Experimental Workflow for Energy and Temperature Profiling on HPC Systems

Kameswar Rao Vaddina, Laurent Lefèvre, Anne-Cécile Orgerie

Experimental Workflow for Energy and Temperature Profiling on HPC Systems

Kameswar Rao Vaddina
Univ. Rennes, Inria, CNRS, IRISA
Rennes, France
kvaddina@inria.fr

Laurent Lefèvre
Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP
Lyon, France
laurent.lefevre@inria.fr

Anne-Cécile Orgerie
Univ. Rennes, Inria, CNRS, IRISA
Rennes, France
anne-cecile.orgerie@irisa.fr

***Abstract*—Despite recent advances in improving the performance of high performance computing (HPC) and distributed systems, power dissipation and thermal cooling challenges persist, impacting their total cost of ownership. Making HPC systems more energy and thermal efficient will require understanding of individual power dissipation and temperature contributions of multiple hardware system components and their accompanying software. In this work, we present an experimental workflow for energy and temperature profiling on systems running parallel applications. It allows full and dynamic control over the execution of applications for the entire frequency range. Through its use, we show that the energy response to frequency scaling is highly dependent on the workload characteristics and it is convex in nature with an optimal frequency point. During the course of our experimentation, we encountered a non-intuitive finding, where we observed that the tested low-power processor is consuming more power on average than the standard processor.**

***Index Terms*—Energy profiling, power measurement, temperature profiling, monitoring infrastructure, HPC systems**

I. Introduction

As the workload running on HPC systems is highly variable, power demand becomes unpredictable [6]. This leads to planning for the worst case situation, and consequently to over-budgeting. The unexpected power peaks caused by varied system utilization combined with limited power and cooling capacity poses significant power management challenges. To address these challenges, we can limit the power consumed by system components by dynamically adjusting and redistributing power among server components. At the same time, as the process node shrinks, the on-chip power density of the CPUs increases due to higher and denser transistor integration. The increase in power density contributes to higher on-chip temperatures, thus limiting the available thermal design power. This tightens the energy efficiency constraints and significantly reduces the system performance.

Over the past decade, there is a growing demand for faster floating point operations in newer application domains like data-analytics, machine learning and scientific computing. The CPU vendors are doubling down on the core count and improving core frequencies while taking advantage of the process node shrinkage and relentlessly trying to improve the energy efficiency of the processors.

The HPC platforms available in the market, have a lot of software acting in unison, trying to meet the challenging workload demands. The BIOS, operating system, device drivers and user-level software applications, all have some level of impact on the overall energy consumption of the system. A bulk of this energy is being consumed by the software applications that run on the platform. Thus, it is imperative to observe and measure the impact of software applications on power requirements, so that developers can benefit from this information and optimize their code for low energy consumption. Since, increase in temperature leads to an exponential increase in power dissipation, it is also important to have a unified approach for optimizing for both energy and temperature (energy/temperature co-optimization) [17].

The fine-grained granularity required in terms of temporal profiling (complete and individual code sections) and spatial profiling (power dissipation of dominant components like CPU, memory, etc) can be a great asset to the software developers in understanding the power profile of the system during the software development process. Therefore, having an experimental workflow and methodology which can profile for both energy and temperature is a critical aspect of any energy efficiency research project, and can provide us with very interesting insights.

Power and temperature measurement and characterization of applications on HPC systems can lead to insights that were previously not possible and is hence an interesting research direction. In this paper, we describe a novel experimental workflow for highly accurate and on-line mechanism that allows to perform energy and temperature monitoring and profiling of parallel applications running on HPC systems. This low-overhead monitoring infrastructure uses a combination of Model Specific Registers (MSR's) and a pseudo file system provided by the Linux Kernel called *sysfs*. It is devoid of any external invasive tools. It embeds a non-intrusive and low-overhead logging service which is based on *systemd*. The logging service runs in the background and only logs during the execution of the benchmark, thus providing fine-grained and precise, power and temperature values and achieving good precision and sampling rate (11-16 samples/second for 1200-1800 MHz). Our workflow is very flexible in modeling the experiment by providing control knobs which allow the user to investigate the effect of CPU off-lining on performance, power and temperature. We use simple control knobs to enable/disable standard Intel features like Hyper-threading and Turbo Boost. The internals of this workflow are described in

details to be reproducible by others. Furthermore, using this monitoring infrastructure, we confirm and re-validate the existence of the Energy-/Frequency Convexity Rule which states that the Energy-/Frequency curve exhibits a convex behavior, with an optimal frequency point where energy consumption can be minimized.

The remainder of this paper is structured as follows. In Section II, we describe the state-of-the-art in energy and temperature profiling and show how our experimental flow differs from them. In Section IV, we describe our monitoring infrastructure. The experimental setup is described in Section IV. The results obtained by running various benchmarks, our hypothesis and data analysis are discussed in detail in Section V. Finally, we conclude in Section VI and provide remarks on future work.

II. Related work

Over the past years, HPC systems have been at the forefront of new application domains like simulation based computational science, data intensive computing, machine learning etc. This tremendous amount of growth in IT infrastructure has a staggering effect on the energy usage and efficiency of HPC systems [4]. Energy-efficiency has become one of the most important metrics to aspire, for contemporary green computing and data-centers. Optimizing for energy along with temperature decreases ageing related issues and increases the longevity of the infrastructure by increasing mean time to failure (MTTF) [22]. In HPC systems, for some workloads the substantial contributor of energy consumption are the CPU's [15], whose energy consumption is highly dependent on the CPU frequency and the characteristics of the workload.

The prior works on accounting for energy on processors and server platforms for various kinds of workloads already exist. Many of them differ at what hardware IP block level the profiling is being performed, type of profiling mechanism, granularity of profiling etc. But, mostly they can be broadly classified into two different categories [14]. They are, works that directly measure for energy consumption using external gauges [21] or on-board sensors [7], and works that model energy, based on low-level activity derived from hardware performance counters [13], [20].

External gauges from National Instruments (like NI 9227 for current measurement and NI 9215 for voltage measurement) [21] and from Monsoon power meter [3] can be used to accurately measure hardware component-level or system-wide energy consumption. On-board power sensors like Intel's Running Average Power Limit (RAPL) [18] have also been used previously to get the energy consumption of individual hardware components. The accuracy of tools that model energy consumption from low-level hardware performance counter data is hardware specific and very workload dependent [19].

Previously, it has been presented in several studies that the energy consumption curve of the microprocessor with respect to clock frequency is convex in nature. Many of the approaches were from the perspective of the CPU's dynamic voltage and frequency scaling (DVFS) without considering the impact of software applications on the overall energy consumption [12]. Others have re-validated the existence of Energy-/Frequency convexity curve in the context of embedded systems and energy-critical software applications [21]. Efraim et al. [5] have also validated the existence of an optimal frequency and voltage operational point in order to achieve minimal energy consumption on Intel® Core processors. Some research works [1], [9] have discussed the convex behavior and revealed how to exploit it from the system-level point of view.

Many of the above addressed works study and analyze only the power consumption and that too in isolation. But, leakage power component of the total power dissipation increases exponentially with temperature. Also, in order to do energy/temperature accounting per user or per application, a thorough study has to be commissioned to measure these metrics while the user is logged in or while running a particular application. This study could be used as the basis for future hardware and software co-optimization of energy and temperature. Our current work does profiling of both power and temperature in unison. This allows to build an accurate analytical model for power dissipation as our future work.

We use Intel's RAPL interface which is part of their power-capping framework. Intel introduced RAPL power metering capability first in Sandy Bridge micro-architecture. It is a model based power meter with accuracy comparable to external analog power meters [18]. The RAPL's interface is via the model specific registers (MSR's), which are control registers used for performance monitoring and toggling of certain CPU features. RAPL also provides power limiting feature, which can be used to limit the power and dynamically adjust the p-states. We read the files under Sysfs powercap interface for energy profiling and at the same time use raw-access to the underlying MSR's for temperature profiling. More details about our experimental setup, results and analysis is described below.

III. Energy and Temperature Monitoring Infrastructure

Our experimental workflow is depicted on Figure 1 and detailed hereafter.

A. Tuned hardware parameters

The default $intel\_pstate$ CPU performance scaling driver only supports two CPU governors (namely *powersave* and *performance*) at the time of writing this paper. So, at boot time we pass a kernel line parameter (*intel_pstate=disable*), which disables the $intel\_pstate$ driver. This allows the test environment to boot up with the generic $acpi\_cpufreq$ scaling driver. The $acpi\_cpufreq$ scaling driver supports the *userspace* governor, which allows us to have full and dynamic control over the entire frequency range. This governor allows Linux user space to set the CPU frequency for the policy it is attached to, by writing to the $scaling\_setspeed$ attribute of that policy.

We have disabled Intel's proprietary Turbo Boost feature which might introduce some variation in the performance

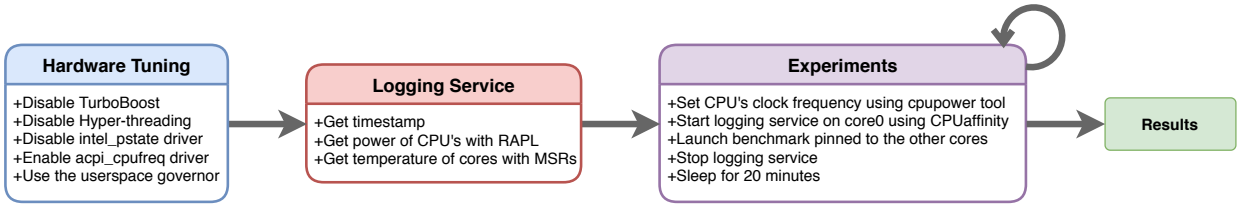

Fig. 1. Experimental workflow

between different benchmark runs [8]. After the boot-up of the operating system, we also disable Hyper-threading as it causes indeterminism in the execution of benchmark applications. Hence, all the benchmark runs in this paper uses only one logical processor per physical core.

We vary the frequencies from the base to nominal range (1200-1800MHz for E5-2630L and 1200-2200MHz for E5-2630) for each benchmark run. Initially, the main launch script changes the CPU governor from *ondemand* to *userspace* and then sets the CPU frequency with the help of a Linux tool called *cpupower*. This allows us to manually control the CPU's clock frequency. For each CPU frequency, we first start the logging service, execute the benchmark and then immediately stop the logging service. The script then sleeps for about 20 minutes before executing the benchmark for another frequency. This abundant time allows the CPU to return to steady-state idle temperature. At the end of the last frequency run, the CPU governor is reverted back to *ondemand* from *userspace*. At this point the power and temperature trace is written to the disk. Post processing of the collected trace happens offline.

B. Logging service

We are using *Systemd* - system and service manager to initialize the logging service. The logging service loads in the background immediately after the boot-up of the Linux kernel image. The *ExecStart* directive of the logging service specifies the main script which gets executed and does the logging of energy and core temperatures. We use *Systemd's* controlling interface and inspection tool called *systemctl* to start and stop the logging service. We achieve extremely fine-grained logging granularity which is devoid of unnecessary data points by starting the logging service before the execution of the benchmark and stopping the service right after the execution of the said benchmark. The *core0* of the CPU is dedicated exclusively to logging by setting the *CPUAffinity* directive to 0. We achieve asynchronous logging capability by pinning the execution of benchmarks to the rest of the physical cores. This step is crucial to avoid unwanted processes movements between cores during the execution time due to the process management policy of the Linux kernel. We have empirically observed that the very act of writing power and temperature trace to the disk, has only up to 10% to 15% of CPU utilization on *core0*. Hence, we conclude that the impact of it on the overall power and temperature is marginal and can be considered to be constant across our benchmark runs.

In the main logging script, we use the Linux Power Capping Framework which exposes power capping devices to the user space via *sysfs* file paths in the form of a tree of objects. The *intel-rapl* control type object represents Intel's "Running Average Power Limit" (RAPL) interface. It contains two power zones, *intel-rapl:0* and *intel-rapl:1*, representing CPU packages (PACKAGE_ENERGY0 and PACKAGE_ENERGY1). We also use the generic thermal *sysfs* driver to read the thermal zone package temperatures (PACKAGE_ID0 and PACKAGE_ID1). Apart from logging the package energy values, package temperatures, the logging script also logs the core temperatures. This is done by directly reading the Model Specific Registers (MSR's) for each corresponding core [11]. The package energy values, package temperatures, and the core temperatures are then time-stamped with a high-precision timer and stored for further processing. We find it easier to minimize overhead and ensure complete control over the hardware by using our own monitoring software written from the ground-up.

IV. Experimental Setup

The experiments were conducted on the Grid'5000 infrastructure, which is a large-scale and flexible test-bed for experiment-driven research. The test-bed allows experiments in a fully controllable and observable environment and supports high-quality, reproducible experiments. We used two Dell PowerEdge R630 servers from two different physical locations of the Grid'5000 infrastructure.

The two servers have different variants of Intel Xeon E5 processors. One has a Xeon E5-2630L v4 (Broadwell, 1.80GHz nominal frequency, 2 CPUs/node, 10 cores/CPU) and the other has Xeon E5-2630 v4 (Broadwell, 2.20GHz nominal frequency, 2 CPUs/node, 10 cores/CPU). The Xeon E5-2630L variant is a low power processor. The test environment is running a minimalist image of Debian10 Linux OS. The deployment of the test environment is fully automated.

The nodes within the cluster group spread across several racks are cooled with a central cooling system (CCS). The CCS leverages "Schneider Electric IN-ROW" rack system which creates an airtight environment for the nodes. It is connected to a dedicated management network that enables remote cooling and control [16]. The temperature outside the cluster group is regulated by a secondary cooling system (SCS), which is tasked to maintain a constant temperature in the server room.

We use the NAS Parallel Benchmark suite [2] (EP - Embarrassingly Parallel, and LU - Lower-Upper Gauss-Seidel solver) and the DGEMM benchmark [10] with the most recent version of the Intel C/C++ compiler (icc) and its associated Math Kernel Library (MKL library). The EP benchmark has been compiled with *CLASS=D* problem size, whereas the LU benchmark has been compiled with *CLASS=C*. The execution time of the benchmarks for other sizes we tested were either too short or too long to arrive at a consequential experiment in a reasonable amount of time.

V. Results and Analysis

We use our framework to perform an experimental analysis of the energy-temperature behavior of the two processors presented above. Since, we disabled Hyper-threading on our test servers, we have 20 cores per CPU, spread between 2 sockets. The *core0* is dedicated to asynchronous logging while simultaneously executing both benchmarks. Hence, 19 cores are available for scheduling the MPI tasks. We use *mpirun* to launch the MPI benchmarks. The EP benchmark is launched with 19 threads running on 19 cores. The LU benchmark is launched with 18 threads running on 18 cores. The inherent nature of the LU benchmark is that, it has to form a process grid. Since, 19 is a prime number, we used 18 threads running on 18 cores. Hence, we did not schedule any task on the last core (*core19*) which has been kept idle. We run 4 iterations of the benchmark runs and found no significant statistical differences in the results to be reported.

A. Power dissipation and application performance

Figure 2 shows that the average power dissipation of EP and LU benchmarks increases linearly with the increase in frequency on Broadwell (BW) and Broadwell_lowpower (BW_l) architectures. The increase is quite steep with higher slope during the last segment of the frequency (the last 100MHz segment of the frequency) for both platforms. It can be seen that the maximum average power consumption for the low-power processor (BW_l) is lower than the standard processor (BW), due to having lower maximum frequency. But, the average power dissipation for the low-power processor (BW_l) architecture is higher than the standard processor (BW). This is a counter-intuitive result when considering the fact that the package size, cache sizes (for L1i, L1d, L2 and L3) are the same for both processors.

1) Analysis of why a low-power processor consumes more than standard processor: We investigate 4 directions which could aid us in understanding why the low-power processor is consuming more power:

1) differences in the processor specifications.
2) performance and energy bias hint from the software.
3) analyzing the contribution of DRAM traffic by understanding the run times of the benchmarks.
4) identifying rarely occurring performance variations by evaluating purely CPU bound benchmarks.

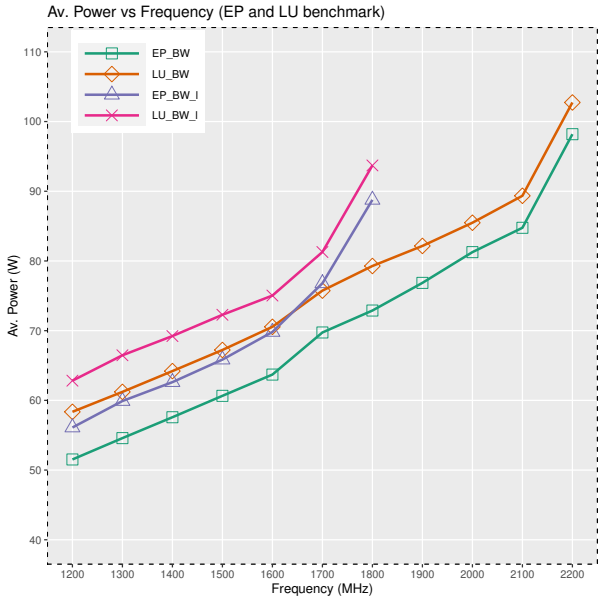

Fig. 2. Average power dissipation of EP and LU benchmarks for Broadwell (BW) and Broadwell-lowpower (BW_l) architecture.

We did not notice any specific difference when we compared the product specifications for both processors.

We further investigated, whether the benchmarks have any software hint, which can guide the hardware heuristic of power management in order to favor dynamic performance over energy consumption [11]. We did not find any such reference in the source code of the benchmarks [2]. We also checked the lowest 4-bits of IA32_ENERGY_PERF_BIAS model specific register (MSR) during the execution of the benchmarks and it was always set to 7. The MSR value can be anywhere between 0-15, where a value of 0 corresponds to the highest possible dynamic performance and a value of 15 corresponds to the maximum possible energy savings. A value of 7 can be interpreted as the processor maintains a balance between dynamic performance and energy consumption. Hence, we rule out the possibility of software hinting and changing the hardware heuristic nudging it towards more dynamic performance thereby consuming more power on average for the BW_l architecture.

Figure 3, shows the performance of the CPU's i.e., execution time with respect to the frequency of the processors for both benchmarks running on our two test platforms. It can be seen that the execution times for both benchmarks is almost the same on the test platforms (barring for the additional available frequency range of the Broadwell platform). Hence, we can rule out the possibility, that one of the CPU's might be waiting on the data from the DRAM and thus contributing to the increase in the power consumption. The execution time decreases linearly with the increase in frequency. This highlights the importance of frequency scaling (DFS) to fine-tune the performance of the system.

2) Faster memory bus: In order to understand why the low-power processor on an average consumes more power

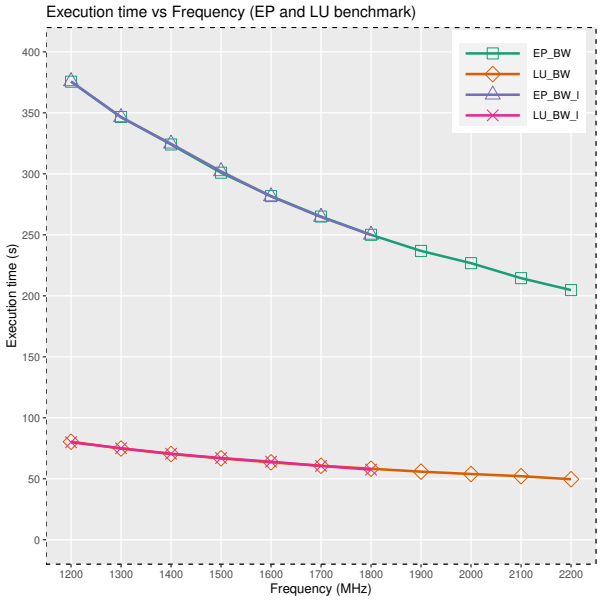

Fig. 3. Execution time of EP and LU benchmarks for Broadwell (BW) and Broadwell-lowpower (BW_l) architecture.

for the same benchmarks when compared to the standard processor, we used a CPU bound double-precision floating point general matrix multiply (DGEMM) benchmark. We used Intel® C/C++ compiler along with Intel® Math Kernel Library (Intel® MKL) to compile and build the DGEMM benchmark. The Math Kernel Library provides fast implementations for many frequently used math routines. The benchmark takes as its only parameter the problem size N. It then allocates 3 matrices of size N x N and initializes them with random data. We chose a small matrix size of 100x100 numbers which would fit in the L3 Cache (25MB shared memory per NUMA node). The script is setup to run the DGEMM benchmark 1 million times. The script uses *numactl* utility, which runs the processes with a specific scheduling and memory placement policy. We use this utility to launch the benchmark on all the available 19 cores. We found that even for DGEMM benchmark, the low-power CPU is consuming more power on average, but it is executing the benchmark faster than the standard CPU across the frequency range.

Considering the fact that the benchmark's data is small enough to fit in the last-level (L3) cache, and the fact that the benchmark gets executed faster on the low-power CPU and also that it is consuming more power on average, we conclude that the memory bus between the last-level shared cache and the CPU cores is faster and therefore consuming more power. Although, it can simply be assumed that this discrepancy is the result of the manifestation of Intel's product binning, it warrants an extensive study on caches (eviction rates) using hardware performance counters which might enable to identify the underlying mechanism for the increased power consumption of the low-power CPU. Such a detailed micro-architectural study is left as the future work.

B. *Energy consumption*

Figure 4 and Figure 5 shows the plot of energy consumption of EP and LU benchmarks with respect to frequency for Broadwell and Broadwell-lowpower CPU architectures respectively. The plots reveal that the energy-frequency curve for both benchmarks on both architectures has a convexity with an optimal frequency point (f_opt) at which the energy consumption is minimized. This optimal frequency point is at 1600 MHz for both architectures.

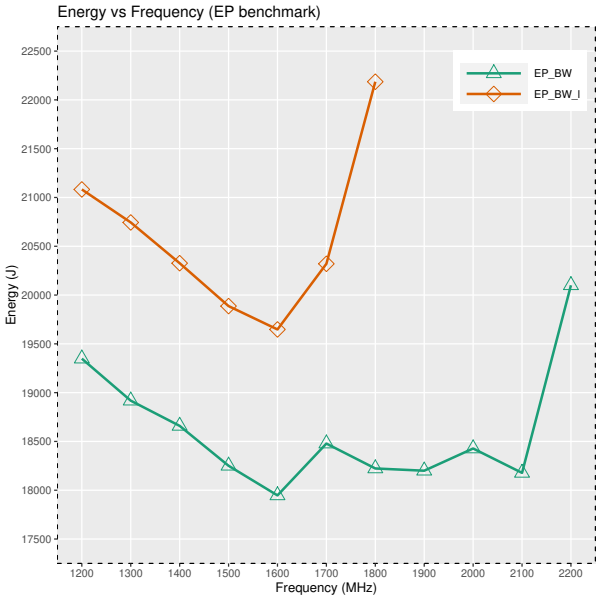

Fig. 4. Energy consumption of EP benchmark for Broadwell (BW) and Broadwell-lowpower (BW_l) architecture.

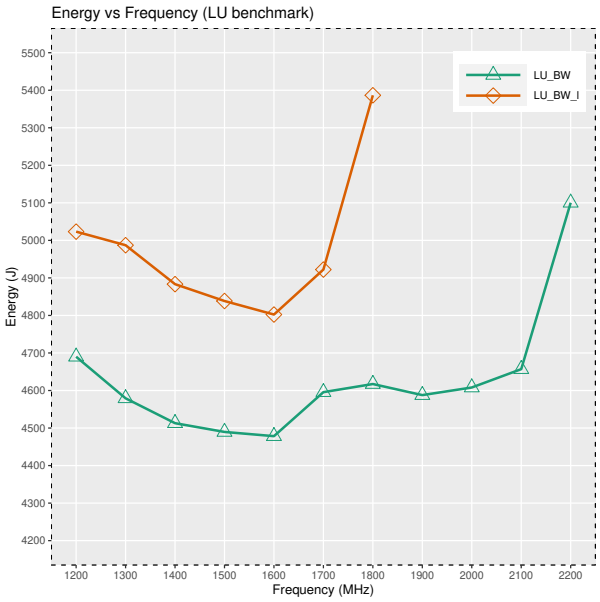

Fig. 5. Energy consumption of LU benchmark for Broadwell (BW) and Broadwell-lowpower (BW_l) architecture.

As the CPU frequency decreases, the execution times of the benchmarks increases linearly, while the leakage power continues to be drawn during the time the CPU is active. This leakage power contributes to the convexity of the energy curve on the lower end of the frequency range, i.e., when the operating frequency f is lower than the optimal frequency f_{opt}. For the region where the operating frequency is greater than the optimal frequency ($f > f_{opt}$), the energy consumption is attributed to the increase in frequency (and thus voltage), which contributes to the dynamic power.

It can also be seen that for the standard CPU which has an extended frequency range of up to 2200 MHz, the energy consumption is relatively flat (with some minor variation) after the f_{opt} point i.e., between 1600 MHz and 2100 MHz. This means, that we can trade-off small amounts of energy increases for faster execution times. In the case of EP benchmark running on BW_l architecture, we can increase the performance of the benchmark by 23.8% by increasing the clock frequency from 1600 MHz to 2100 MHz. This would marginally increase the energy consumption by about 1.28%. Similarly, we can increase the performance of LU benchmark running on LU_l architecture by 17.89%, by increasing the clock frequency from 1600 MHz to 2100 MHz. This would marginally increase the energy consumption by about 3.98%.

So, when a new benchmark enters the system and the CPU happens to be idling at the lowest possible frequency then the faster we approach the f_{opt} point and run the benchmark, the more the energy savings. Similarly, if the CPU happens to be at the maximum frequency by virtue of its energy-performance policy and a new benchmark enters the system whose policy is to conserve energy, then the faster we approach the f_{opt} point and run the benchmark, the more the energy savings.

C. CPU temperature

Figure 6 shows the average CPU core temperatures plotted with respect to frequency for EP and LU benchmarks running on our test platforms. The temperature profiles for all the cores are captured for the whole duration of the benchmark execution and for all the frequency ranges. We then average the core frequencies of all the cores (across the two sockets) and then plot the curves with respect to the frequency. The critical temperature for the BW architecture is at 60°C and 70°C respectively, whereas for the BW_l it is 80°C and 90°C. It can be seen that the average core temperatures of both benchmarks executing on the two platforms increases linearly. For both benchmarks, the average temperatures are well below their critical temperatures.

Figure 7 shows that the average package temperatures plotted with respect to frequency for EP and LU benchmarks running on our test platforms. Similar to the core temperatures, the package temperatures are also captured for the whole duration of the benchmark execution and for all the frequency ranges. We then average the package temperatures for both sockets and then plot the curves with respect to the frequency. It can be seen that the average package temperatures of both benchmarks executing on the two platforms increases linearly.

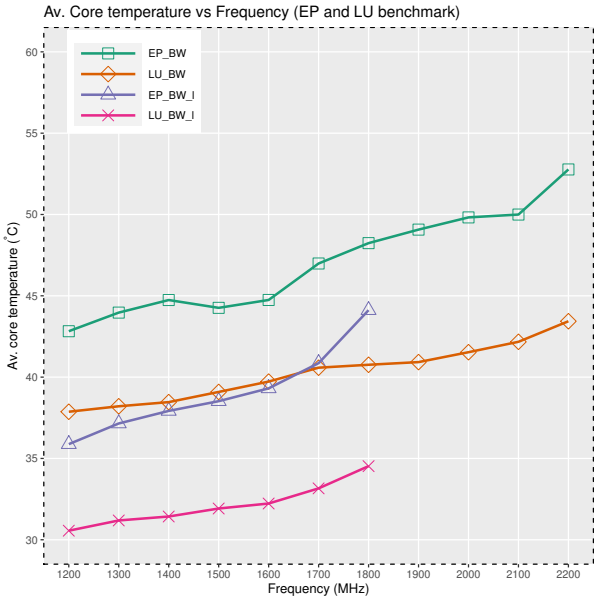

Fig. 6. Average core temperature of EP and LU benchmarks for Broadwell (BW) and Broadwell-lowpower (BW_l) architecture.

It can also be seen that the average package temperatures is higher than the average core temperatures. The Package temperatures are point temperatures provided by onboard sensors placed strategically at a known hotspot region. Even in this case, the package temperatures are well below the critical temperatures.

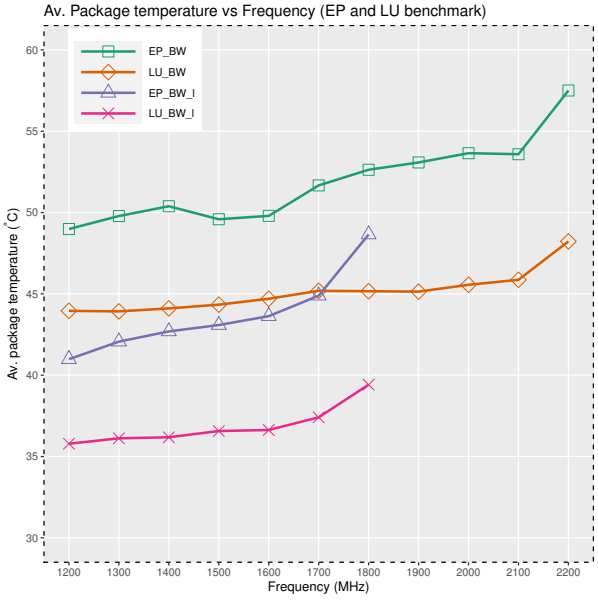

Fig. 7. Average package temperature of EP and LU benchmarks for Broadwell (BW) and Broadwell-lowpower (BW_l) architecture.

VI. Conclusions and Future Work

As HPC systems are becoming more and more constrained with thermal envelopes, optimizing parallel applications for

energy efficiency remains a critical challenge. We argue that understanding power dissipation and its link with temperature is the necessary first step towards optimizing the energy efficiency of HPC systems. In this work, we have presented a new experimental workflow for energy and temperature monitoring and profiling of parallel applications. Our experimental workflow relies on low-overhead monitoring which gives a complete and flexible control over the execution of applications for the entire frequency range. We detailed the software and hardware parts used by our workflow in order to provide precise and exhaustive insights on how to resuse it for other studies. This work is also an attempt to fill the methodological gap in the literature about reproducible experiments on energy-efficient computing architectures. Our experimental analysis reaffirms that the energy/frequency response is convex in nature with an optimal frequency point where the energy consumption is minimal. We also encountered a non-intuitive finding, where the average power dissipation of a tested low-power processor was found to be more than the standard processor. We analyzed that result in greater detail.

Our future work will focus on studying the caches and their eviction rates using hardware performance counters, in order to completely understand as to why the low-power processor is consuming more power on average than the standard processor. We also plan to build an analytical model for the energy/frequency convex curve, which would give software developers access to energy and temperature profiles during the early phase of software development cycle. In that regard, we have investigated the undocumented feature of Intel Opcodes for supply voltage monitoring and have obtained the voltage part of the frequency/voltage pair. The level of accuracy and sampling rate achieved in this work will allow in the future to do energy and temperature profiling of software applications. It will also allow to analyze the impact of smaller segments of software programs on the energy budget, thereby paving the way for fine-grained energy and temperature optimizations.

ACKNOWLEDGMENT

Experiments presented in this paper were carried out using the Grid'5000 test-bed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr). This work is supported by the Hac Specis Inria Project Lab.

REFERENCES

[1] B. Austin and N. J. Wright, "Measurement and interpretation of micro-benchmark and application energy use on the cray xc30," in *Energy Efficient Supercomputing Workshop*, 2014, pp. 51–59.

[2] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber *et al.*, "The nas parallel benchmarks," *The International Journal of Supercomputing Applications*, vol. 5, no. 3, pp. 63–73, 1991.

[3] N. Brouwers, M. Zuniga, and K. Langendoen, "NEAT: a novel energy analysis toolkit for free-roaming smartphones," in *ACM conference on embedded network sensor systems*, 2014, pp. 16–30.

[4] T. Chainer, M. Schultz, P. Parida, and M. Gaynes, "Improving data center energy efficiency with advanced thermal management," *IEEE Trans. on Components, Packaging and Manufacturing Technology*, vol. 7, no. 8, pp. 1228–1239, 2017.

[5] R. Efraim, R. Ginosar, C. Weiser, and A. Mendelson, "Energy aware race to halt: A down to earth approach for platform energy management," *IEEE Computer Architecture Letters*, vol. 13, no. 1, pp. 25–28, 2012.

[6] A. Gainaru, H. Sun, G. Aupy, Y. Huo, B. A. Landman, and P. Raghavan, "On-the-fly scheduling versus reservation-based scheduling for unpredictable workflows," *The International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1140–1158, 2019.

[7] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, "Powerpack: Energy profiling and analysis of high-performance systems and applications," *IEEE Trans. on Parallel and Distributed Systems*, vol. 21, no. 5, pp. 658–671, 2009.

[8] A. Guermouche and A.-C. Orgerie, "Thermal design power and vectorized instructions behavior," *Concurrency and Computation: Practice and Experience*, 2021.

[9] G. Hager, J. Treibig, J. Habich, and G. Wellein, "Exploring performance and power properties of modern multi-core chips via simple machine models," *Concurrency and computation: practice and experience*, vol. 28, no. 2, pp. 189–210, 2016.

[10] Intel. (2017) Benchmarking gemm on intel® architecture processors. [Online]. Available: https://software.intel.com/

[11] ——, "Intel® 64 and ia-32 architectures software developer's manual," *Combined Volumes*, October 2019.

[12] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *International Conference on Power aware computing and systems*, 2010.

[13] I. Manousakis, F. S. Zakkak, P. Pratikakis, and D. S. Nikolopoulos, "TProf: An energy profiler for task-parallel programs," *Sustainable Computing: Informatics and Systems*, vol. 5, pp. 1–13, 2015.

[14] L. Mukhanov, P. Petoumenos, Z. Wang, N. Parasyris, D. S. Nikolopoulos, B. R. De Supinski, and H. Leather, "Alea: A fine-grained energy profiling tool," *ACM Trans. on Architecture and Code Optimization*, vol. 14, no. 1, 2017.

[15] A.-C. Orgerie, M. D. d. Assuncao, and L. Lefevre, "A survey on techniques for improving the energy efficiency of large-scale distributed systems," *ACM Computing Surveys*, vol. 46, no. 4, 2014.

[16] J. Pastor and J. M. Menaud, "Seduce: a testbed for research on thermal and power management in datacenters," in *International Conference on Software, Telecommunications and Computer Networks*, 2018.

[17] T. Patel, A. Wagenhäuser, C. Eibel, T. Hönig, T. Zeiser, and D. Tiwari, "What does Power Consumption Behavior of HPC Jobs Reveal? : Demystifying, Quantifying, and Predicting Power Consumption Characteristics," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020, pp. 799–809.

[18] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE micro*, vol. 32, no. 2, pp. 20–27, 2012.

[19] S. Schubert, D. Kostic, W. Zwaenepoel, and K. G. Shin, "Profiling software for energy consumption," in *IEEE International Conference on Green Computing and Communications*, 2012, pp. 515–522.

[20] C.-H. Tu, H.-H. Hsu, J.-H. Chen, C.-H. Chen, and S.-H. Hung, "Performance and power profiling for emulated android systems," *ACM Trans. on Design Automation of Electronic Systems*, vol. 19, no. 2, 2014.

[21] K. R. Vaddina, F. Brandner, G. Memmi, and P. Jouvelot, "Experimental energy profiling of energy-critical embedded applications," in *International Conference on Software, Telecommunications and Computer Networks*, 2017.

[22] K. R. Vaddina, J. M. Cebrián, and L. Natvig, "Transient temperature prediction for aging thermal sensors using artificial neural network," in *Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 2016, pp. 51–57.