enabling several kind of advanced visual and numerical analysis on the collected metrics. In this work we aim to show how Extrae and Paraver can be used to perform performance and power/energy analysis on generic applications, both on architectures where hardware counters are available [5] and integrated in the PAPI library [6], and also where external sensors and power meters have to be used. In agreement with the ideas proposed by C. Bekas et al. [7] we present in our work how to easily access metrics such FLOPS/Watt, but also how to derive metrics such as *energy to solution* and *energy delay product*.

Another promising approach towards efficiency in HPC is the one of the EEHPCWG of N. Bates [8], trying to push an awareness action at data center level. It is part of this effort the approach of optimizing the job scheduling using different power-aware policies presented by D. Tafani et al. [9] or adding hardware/software extensions for improving energy awareness as presented by W. A. Ahmad et al. [10].

Various attempts to take advantage of mobile technology for increasing energy efficiency of HPC systems have been taken in the recent past. The closest to our work are the EU Mont-Blanc project [11,12] and the COSA project [13], but several other examples can be found in the literature [14,15,16,17].

Our work is complementary to all these efforts, as we aim to have an ecosystem of tools, allowing to analyze performance and power/energy related metrics of large scale applications while running on both, classical high-end HPC clusters, as well as innovative and experimental setups. Such tools target the HPC application developers more than the data-center engineers, rising an “energy awareness” in application experts, and making easier the comparison of different architectures.

This document is organized as follows: in Section 2 we introduce the problem we want to tackle, in Section 3 we explain the hardware and software configuration in which we performed our experiments, in Section 4 we mention the benchmarking application used then in Section 5 to perform the actual performance and power measurements. Section 6 collects our final comments and future research steps.

2. Problem analysis

The sensitivity of High-Performance Computing (HPC) scientific community towards energy efficiency has grown more and more in the last years. The number of Google Scholar hits for the key words “HPC energy efficiency” for year 2016 is roughly the same as for the triennium 2013–2015. This is just a coarse metric that confirms the urgency of making HPC systems more energy efficient. As we believe that part of this efficiency can be obtained by fine tuning codes and system configurations, we focus in this work in addressing the question: *How can we make parallel application developers and scientists, closer to concepts like compute efficiency and power consumption?*

As first corollary of this broad question, we address in this work the problem of studying in a graphical manner both performance and power figures on different cluster configurations, based on the most relevant modern architectures, such as Intel and Arm.

As a second corollary of the general question, while is relatively easy to have overall figures of performance and power (e.g., total number of floating point operation executed, total execution time, average power consumption, etc.), being able to analyze portions of the execution on multi-node cluster is not trivial. We believe that this last approach is relevant, since knowing and overcoming performance inefficiency at microscopic level can lead to performance improvements and/or power optimizations. As an example, the identification of memory-bound phases, or in general phases in which not all the computational resources could be exploited, would allow to estimate the effectiveness of techniques such as using Dynamic Voltage and Frequency Scaling (DVFS) to lower the processor frequency to save energy.

3. Proposed methodology

To address the first corollary presented in Sec. 2, we consider in this work two HPC clusters based on different architectures. An high-end HPC cluster based on Intel CPUs and NVIDIA GPUs and a low-power cluster, made of NVIDIA Jetson TX boards.

The High-end HPC cluster [3] comprises 5 computing nodes, where every node embeds 2× Intel Xeon E5-2630v3 CPUs and 8× NVIDIA K80 dual-GPU boards. Each board contains 2× GK210 GPUs, accounting for 16 CUDA devices per node. This cluster is named Computing On Kepler Architecture (*COKA*) and it is managed by INFN & University of Ferrara (Italy).

The Embedded cluster [4] composes 15 nodes, each of them housing a NVIDIA Tegra X1[5] SoC, embedding a Quad Arm Cortex-A57, with 2 MB of L2 cache and 4 GB LPDDR4, supported by a 16 GB eMMC storage device and accelerated with an embedded GPU NVIDIA Maxwell, 256 CUDA cores. Node interconnection is performed using a single Gigabit Ethernet link per node. The cluster is installed at the Barcelona Supercomputing Center (Spain) and we refer to it as *Jetson* cluster in the rest of the text.

In this first work we just analyze applications running on the CPUs of these systems, and thus CPU related metrics, but GPUs could also be taken into account and we do not see technical limitations in following the same approach also on those architectures. For power measurements we rely on RAPL energy counters, accessed via PAPI library [18], for the COKA cluster and on an embedded power meter for the nodes of the Jetson Cluster.

To address the second corollary of our problem, we selected the performance instrumentation tool, Extrae, and the visual performance analyzer, Paraver [4]. Extrae is a tool which uses different interposition mechanisms to inject probes into a generic target application in order to collect performance metrics. This tool make extensive use of the PAPI interface to collect information regarding the microprocessor performance, allowing to capture such information when parallel programming calls happen, but also at the entry and exit points of instrumented user routines. Extrae is the package devoted to generate Paraver trace-files. Paraver, on the other side, is a visualization tool allowing to have a qualitative global perception of the behavior of an application previously run acquiring Extrae traces.

[3]http://coka.unife.it
[4]https://wiki.hca.bsc.es/dokuwiki/wiki:prototype
[5]http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html

4. Benchmarking Application

As a benchmarking application, representative of a wider class of lattice based HPC applications, we adopt a Lattice Boltzmann simulation which has been highly optimized for several architectures.

Lattice Boltzmann methods (LB) are widely used in computational fluid dynamics, to describe flows in two and three dimensions. LB methods [19] – discrete in position and momentum spaces – are based on the synthetic dynamics of *populations* sitting at the sites of a discrete lattice. At each time step, populations *propagate* from lattice-site to lattice-site and then incoming populations *collide* among one another, that is, they mix and their values change accordingly. LB models in n dimensions with p populations are labeled as $DnQp$ and in this work we consider a state-of-the-art $D2Q37$ model that correctly reproduces the thermo-hydrodynamical evolution of a fluid in two dimensions, and enforces the equation of state of a perfect gas ($p = \rho T$) [20,21]; this model has been extensively used for large scale simulations of convective turbulence (e.g., [22,23]).

A Lattice Boltzmann simulation starts with an initial assignment of the populations and then iterates for each point in the domain, and for as many time-steps as needed, two critical kernel functions. The first kernel, called *propagate*, moves populations across lattice sites according to an appropriate stencil depending on the LB model used. It performs only a large number of sparse memory accesses, and for this reason is strongly memory-bound. The latter, called *collide*, uses as input the populations gathered by the *propagate* kernel, and performs all the mathematical steps associated to the computation of the new population values. This function is strongly compute-bound making heavy use of the floating-point units of the processor. These two kernels take most of the execution time of any LB simulation.

In the last years several implementations of this model were developed, which were used both for convective turbulence studies [22,23], as well as a benchmarking application for programming models and HPC hardware architectures [24,25,26,27]. In this work we utilize two different implementations – of the same model – targeting the two different CPU architectures embedded respectively in the high-end and the embedded clusters. Consequently, one developed for Intel CPUs [28] and the other, derived from the former, initially ported to Armv7 architecture [29] and recently also to Armv8.

To fully exploit the high level of parallelism made available by the LB model, both implementations exploits MPI (Message Passing Interface) to divide computations across several processes and OpenMP to further divide them across threads. Moreover, to exploit CPU vector units, they both use, respectively, AVX2 and NEON *intrinsics*.

In particular, for all the tests presented in this work, we simulate a 2-dimensional fluid described by a lattice of 6144×8192 sites. Each of the N_p MPI processes handle a partition of the lattice of size $6144/N_p \times 8192$ and further divides it across N_t OpenMP threads, which therefore on their turn will handle a sub-lattice of size $\frac{6144/N_p}{N_t} \times 8192$. Each MPI process is bind to a CPU and each OpenMP thread to a core. From the physical simulation point of view, MPI processes are logically arranged in a ring, thus simulating a 2-dimensional fluid shaped as the surface of a cylinder. Consequently, data exchange happens only between neighboring processes.

As already mentioned, the sub-lattice handled by each process is further divided along the x-dimension across the spawned N_t OpenMP threads. The two threads taking care of the leftmost and rightmost part of the sub-lattice (i.e., the first and the last) for each process, initiate the MPI communications with the left and right neighbors. Moreover, relieving these two threads from part of the propagate computation duties, while performing MPI transfers, allow to overlap MPI communications with computations.

5. Application analysis with Extrae and Paraver

On the high-end HPC cluster, COKA, the use of SLURM Energy Accounting Plugin[6] already allowed to gather overall energy figures for the jobs scheduled on the cluster and custom tools were already developed to instrument generic codes on this system [30]. Despite of this, these tools did not allow to easily correlate these figures with other performance metrics and/or required the manual instrumentation of applications. Then, we installed Extrae v3.5.0rc4 instrumentation tool and recompiled it with support for OpenMPI 1.10.6, in order to allow Extrae to add its probes whenever an MPI call is invoked or OpenMP regions are encountered, by a generic application.

On the embedded cluster, Jetson, we developed a set of prolog and epilog scripts that start/stop a power monitor daemon running with minimum overhead (measured below 2%) on one of the core. The daemon simply configure, read and write registers via I2C protocol into the Texas Instruments INA3221 device[7]. The device is configured for monitoring three channels: *i)* the power drain of the CPU cores, *ii)* the power drain of the embedded GPU, and *iii)* the overall power drain of the Jetson compute module, including memories, but excluding the I/O devices on the carrier board of the development kit. It is important to note that the final result of this configuration mimics the behavior of the SLURM Energy Accounting plugin, but also generates a Paraver trace with the power data gathered while the job was running. Also, with the due changes in the backend handling the power measurements, the same SLURM infrastructure has been extended in order to support other Arm based mini-clusters[8] installed at BSC.

5.1. Metrics visualization and analysis with Paraver

After gathering performance and power traces on both clusters, using Extrae, we have been able to plot them and navigate them in the Paraver visualizer.

Figure 1 shows the instantaneous power consumption of one execution of ten iterations of the LB application introduced in Sec. 4: on the top the power drain derived from the RAPL energy counters on COKA, while on the bottom the power drain as measured by the INA3221 device on Jetson. As on the COKA cluster we run on a single dual socket node, we see a timeline with two color-coded measurements: one for socket 1 and the other for socket 2. We see values exceeding the 85W TDP of these CPUs since we are summing CPU and DRAM power drain. On the Jetson cluster we run on 12 nodes, so we can see twelve color-coded lines, encoding the power consumption of each of the nodes.

For both the architectures we clearly see an initial phase of initialization on the left, the ten iterations with visible different power drains for the alternating *propagate* and *collide* functions, and eventually on the right the final results check phase.

[6]`https://slurm.schedmd.com/acct_gather_energy_plugins.html`
[7]`http://www.ti.com/product/INA3221`
[8]`https://wiki.hca.bsc.es/dokuwiki/wiki:prototype:power_monitor`

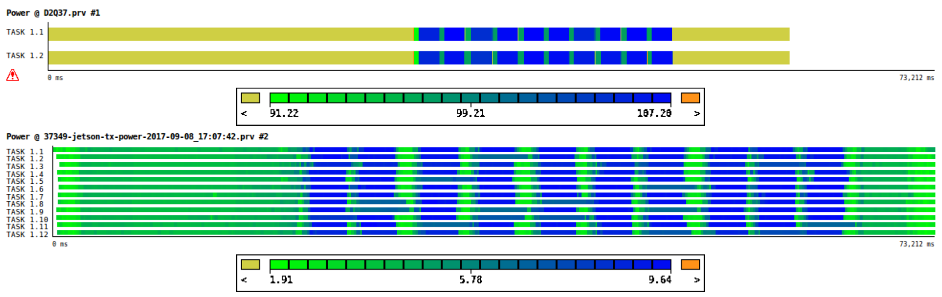


Figure 1. Color-coded timelines of the power drain (in Watts) on one node with two sockets of the COKA cluster (top) and 12 nodes of the Jeston cluster (bottom).

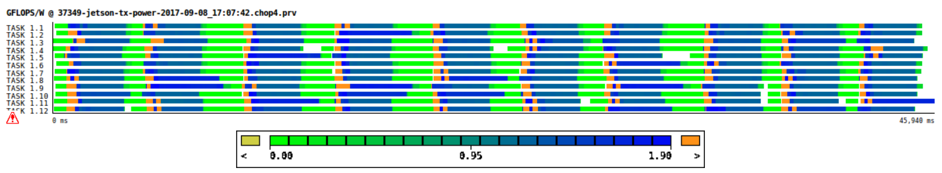
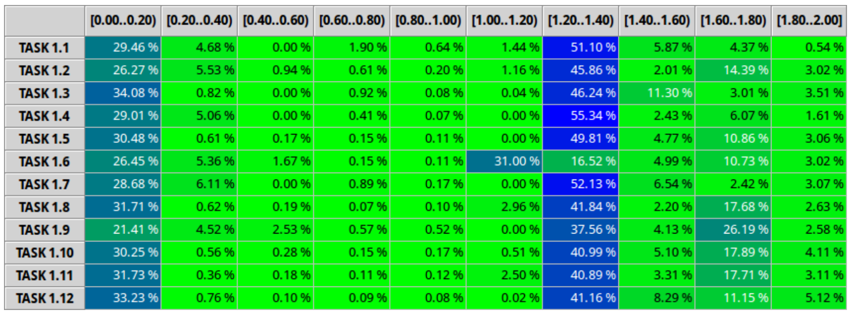


Figure 2. Timeline of the efficiency (GFLOPS/W) of the Jetson cluster.

The flexibility of the Paraver analyzer allow us to combine performance and power figures. As an example of this, we show in Figure 2 the timeline of power efficiency, i.e., GFLOPS/W, of the parallel region within the same run as before, for the Jetson cluster. This figure has been obtained simply plotting the timeline of the PAPI event accounting for SIMD operations and dividing it by the power timeline of Figure 1. Another feature



	[0.00..0.20)	[0.20..0.40)	[0.40..0.60)	[0.60..0.80)	[0.80..1.00)	[1.00..1.20)	[1.20..1.40)	[1.40..1.60)	[1.60..1.80)	[1.80..2.00]
TASK 1.1	29.46 %	4.68 %	0.00 %	1.90 %	0.64 %	1.44 %	51.10 %	5.87 %	4.37 %	0.54 %
TASK 1.2	26.27 %	5.53 %	0.94 %	0.61 %	0.20 %	1.16 %	45.86 %	2.01 %	14.39 %	3.02 %
TASK 1.3	34.08 %	0.82 %	0.00 %	0.92 %	0.08 %	0.04 %	46.24 %	11.30 %	3.01 %	3.51 %
TASK 1.4	29.01 %	5.06 %	0.00 %	0.41 %	0.07 %	0.00 %	55.34 %	2.43 %	6.07 %	1.61 %
TASK 1.5	30.48 %	0.61 %	0.17 %	0.15 %	0.11 %	0.00 %	49.81 %	4.77 %	10.86 %	3.06 %
TASK 1.6	26.45 %	5.36 %	1.67 %	0.15 %	0.11 %	31.00 %	16.52 %	4.99 %	10.73 %	3.02 %
TASK 1.7	28.68 %	6.11 %	0.00 %	0.89 %	0.17 %	0.00 %	52.13 %	6.54 %	2.42 %	3.07 %
TASK 1.8	31.71 %	0.62 %	0.19 %	0.07 %	0.10 %	2.96 %	41.84 %	2.20 %	17.68 %	2.63 %
TASK 1.9	21.41 %	4.52 %	2.53 %	0.57 %	0.52 %	0.00 %	37.56 %	4.13 %	26.19 %	2.58 %
TASK 1.10	30.25 %	0.56 %	0.28 %	0.15 %	0.17 %	0.51 %	40.99 %	5.10 %	17.89 %	4.11 %
TASK 1.11	31.73 %	0.36 %	0.18 %	0.11 %	0.12 %	2.50 %	40.89 %	3.31 %	17.71 %	3.11 %
TASK 1.12	33.23 %	0.76 %	0.10 %	0.09 %	0.08 %	0.02 %	41.16 %	8.29 %	11.15 %	5.12 %

Figure 3. Histogram of the percentage of execution time spent in each interval of efficiencies

of the Paraver tool is the possibility to generate histograms: in Figure 3 we can see the histogram of the power efficiency. On the x-axis we have bins of power efficiencies (in GFLOPS/W), while on the y-axes we show each of the twelve nodes. The light green color translate to a low percentage of execution time spent at that efficiency, while a dark blue translate to an high percentage of execution time spent in the given range of efficiency. We can clearly see two tendencies: one around 0.1 GFLOPS/W, corresponding to the *propagate* phase of the code (which is a memory-bound phase), and one around 1.3 GFLOPS/W, corresponding to the *collide*. As a comparison, on the COKA cluster the *collide* reach ~ 1.1 GFLOP/W.

Table 1. *Time to solution* (T_S), average power drain, *Energy to solution* (E_S) and *Energy Delay Product* EDP, of the LB application run respectively on 12 nodes of the Jetson cluster and on one node of COKA. We are here ignoring the power drain given by motherboards, power supply inefficiencies, network devices, cooling, etc.

	T_S [s/iter]	P_{avg} [W]	E_S [J/iter]	EDP [$J \cdot s$/iter]
12× Jetson Nodes	4.70	89.92	417.41	1961.83
1× COKA Node	2.13	206.78	439.97	937.14

As mentioned by Curioni et al. in [7], metrics like GFLOPS/W, even if relevant for worldwide ranking like Green500, are not a sufficient indicators of global efficiency of a complex HPC application running on a large HPC cluster. For this reason we show in Tab. 1 that also the *energy to solution* and *energy delay product* can be computed taking advantage of the Paraver analyzer, highlighting the interesting possibility to compare different architectures. In particular, as a preliminary result, we can appreciate from Tab. 1 that 12 Jetson nodes are equivalent to a COKA node from the *Energy to solution* point of view, although the former are a factor of 2 less power hungry, while the latter is a factor of 2 faster.

Apart from visualizing and analyzing instantaneous and global metrics, Paraver could be used also to visually correlate changes in power related metrics and the different application phases.

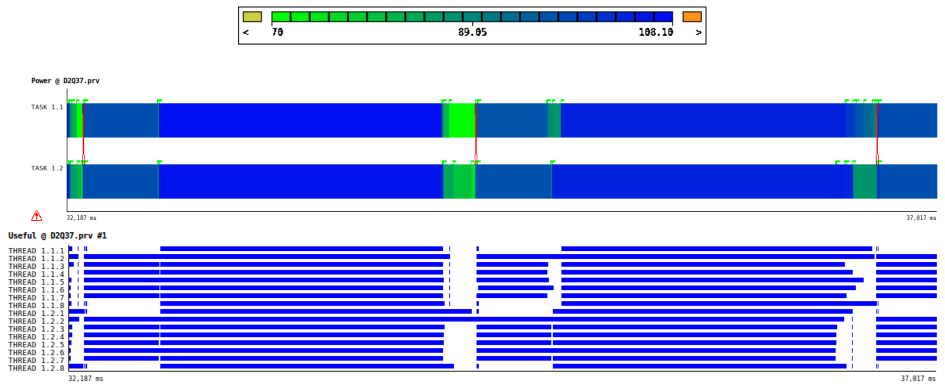


Figure 4. Two iterations of the LB simulation on one COKA node. In the upper part, using a color-code spanning between 70 and 110 Watt, we see the power drain of the two CPUs (plus DRAMs contribution), while in the bottom part, in solid blue, we see the corresponding OpenMP threads executing in the respective CPU cores.

As an example, in Figure 4 we show just 2 iterations of the LB simulation, where we plot the power drain of the two CPUs plus DRAMs contribution (using a color-code spanning between 70 and 110 Watt), on top of a view of the 16 OpenMP threads executing. This view gives a good perception of where the major computational phases are, and their balance across cores. Looking at the bottom plot, we can spot a non negligible amount of time (white parts) of about 150ms, where most of the threads are waiting (for synchronizations or communications). Interestingly, in correlation with these phases, we can see in the upper plot a lower power drain (green areas correspond to $\sim$ 70 Watt). We can also appreciate the fact that during the *propagate* phase the average power drain is $\sim$ 97 Watt, while during *collide* is $\sim$ 107 Watt.

In Figure 5 we show a similar case, but in the Jetson cluster: on the left we can see the *Instruction Per Clock* cycle, IPC, of one iteration and on the right the corresponding power drain.

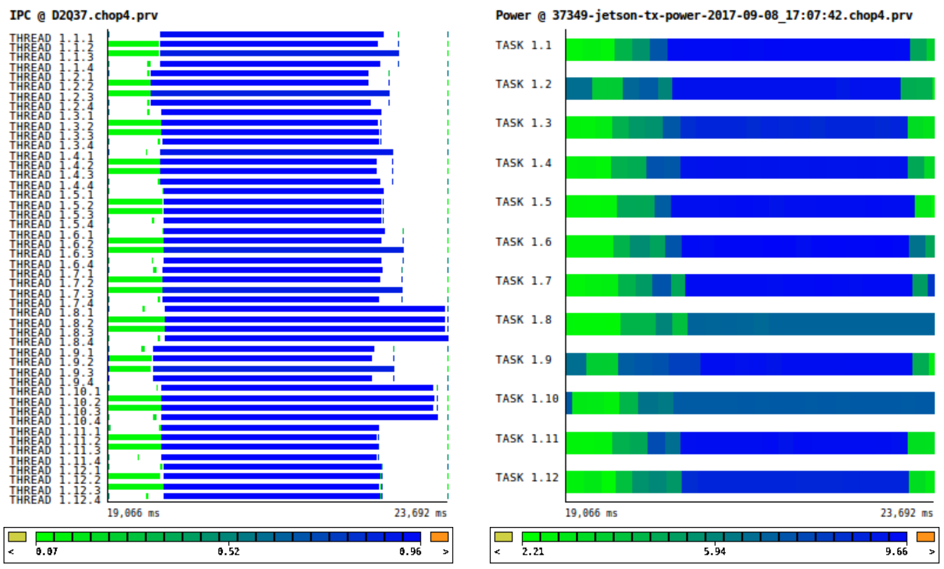


Figure 5. Timeline of one iteration of the LB code. On the left we show the *Instruction per Clock cycle* (IPC), low IPC corresponds to the *propagate* phase, i.e., the memory-bound phase. On the right the power drain during the same iteration. It is important to note that during the memory bound phase the power consumption is lower (light green) than during the compute bound phase (dark blue).

For both cases, we want to highlight here that an easy visual inspection can highlight portion of the code where optimizations could be applied. All these are useful hints, not only for possible performance optimizations, but also from the energy-efficiency point of view. In [26] we show in fact that a careful selection of the CPU frequency can reduce the overall energy consumption of the same code by $\sim$ 10% on the CPUs of the COKA cluster.

5.2. Limitations

We know the presented methodology still have some limitations such as: i) we can not be sure about the fact that power metrics derived by RAPL counters on Intel CPUs can be directly compared with the ones acquired on Jetson boards; ii) All the power figures presented in this work only take into account CPU and memory, so they do not include power for network, storage and other passive/active components in the cluster, nor the cooling; iii) synchronization of power and performance traces in the Jetson setup has still to be tuned. This can imply some variability, that can induce limitations when investigating very small time windows. Thus Tab. 1 in particular should be interpreted in view of these caveats.

6. Conclusions and future work

As further applications of this methodology, we plan to use Paraver to spot code regions where computing resources are underutilized to apply energy-efficiency optimizations. Knowing the overhead for changing CPU frequency with DVFS, we can consider to isolate the cases where lowering the frequency can result in an overall benefit in the *energy to solution*. Moreover, we plan to combine our analysis with compilation techniques such as the one presented in [31]. Eventually, as both the platforms considered for this preliminary study includes GPUs, we plan to extend our study including also GPUs.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] and Horizon 2020 under the Mont-Blanc projects [12], grant agreements n. 288777, 610402 and 671697. E.C. was partially founded by "Contributo 5 per mille assegnato all'Università degli Studi di Ferrara - dichiarazione dei redditi dell'anno 2014".

References

[1] Feng, W.c., Cameron, K.: The green500 list: Encouraging sustainable supercomputing. Computer 40(12) (2007)

[2] Lucas, R., Ang, J., Bergman, K., Borkar, S., Carlson, W., Carrington, L., Chiu, G., Colwell, R., Dally, W., Dongarra, J., et al.: Top ten exascale research challenges. DOE ASCAC subcommittee report pp. 1–86 (2014)

[3] Benedict, S.: Energy-aware performance analysis methodologies for hpc architecturesan exploratory study. Journal of Network and Computer Applications 35(6), 1709–1719 (2012)

[4] Pillet, V., Labarta, J., Cortes, T., Girona, S.: Paraver: A tool to visualize and analyze parallel code. In: Proceedings of WoTUG-18: transputer and occam developments. vol. 44, pp. 17–31 (1995)

[5] Servat, H., Llort, G., Giménez, J., Labarta, J.: Detailed and simultaneous power and performance analysis. Concurrency and Computation: Practice and Experience 28(2), 252–273 (2016)

[6] Dongarra, J., at al.: Using PAPI for hardware performance monitoring on linux systems. In: Conference on Linux Clusters: The HPC Revolution. vol. 5. Linux Clusters Institute (2001)

[7] Bekas, C., Curioni, A.: A new energy aware performance metric. Computer Science-Research and Development 25(3), 187–195 (2010)

[8] Scogland, T.R., Steffen, C.P., Wilde, T., Parent, F., Coghlan, S., Bates, N., Feng, W.c., Strohmaier, E.: A power-measurement methodology for large-scale, high-performance computing. In: Proceedings of the 5th ACM/SPEC international conference on Performance engineering. pp. 149–159. ACM (2014)

[9] Rajagopal, D., Tafani, D., Georgiou, Y., Glesser, D., Ott, M.: A novel approach for job scheduling optimizations under power cap for arm and intel hpc systems. In: Proceedings of 24th IEEE International Conference on High Performance Computing, Data, and Analytics (HiPC 2017) – in press (2017)

[10] Ahmad, W.A., et al.: Design of an energy aware petaflops class high performance cluster based on power architecture. In: Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International. pp. 964–973. IEEE (2017)

[11] Rajovic, N., Carpenter, P., Gelado, I., Puzovic, N., Ramirez, A., Valero, M.: Supercomputing with commodity CPUs: Are mobile SoCs ready for HPC? In: High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for. pp. 1–12 (Nov 2013)

[12] Rajovic, N., et al.: The Mont-blanc Prototype: An Alternative Approach for HPC Systems. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 38:1–38:12. SC '16, IEEE Press, Piscataway, NJ, USA (2016)

[13] Cesini, D., Corni, E., Falabella, A., Ferraro, A., Morganti, L., Calore, E., Schifano, S., Michelotto, M., Alfieri, R., De Pietri, R., Boccali, T., Biagioni, A., Lo Cicero, F., Lonardo, A., Martinelli, M., Paolucci,

P., Pastorelli, E., Vicini, P.: Power-efficient computing: Experiences from the COSA project. Scientific Programming (2017), doi:10.1155/2017/7206595

[14] Nikolskiy, V.P., Stegailov, V.V., Vecher, V.S.: Efficiency of the Tegra K1 and X1 systems-on-chip for classical molecular dynamics. In: 2016 International Conference on High Performance Computing Simulation (HPCS). pp. 682–689 (July 2016), doi:10.1109/HPCSim.2016.7568401

[15] Ukidave, Y., Kaeli, D., Gupta, U., Keville, K.: Performance of the nvidia jetson tk1 in hpc. In: Cluster Computing (CLUSTER), 2015 IEEE International Conference on. pp. 533–534. IEEE (2015)

[16] Geveler, M., Ribbrock, D., Donner, D., Ruelmann, H., Höppke, C., Schneider, D., Tomaschewski, D., Turek, S.: The ICARUS White Paper: A Scalable, Energy-Efficient, Solar-Powered HPC Center Based on Low Power GPUs, pp. 737–749. Springer International Publishing (2017), doi:10.1007/978-3-319-58943-5_59

[17] Durand, Y., Carpenter, P.M., Adami, S., Bilas, A., Dutoit, D., Farcy, A., Gaydadjiev, G., Goodacre, J., Katevenis, M., Marazakis, M., et al.: Euroserver: Energy efficient node for european micro-servers. In: Digital System Design (DSD), 2014 17th Euromicro Conference on. pp. 206–213. IEEE (2014)

[18] Weaver, V., Johnson, M., Kasichayanula, K., Ralph, J., Luszczek, P., Terpstra, D., Moore, S.: Measuring energy and power with PAPI. In: Parallel Processing Workshops (ICPPW), 2012 41st International Conference on. pp. 262–268 (2012), doi:10.1109/ICPPW.2012.39

[19] Succi, S.: The Lattice-Boltzmann Equation. Oxford university press, Oxford (2001)

[20] Sbragaglia, M., Benzi, R., Biferale, L., Chen, H., Shan, X., Succi, S.: Lattice Boltzmann method with self-consistent thermo-hydrodynamic equilibria. Journal of Fluid Mechanics 628, 299–309 (2009), doi:10.1017/S002211200900665X

[21] Scagliarini, A., Biferale, L., Sbragaglia, M., Sugiyama, K., Toschi, F.: Lattice Boltzmann methods for thermal flows: Continuum limit and applications to compressible Rayleigh–Taylor systems. Physics of Fluids (1994-present) 22(5), 055101 (2010), doi:10.1063/1.3392774

[22] Biferale, L., Mantovani, F., Sbragaglia, M., Scagliarini, A., Toschi, F., Tripiccione, R.: Second-order closure in stratified turbulence: Simulations and modeling of bulk and entrainment regions. Physical Review E 84(1), 016305 (2011), doi:10.1103/PhysRevE.84.016305

[23] Biferale, L., Mantovani, F., Sbragaglia, M., Scagliarini, A., Toschi, F., Tripiccione, R.: Reactive Rayleigh-Taylor systems: Front propagation and non-stationarity. EPL 94(5), 54004 (2011), doi:10.1209/0295-5075/94/54004

[24] Biferale, L., Mantovani, F., Pivanti, M., Pozzati, F., Sbragaglia, M., Scagliarini, A., Schifano, S.F., Toschi, F., Tripiccione, R.: A Multi-GPU implementation of a D2Q37 lattice boltzmann code. In: Parallel Processing and Applied Mathematics: 9th International Conference, PPAM 2011, pp. 640–650. Lecture Notes in Computer Science (2012), doi:10.1007/978-3-642-31464-3_65

[25] Calore, E., Schifano, S.F., Tripiccione, R.: On portability, performance and scalability of an MPI OpenCL lattice boltzmann code. In: Euro-Par 2014: Parallel Processing Workshops, pp. 438–449. LNCS, Springer (2014), doi:10.1007/978-3-319-14313-2_37

[26] Calore, E., Gabbana, A., Kraus, J., Schifano, S.F., Tripiccione, R.: Performance and portability of accelerated lattice Boltzmann applications with OpenACC. Concurrency and Computation: Practice and Experience 28(12), 3485–3502 (2016), doi:10.1002/cpe.3862

[27] Calore, E., Gabbana, A., Kraus, J., Pellegrini, E., Schifano, S.F., Tripiccione, R.: Massively parallel latticeboltzmann codes on large GPU clusters. Parallel Computing 58, 1 – 24 (2016), doi:10.1016/j.parco.2016.08.005

[28] Mantovani, F., Pivanti, M., Schifano, S.F., Tripiccione, R.: Performance issues on many-core processors: A D2Q37 lattice boltzmann scheme as a test-case. Computers & Fluids 88, 743 – 752 (2013), doi:10.1016/j.compfluid.2013.05.014

[29] Calore, E., Schifano, S.F., Tripiccione, R.: Energy-performance tradeoffs for HPC applications on low power processors. In: Euro-Par 2015: Parallel Processing Workshops: Euro-Par 2015 International Workshops, Vienna, Austria, August 24-25, 2015, Revised Selected Papers. pp. 737–748. LNCS, Springer (2015), doi:10.1007/978-3-319-27308-2_59

[30] Calore, E., Gabbana, A., Schifano, S.F., Tripiccione, R.: Evaluation of dvfs techniques on modern hpc processors and accelerators for energy-aware applications. Concurrency Computation: Practice and Experience (2017), doi:10.1002/cpe.4143

[31] Tran, K.A., Carlson, T.E., Koukos, K., Själander, M., Spiliopoulos, V., Kaxiras, S., Jimborean, A.: Clairvoyance: look-ahead compile-time scheduling. In: Proceedings of the 2017 International Symposium on Code Generation and Optimization. pp. 171–184. IEEE Press (2017)