The Impact of Turbo Frequency on the Energy, Performance, and Aging of Parallel Applications

Sandro Matheus V. N. Marques*, Thiarles S. Medeiros*, Fábio D. Rossi†, Marcelo C. Luizelli*,
Alessandro G. Girardi*, Antonio Carlos S. Beck‡, and Arthur F. Lorenzon*
† Federal Institute of Education, Science and Technology Farroupilha, Brazil
‡Institute of Informatics - Federal University of Rio Grande do Sul, Brazil
*Optimization Systems Laboratory - Federal University of Pampa, Brazil
aflorenzon@unipampa.edu.br

***Abstract*—Technologies that improve the performance of parallel applications by increasing the nominal operating frequency of processors respecting a given TDP (Thermal Design Power) have been widely used. However, they may impact on other non-functional requirements in different ways (e.g. increasing energy consumption or aging). Therefore, considering the huge number of configurations available, represented by the range of all possible combinations among different parallel applications, amount of threads, dynamic voltage and frequency scaling (DVFS) governors, boosting technologies and simultaneous multithreading (SMT), selecting the one that offers the best trade-off for a non-functional requirement is extremely challenging for software designers. Given that, in this work we assess the impact of changing these configurations on the energy consumption, performance, and aging of parallel applications on a turbo-compliant processor. Results show that there is no single configuration that would provide the best solution for all non-functional requirements at once. For instance, we demonstrate that the configuration that offers the best performance is the same one that has the worst impact on aging, accelerating it by up to 1.75 times. With our experiments, we provide guidelines for the developer when it comes to tuning performance using turbo boosting to save as much energy as possible and increase the lifespan of the hardware components[1].**

***Index Terms*—Turbo Frequency, TLP Exploitation, Aging, Energy.**

I. Introduction

The advance of modern multicore architectures has brought together the need for using more advanced DVFS (dynamic voltage and frequency scaling) techniques to control the operating frequency and voltage of each core. A DVFS system takes advantage of processor idleness (usually caused by I/O operations or by memory requests) to decrease the operating frequency of the core with a corresponding reduction in the supply voltage, which results in significant energy savings [1]. On the other hand, the so-called boosting techniques (e.g., Turbo Boost, Turbo CORE, and Game Mode) also apply DVFS, but to maximize processors performance by raising the nominal operating frequency within the limits of the Thermal Design Power (TDP) [2]. The Boosting technology was first introduced in the market by Intel through the implementation of the Turbo Boost in the Nehalem microarchitecture [3]. Later, Advanced Micro Devices (AMD) implemented it as Turbo CORE technology in its K10 microarchitecture [4]. Although both have some differences regarding the combination of frequencies and cores that are allowed to scale up/down, they share the same principle: given a TDP, the processor opportunistically increases the operating frequency of a group of cores, while the remaining ones run at a lower operating frequency or are powered down [2].

Such technologies have been largely used to improve performance of parallel applications from different domains [2], [5]–[8], which naturally contain parallel and sequential phases. In sequential phases (e.g., critical sections), as only one core is being used and is in the critical path of execution, the processor usually applies frequency boosting to this core while lowering the operating frequency or cutting off the supply of the other ones that are waiting. However, even though the boosting technology may reduce the execution time of parallel applications, many other metrics that are equally important as performance will not necessarily follow the same trend:

- Power dissipation and hence energy may increase due to the rising in the CPU frequency and supply voltage, which has quadratic influence in dynamic power.
- The temperature can also increase as a product of the power dissipation. This, in turn, could also raise the costs related to cooling.
- Higher operating temperatures profoundly influence aging phenomena in hardware components (e.g., negative bias temperature instability - NBTI and hot carrier injection - HCI). Therefore, temperature raising will increase the vulnerability of the systems, making them more susceptible to different types of failures (e.g., electromigration, dielectric breakdown, and stress migration) [9].

Therefore, at the same time that boosting techniques can improve performance, they may also negatively affect power dissipation, energy consumption, and CPU temperature, which will very likely reduce the lifetime of hardware components. This work assesses the influence of such technology on the aforementioned non-functional requirements. This is done by executing sixteen parallel applications from many domains on a 16-core AMD processor considering:

- Three different DVFS governors (*powersave*, *ondemand*,

[1]This study was financed in part by the Coordenao de Aperfeioamento de Pessoal de Nvel Superior - Brasil (CAPES) - Finance Code 001

978-1-7281-3915-9/19/$31.00 ©2019 IEEE

and *performance*)

- Eight possible configurations concerning the combination of the following features turned on/off: Turbo Core; Game Mode, and simultaneous multithreading (SMT).
- Exploiting TLP from 1 to 16 threads

We show that (*a*) there is no unique configuration that delivers the best outcome in all the non-functional requirements at the same time. (*b*) the configurations that provides either the best performance or the lowest processor temperature during the application execution are the ones that more degrade the processor. (*c*) the higher the operating frequency, the worst the energy consumption, energy-delay product (EDP [10]), temperature, and aging of memory-bound applications. The remainder of the paper is organized as follows. Related work is discussed in Section II. Section III describes the AMD Turbo Core. The methodology is discussed in Section IV, where the benchmarks and the execution environment are described. Section V discusses the results. Finally, Section VI draws the final considerations.

II. Related Work

We start by presenting the studies that assess the effects of the boosting technology on the performance of parallel applications, such as [11]–[13]. Gepner et al. [11] evaluate the Intel Turbo Boost on three different multicore processors and show that the performance can be improved by only 1.04% when the Turbo Boost is turned on for the Linpack benchmark because its execution without turbo frequency almost reaches the TDP limit quickly. Saini et al. [12] show that the Intel Turbo Boost 2.0 has increased the processor performance by up to 9% in comparison with its version 1.0. Verner et al. [13] discuss the behavior of Amdahl's Law model when the Intel Turbo Boost is turned on.

Many works have studied the influence of turbo frequency on the performance and energy consumption of parallel applications. Charles et al. [2] show that Turbo Boost reduces the execution time of sequential phases of parallel applications, improving their performance, but with an increase of up to 14.6% on the energy consumption. Esmaeilzadeh et al. [14] show that Turbo Boost is not energy efficient on the Intel Core i7. Wamhoff et al. [5] show that turbo frequencies can optimize the performance of applications that process a workload asymmetrically, by speeding up the cores with more workload.

Calore et al. [15] evaluate different DVFS techniques on HPC systems and conclude that the best trade-off between performance and energy consumption for memory-bound applications is achieved in lower values of CPU frequency. Lo et al. [4] show that turbo frequencies are generally beneficial for CPU-bound applications. However, for applications that share data and communicate through shared memory, the efficiency of the boosting feature is negatively affected due to the waiting time of the processor. Balaji et al. [16] observed that the execution with Intel Turbo Boost and Hyper-Threading feature turned on, the power consumption was increased by up to 17% when compared to the results achieved with such features turned off. Acun et al. [17] show that the temperature and power dissipation of each chip increases with the variations in the frequency caused by the Turbo Boost.

TABLE I: Comparison of our contributions with related work.

		[11] [19]	[12] [13]	[2] [14] [16]	[5]	[15]	[4]	[17] [18]	Our Work
Execution Envir.	*Seq. App.*	x		x	x		x		**x**
	Parallel App.	x	x	x	x	x	x	x	**x**
	Governors	1	1	1	1	4	1	1	**3**
	Game Mode								**x**
Evaluated Metrics	*Perf.*	x	x	x	x	x	x	x	**x**
	Energy			x	x	x	x	x	**x**
	EDP						x		**x**
	Temperature							x	**x**
	Aging								**x**

Furthermore, some works propose modifications in the boosting feature to optimize the behavior of applications. Wamhoff et al. [5] present a library to manually control the boosting on AMD and Intel processors. Raghavan et al. [18] propose a computational sprinting, which boosts the frequency and voltage to power levels that exceed the TDP for a few seconds in order to deliver better performance for end users. In the same context, Zahedi et al. [19] propose an architecture for computational sprinting, which decide whether to sprint based on application phases and system conditions.

Our Contributions. As one can observe in Table I, few works investigated both sequential and parallel applications, and they do so in a very limited environment or in a restricted number of metrics. In [2], [5], [14]–[16] only performance and energy are considered, while in [4], EDP is also evaluated. Even though the authors in [17] and [18] perform an extensive evaluation of performance, energy, and temperature, they consider only 1 DVFS governor and do not explore other metrics, such as EDP, aging, and reliability, as we do in this work. Therefore, this study extends the works above by performing an extensive evaluation of the AMD Turbo Core considering different DVFS governors and the game boosting feature looking for optimal points regarding many metrics (performance, energy, EDP, temperature, and aging); and executing a large range of applications with different memory and TLP behaviors.

III. Turbo Frequency

Boosting techniques can improve the processor core performance by extending the nominal operating frequency within the limits of the TDP [4], [20]. In modern architectures, the CPU frequency and power management of hardware components can be controlled by the advanced configuration and power interface (ACPI). It defines power and performance states (*C-states* and *P-states*, respectively): *C-states* are used to reduce power, and consequently, the energy consumption, by powering down subsystems; while *P-states* allow to change the frequency and voltage operating points of the core when it is active in order to either improve the performance or reduce the power consumption.

As one can observe in Figure 1, the basic *C-states* defined by the ACPI are the following: *CO*, in which the CPU/Core is active and executing instructions; and from *C1* to *Cn*, where

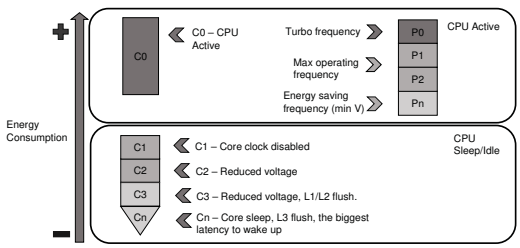


Fig. 1: Abstract ACPI's P-states and C-states organization

the CPU/Core is set to reduce the energy consumption by cutting the clock signal and power from unused cores, and switching off hardware components. In such cases, the more units switched off (the greater the value of *Cn*), the less energy is spent. On the other hand, *P-states* allow to change the frequency and voltage operating points of the CPU/core in order to either improve the performance or reduce the power consumption while it is active (*CO*). As one can note in Figure 1, there is a set of *P-states* that correspond to different operating points (i.e., pairs of frequency-voltage), and a given *P-state* refers to one specific point ranging from *P0* to *Pn*.

In processors that implement boosting technology and use the ACPI standard for power management, the *P0-state* represents the highest operating frequency reached when the boosting is turned on. When it is turned off, the *P1-state* is set, which represents the maximum base operating frequency without the interference of any boosting technique. On the other hand, the higher the value of *Pn*, the lower the operating frequency and voltage level [5]. ACPI implements methods that allow the operating system (OS) to change the *P-state* level (e.g., through the use of DVFS governors). In such cases, the OS selects an operating frequency while the voltage is set automatically by the processor. The most common governors are *performance* and *powersave*, in which the CPU frequency is set statically to the highest and lowest frequency, respectively; and *ondemand*, where the CPU frequency is set depending on the current system load [1].

AMD and Intel have some differences regarding the combination of frequencies and cores that can be used. Because this work uses an AMD Ryzen 1700 processor as a case study, we now briefly describe the organization of the AMD Turbo Core on it. This processor has sixteen cores (eight physical and eight logical through SMT support) and the Turbo Core works as follows: (*a*) when one or two physical cores are active, they can reach a frequency of up to 3.7 GHz; (*b*) when more than two physical cores are active, the maximum turbo frequency of all cores is set to 3.2 GHz [20]. Regardless the number of physical cores working at the turbo frequency, the unused cores will operate below the base frequency (3.0 GHz) in order to keep the power consumption and temperature within the limits of the TDP. On top of that, the AMD processor used in this study implements the Game Mode, which does the following: (*i*) it changes the memory from uniform memory access to non-uniform memory access. In this way, the system can determine between near and far memory to use up all

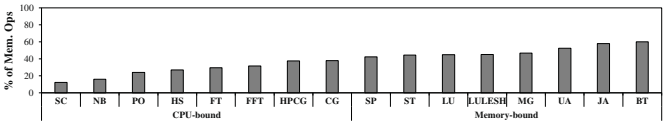


Fig. 2: Ratio of memory operations

available near memory first, before moving to higher latency memory; (*ii*) when there are more than one silicon die, the Game Mode applies power gating in some cores from one die to ensure that all active cores have access to near memory; and (*iii*) in the processor used in this work, all the physical cores active are set to operate at 3.5 GHz.

IV. Methodology

A. Benchmarks

Sixteen applications already written in C and C++ and parallelized with Open Multi-Processing (OpenMP) from assorted benchmark suites and domains were chosen. The benchmarks used were: **Seven kernels** from the NAS Parallel benchmark [21]: block tri-diagonal solver (BT), conjugate gradient (CG), discrete 3D fast Fourier Transform (FT), lower-upper gauss-seidel solver (LU), multi-grid on a sequence of meshes (MG), scalar penta-diagonal solver (SP), and unstructured adaptive mesh (UA). As the original version of NAS is written in FORTRAN, we use the OpenMP-C version developed by the authors in [22]. **Two applications** from the Rodinia Benchmark Suite [23]: *hotspot* (HS) and *streamcluster* (SC). **Seven applications** from different domains: *fast Fourier transform* (FFT) [24]; *the high performance conjugate gradient* (HPCG) benchmark [25]; *Jacobi Iteration Method* (JA) [26]; *Livermore unstructured Lagrangian explicit shock hydrodynamics* (LULESH2) [27]; *n-body* (NB) [26]; *Poisson* (PO) [26]; and *STREAM* (ST) [28]; As one can note in Figure 2, the chosen benchmarks cover a wide range of different memory accesses behaviors and are classified into two classes: CPU-Bound, which contain the applications with less than 40% of memory operations (from SC to CG); and Memory-bound, where the amount of memory operations is greater than 40% (from SP to BT). We measured the data memory accesses behavior and total of executed instructions through the AMD μProf.

B. Execution Environment

The experiments were performed on an AMD Ryzen 7 1700 (Zen) with SMT (16 threads total) running Ubuntu OS with Kernel v. 4.15. The normal operating frequency ranges from 1.5 GHz to 3.0 GHz when Turbo Core and Game Mode are turned off. Table II depicts the scenarios evaluated, in which the CPU frequency of each configuration follows the organization described in Section III. We executed each application with a medium input set (values predefined on each benchmark suite) and with a different number of threads, ranging from 1 to 16, even for the configurations that do not apply SMT. We also executed the applications with the *powersave*, *ondemand*, and *performance* DVFS governors. We compiled the applications with gcc/g++ 8.2.0, the optimization flag -O3, and the OpenMP distribution v. 5.0. The results

TABLE II: Scenarios evaluated in this work

Config	Turbo Core	Game Mode	SMT	Max Freq.	#Active Cores
TdGdSd	Off	Off	Off	3.0 GHz	8
TdGdSe	Off	Off	On	3.0 GHz	16
TdGeSd	Off	On	Off	3.5 GHz	8
TdGeSe	Off	On	On	3.5 GHz	16
TeGdSd	On	Off	Off	3.7 GHz	8
TeGdSe	On	Off	On	3.7 GHz	16
TeGeSd	On	On	Off	3.5 GHz	8
TeGeSe	On	On	On	3.5 GHz	16

T*: Turbo Core;* ***G****: Game Mode;* ***S****: SMT;* ***d****: disabled/turned off;* ***e****: enabled/turned on*

presented in the next section are the average of ten executions with $\sigma \leq 0.5\%$.

Section V discusses the results for the following metrics: performance, energy consumption, EDP, processor temperature and aging. The execution time of each application was obtained through the *omp_get_wtime* function, while the Application Power Management library was used to get the energy consumption. We used the Linux monitoring sensors to get the current CPU temperature/second directly from the hardware counters at run-time. We calculated the aging process based on the temperature of hardware components through the Arrhenius equation as defined by the authors in [29]. Although this equation does not include other reliability effects like thermal cycling and spatial thermal gradients, the exponential dependence of reliability on temperature is an important reason to favor lower operating temperatures [30]. The Arrhenius equation produces the acceleration factor (*Af*) of the aging process in the moment at which the temperature is measured (e.g., each second of the application execution). Hence, to estimate the total processor aging for the whole application execution, one can represent it as an integral of the *Af* (from Equation 1) over the application execution time (*Et*), as depicted in Equation 1.

$$ProcAging = \int_{i=0}^{Et} Af \tag{1}$$

V. Results

A. Performance, Energy, and EDP

We start by showing the results for the *performance* DVFS governor because it has presented the most significant results. Figure 3 presents the results for each benchmark class w.r.t. the geometric mean of the applications within each class considering all scenarios discussed earlier. Results are normalized to the standard way that AMD processors are configured, that is, ***TdGdSe*** (Table II). Values lower than 1 mean that the particular configuration is better than the baseline.

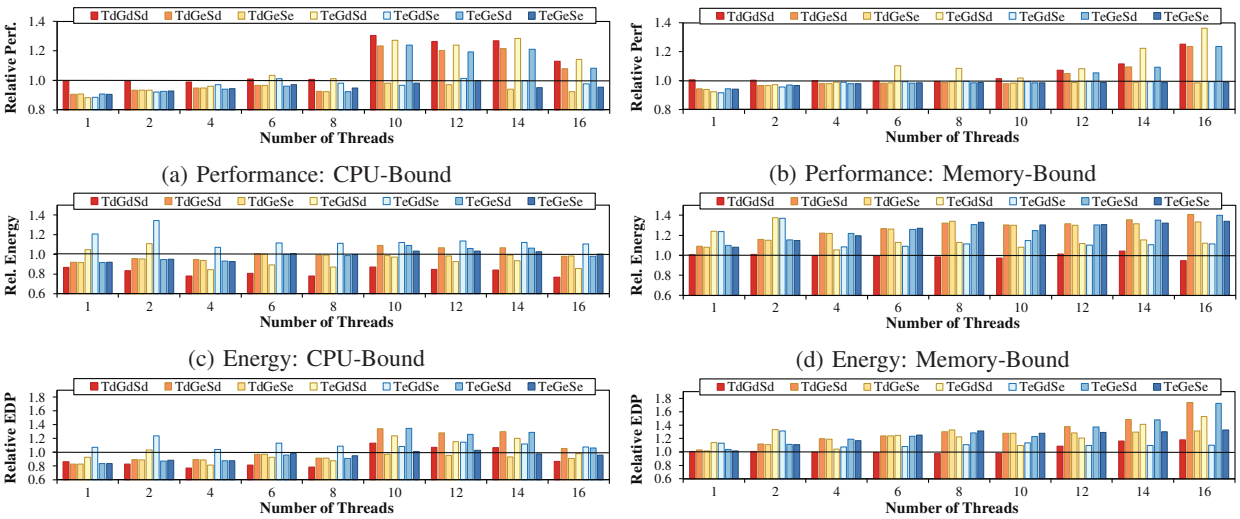
Let us discuss the behavior of CPU-bound applications. The best performance for the sequential versions (1 thread) is achieved with the CPU frequency set at the maximum (***TeGdSd*** and ***TeGdSe***), in which the execution time is 12% lower than the baseline (Figure 3a). However, as the number of threads increases, the performance difference between the configurations that apply turbo frequency and the ones that do not apply it decreases because of the following reasons: (*i*) the greater the number of active cores, the lower the max frequency reached by each core, which reduces the processor performance when Turbo Core is turned on; and (*ii*) as the number of threads increases, characteristics that influence parallel CPU-bound applications start to play a decisive role overlapping the gains achieved by the turbo frequency, such as data-synchronization and functional units saturation [31], [32]. Besides that, all the configurations with SMT enabled presented better performance than the ones with it turned off for the execution with more than 8 threads, showing the advantages of this feature.

Differently from the performance behavior, the lowest energy consumption is achieved when all features are turned off (Figure 3c - ***TdGdSd***). In the most significant case (16 threads), it consumes 36% less energy than the baseline. On the other hand, the worst outcome is when each core operates with Turbo Core and SMT turned on (***TeGdSe***), spending 35% more energy than the baseline (2 threads). Through the comparison between the results achieved by the ***TeGdSe*** and ***TeGdSd***, we can observe that by just turning on the SMT feature, the energy consumption increased by 21% (average considering the number of threads and benchmark set). The very same behavior is observed for the EDP results (Figure 3e), where ***TdGdSd*** presented better results while ***TeGdSe*** the worst ones.

When considering the performance of memory-bound applications (Figure 3b), the turbo frequency plays a decisive role only for the execution with 1 and 2 threads. In such cases, all configurations that apply boosting presented lower execution time than the baseline. However, from this point on, two characteristics that limit the performance improvements of this kind of parallel application overlap the theoretical gains of the turbo frequency: overhead of the communication between threads through shared memory regions and off-chip bus saturation [7], [31]. For the energy consumption (Figure 3d), the two configurations with turbo frequency turned off delivered better results (***TdGdSd*** and ***TdGdSe***). One can also note that the worst outcome is when turbo frequency is turned on because the raising in the CPU frequency did not result in any performance improvements when compared to the configurations with boosting turned off (Figure 3b). In this sense, the higher the frequency, the greater the power consumption and therefore the higher the energy consumption. As an example, for the execution with 1 and 2 threads, the configurations with CPU frequency set to 3.7 GHz presented the worst results, while the configurations with CPU frequency set to 3.5 GHz spent more energy on the execution with 4 to 16 threads. The very same behavior observed on the energy consumption repeats on the EDP results (Figure 3f).

B. Processor Temperature and Aging

Figure 4 presents the average processor temperature and aging (calculated through Eq. 1) of each configuration normalized to the baseline (***TdGdSe***). Let us first discuss the results of temperature where, in most cases, the execution of the applications with all the features turned off (***TdGdSd***) presented the best result (i.e., lowest temperature) regardless the benchmark class. On the other hand, the worst result in



(a) Performance: CPU-Bound

(b) Performance: Memory-Bound

(c) Energy: CPU-Bound

(d) Energy: Memory-Bound

(e) EDP: CPU-Bound

(f) EDP: Memory-Bound

Fig. 3: Performance, energy, and EDP results considering the ***TdGdSe*** as the baseline (1.0) for the *performance* DVFS governor

most cases of the parallel versions is achieved either when all the features are turned on (***TeGeSe***) or when the processor frequency is set to the maximum possible. Because of the CPU frequency and voltage level influence the power and heat dissipation, keeping the processor operating at lower frequencies and using fewer resources as possible (i.e., SMT turned off) will very likely deliver better temperature results.

When considering the impact of turbo frequency on the processor aging (Figures 4c and 4d), the following behaviors can be listed: (*i*) for the execution with 1 and 2 threads regardless the benchmark class, the configurations in which each active core can operate at up to 3.7 GHz had worst impact on the processor aging due to the higher operating temperature (Figures 4a and 4b); (*ii*) when the application is executed with more than four threads, the Game Mode starts to play a negative role on the processor aging because of the higher temperature caused by higher levels of operating frequency (3.5 GHz for all active cores) and supply voltage; (*iii*) The greater the number of threads, the worst is the behavior of turbo frequency techniques on the aging. This is because the performance improvements are not linear in both benchmark classes (discussed in Section V-A) and hence, the only outcome of increasing the CPU frequency is the raising on the power dissipation, which negatively influences aging.

C. Optimized use of Turbo Frequency

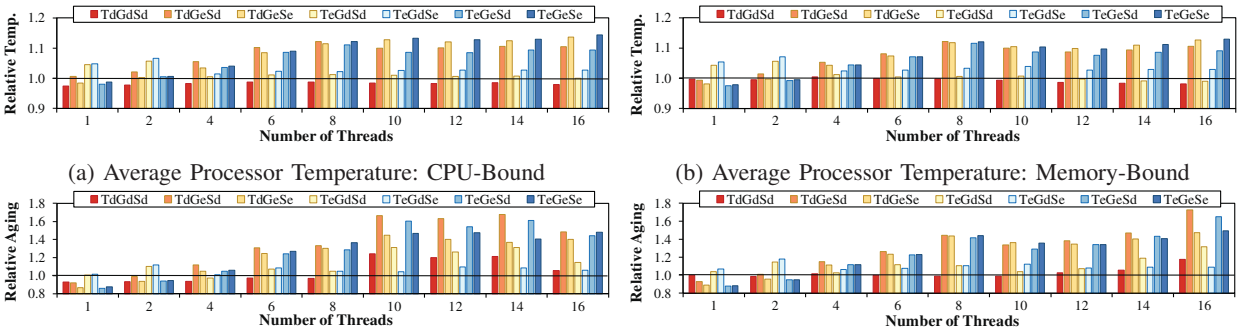
In order to provide insights to software designers, Table III indicates which combination of Turbo Core, Game Mode, SMT, TLP exploitation, and DVFS governor is the best choice regarding the metrics evaluated in this work for each benchmark class. Here, we are not considering relative numbers anymore, but raw values. Besides that, it also shows the results of the best configuration on the other metrics. As one can observe, there is no single configuration that delivers the best outcome for all metrics at the same time. Therefore, the software designer has to choose the combination of features that deliver the best outcome w.r.t to the target metric.

The best performance for each benchmark class is achieved when the application is executed with the number of physical cores (8) and at least one of the turbo frequencies are turned on with the DVFS governor is set to performance. It reinforces the concept that the processor frequency is extremely important to provide better performance. Nonetheless, at the same time that such configurations deliver the lowest execution time, they also present a negative impact on the other metrics. On the other hand, the lowest energy consumption is achieved with the DVFS governor set to *powersave*. It is important to note that when the processor is set in such governor, the turbo frequency level is not reached.

One can also note that the configurations with the lowest processor temperature are the ones with the worst results in the aging. Therefore, even if such configurations can reduce the costs related to cooling solutions, they reduce the processors lifetime by up to 3.6 times for CPU-bound applications. Furthermore, when the configuration is defined to reduce the aging, performance is not so affected as if the opposite is considered. For example, when the software designer wants to optimize the performance of memory-bound applications, it ages the processor 75% faster than the solution that reduces the aging, while the best configuration for the aging is only 13% slower than the optimized solution for performance.

VI. Conclusions and Future Work

This work analyzed the impact of turbo frequency on the energy consumption, performance, and aging of a turbo-compliant processor using different parallel applications. There is no set of settings that is universally optimal in balancing all of the presented non-functional requirements discussed above. Based on the results of the experiments, we can highlight some findings: (*i*) positive results in terms of such requirements depend on the behavior of the application, (*ii*) the organization of the application on the underlying architecture can influence the improvement of those non-functional



(a) Average Processor Temperature: CPU-Bound

(b) Average Processor Temperature: Memory-Bound

(c) Processor Aging: CPU-Bound

(d) Processor Aging: Memory-Bound

Fig. 4: Processor temperature and aging considering the ***TdGdSe*** as the baseline (1.0) for the *performance* DVFS governor

TABLE III: Best configuration for each non-functional requirement

		CPU-Bound					Memory-Bound				
		Time	**Energy**	**EDP**	**Temp.**	**Aging**	**Time**	**Energy**	**EDP**	**Temp.**	**Aging**
Best	**DVFS**	***performance***	***powersave***	***performance***	***powersave***	***performance***	***performance***	***powersave***	***powersave***	***powersave***	***powersave***
	Config	***TeGeSd***	***TdGdSd***	***TdGdSd***	***TeGeSd***	***TdGdSd***	***TeGdSe***	***TeGdSe***	***TeGdSe***	***TeGeSd***	***TdGdSe***
	TLP	***8***	***7***	***7***	***1***	***7***	***8***	***6***	***6***	***1***	***6***
Time (s)		**78.9**	114.8	86.0	452.5	86.0	**213.0**	246.7	246.7	505.0	242.5
Energy (J)		4178.6	**2792.3**	3121.6	6621.6	3121.6	8875.3	**4965.0**	4965.0	7578.6	5057.9
EDP		329770	320638	**268436**	2996507	268436	1890438	1224957	**1224957**	3826827	1226735
Temperature (°C)		39.3	33	34.6	**30.4**	34.6	41.6	32.9	32.9	**30.3**	34.1
Aging (Eq. 1)		176.3	149.2	128.9	467.7	**128.9**	577.2	386.4	386.4	516.8	**329.7**

requirements, and (*iii*) trade-offs between such requirements cannot be eliminated but can be mitigated. As future work, we intend to develop a library to automatically select at run-time the best configuration for a given requirement.

REFERENCES

[1] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *ICPACS*, 2010, pp. 1–8.

[2] J. Charles, P. Jassi, N. S. Ananth, A. Sadat, and A. Fedorova, "Evaluation of the intel® core i7 turbo boost feature," in *IISWC*, 2009, pp. 188–197.

[3] 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.

[4] D. Lo and C. Kozyrakis, "Dynamic management of turbomode in modern multi-core chips," in *IEEE HPCA*. IEEE, 2014, pp. 603–613.

[5] J.-T. Wamhoff, S. Diestelhorst, and C. Fetzer, "The turbo diaries: Application-controlled frequency scaling explained."

[6] M. Annavaram, E. Grochowski, and J. Shen, "Mitigating amdahl's law through epi throttling," in *ISCA*. USA: IEEE, 2005, pp. 298–309.

[7] A. F. Lorenzon, M. C. Cera, and A. C. S. Beck, "Investigating different general-purpose and embedded multicores to achieve optimal trade-offs between performance and energy," *Journal of Parallel and Distributed Computing*, vol. 95, pp. 107–123, 2016.

[8] W. dos Santos Marques, P. S. S. de Souza, A. F. Lorenzon, A. C. S. Beck, M. B. Rutzig, and F. D. Rossi, "Improving edp in multi-core embedded systems through multidimensional frequency scaling," in *IEEE ISCAS*. IEEE, 2017, pp. 1–4.

[9] S. Corbetta and W. Fornaciari, "Nbti mitigation in microprocessor designs," in *GLSVLSI*. NY, USA: ACM, 2012, pp. 33–38.

[10] R. Gonzalez and M. Horowitz, "Energy dissipation in general purpose microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 9, pp. 1277–1284, Sep. 1996.

[11] P. Gepner, M. F. Kowalik, D. L. Fraser, and K. Wackowski, "Early performance evaluation of new six-core intel® xeon® 5600 family processors for hpc," in *ISPDC*. IEEE, 2010, pp. 117–124.

[12] S. Saini, J. Chang, and H. Jin, "Performance evaluation of the intel sandy bridge based nasa pleiades using scientific and engineering applications," in *IWPMBSHPCS*. Springer, 2013, pp. 25–51.

[13] U. Verner, A. Mendelson, and A. Schuster, "Extending amdahls law for multicores with turbo boost," *IEEE Computer Architecture Letters*, vol. 16, no. 1, pp. 30–33, 2017.

[14] H. Esmaeilzadeh, T. Cao, Y. Xi, S. M. Blackburn, and K. S. McKinley, "Looking back on the language and hardware revolutions: Measured power, performance, and scaling," in *ASPLOS*. New York, NY, USA: ACM, 2011, pp. 319–332.

[15] E. Calore, A. Gabbana, S. F. Schifano, and R. Tripiccione, "Evaluation of dvfs techniques on modern hpc processors and accelerators for energy-aware applications," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, p. e4143, 2017.

[16] B. Balaji, J. McCullough, R. K. Gupta, and Y. Agarwal, "Accurate characterization of the variability in power consumption in modern mobile processors," in *USENIX*. USA: USENIX, 2012, pp. 8–8.

[17] B. Acun, P. Miller, and L. V. Kale, "Variation among processors under turbo boost in hpc systems," in *Int. Conf. on Supercomputing*. ACM, 2016, p. 6.

[18] A. Raghavan, L. Emurian, L. Shao, M. Papaefthymiou, K. P. Pipe, T. F. Wenisch, and M. M. Martin, "Computational sprinting on a hardware/software testbed," in *ASPLOS*. NY, USA: ACM, 2013, pp. 155–166.

[19] S. M. Zahedi, S. Fan, M. Faw, E. Cole, and B. C. Lee, "Computational sprinting: Architecture, dynamics, and strategies," *ACM Trans. Comput. Syst.*, vol. 34, no. 4, pp. 12:1–12:26, Jan. 2017.

[20] A. Branover, D. Foley, and M. Steinman, "Amd fusion apu: Llano," *Ieee Micro*, vol. 32, no. 2, pp. 28–37, 2012.

[21] D. H. Bailey, E. Barszcz, J. T. Barton, and et. al., "The nas parallel benchmarks: Summary and preliminary results," in *ACM/IEEE Conf. on Supercomputing*. NY, USA: ACM, 1991, pp. 158–165.

[22] S. Seo, G. Jo, and J. Lee, "Performance characterization of the nas parallel benchmarks in opencl," in *IEEE ISWC*, 2011, pp. 137–148.

[23] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *IEEE ISWC*. DC, USA: IEEE, 2009, pp. 44–54.

[24] W. Petersen and P. Arbenz, *Introduction to Parallel Computing : A practical guide with examples in C*, ser. Oxford Texts in Applied and Engineering Mathematics. OUP Oxford, 2004.

[25] J. Dongarra, M. A. Heroux, and P. Luszczek, "Hpcg benchmark: A new metric for ranking high performance computing systems," Knoxville, Tennessee, 2015, pp. 1–11.

[26] M. Quinn, *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill Higher Education, 2004.

[27] I. Karlin, J. Keasler, and R. Neely, "Lulesh 2.0: Updates and changes." 2013, pp. 1–9.

[28] J. D. McCalpin, "Memory bandwidth and machine balance in current high performance computers," *IEEE Computer Society Technical Committee on Computer Architecture Newsletter*, pp. 19–25, 1995.

[29] S. N. Storino, "Method and apparatus for estimating remaining life of a product," Aug. 10 2004, uS Patent 6,775,624.

[30] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture: Extended discussion and results," *University of Virginia, Tech. Report*, vol. 8, 2003.

[31] A. F. Lorenzon, C. C. de Oliveira, J. D. Souza, and A. C. S. Beck, "Aurora: Seamless optimization of openmp applications," *IEEE TPDS*, vol. 30, no. 5, pp. 1007–1021, May 2019.

[32] A. F. Lorenzon, M. C. Cera, and A. C. S. Beck, "Performance and energy evaluation of different multi-threading interfaces in embedded and general purpose systems," *Journal of Signal Processing Systems*, vol. 80, no. 3, pp. 295–307, 2015.