The Impact of CPU Frequency Scaling on Power Consumption of Computing Infrastructures

PThreads has the lowest power consumption, consuming less than the sequential version for memory-bound benchmarks. Regarding performance, the performance governor achieved 3% of performance over theondemand

Keywords: PAMPAR CPU Frequency Governors Power consumption.

Nowadays, governments are imposing limits on the power and energy consumption for supercomputing infrastructures [5]. The consequence is that di erent supercomputer manufacturers are making an e ort to reduce this consumption [24]. The list of the world's most energy-e cient supercomputers called the Green500 [5] has been showing signi cant changes in the latest results. The USA and some countries in Europe that led the Top500 (the 500 most powerful supercomputers) are not reaching good positions in the Green500. However, that scenario has changed in the past two years. These countries have begun to make investments in Computational Science to improve the energy e ciency of their supercomputing infrastructures and now they lead the Green500 list [5]. Therefore, ways of reducing energy consumption without losing computational performance are widely discussed today.

Computer performance is the amount of useful work accomplished by a computer system [20]. This amount can be increased using parallel programming. To achieve performance with parallel programming, it is necessary to use di erent processing units to execute di erent parts of a program concurrently. However, a program does not run 100% in parallel. At a certain part of the execution, these parties need to communicate to exchange information. This happens at least once at the beginning and again at the end of the program. This communication can occur either through access to shared memory addresses or through the exchange of messages. The problem is that these communication operations can cause extra energy expenditure. While parallelism allows for increased program performance, the need for task-to-task communication can impact the power and energy consumption [23]. Thus, although parallelism allows performance gains, this can lead to higher power and energy consumption. This power and energy consumption grows mainly according to the number of processors that are used in parallel and the volume of communication among them. This increase may present a signi cant impact on the power and energy consumption of supercomputing infrastructures. On the other hand, the reduction in execution time allowed by the parallelization causes a decrease in the total energy consumption in some cases.

Parallelism techniques can be implemented in a program using Parallel Programming Interfaces (PPIs). Some PPIs are best suited for speci c languages, platforms, and architectures. Some interfaces use memory to communicate among di erent parallel tasks for programs running on processors that share memory regions. For programs that will run on a distributed architecture, some PPIs do

the communication through the exchange of messages, such as Message Passing Interface (MPI). In addition to these main features, several other factors and peculiarities of each PPI can impact the performance of a parallel program. One of these factors is the frequency of the processor, which is adjusted through di erent governors. These governors can prioritize performance or energy consumption. These governors can set the processor frequency at maximum (more performance) and minimum (more power savings).

As seen, many factors can impact the performance, power and energy consumption of parallel programs. It is necessary to understand better the role that di erent PPIs play when used with variations in processor frequency. With this understanding, it is possible to nd the best trade-o s between performance, power and energy consumption. Contributing to this subject, this paper evaluates the power and energy consumption of di erent parallel programming interfaces using di erent CPU governors. We evaluate the behavior of the PPIs with the governor's `ondemand`, `performance`, and `powersave`. These governors were chosen because they can highlight the characteristics of the parallel applications. The goal of our work is to show the impact of di erent PPIs and applications on the power and energy consumption and performance varying the governor policies of the processor (`ondemand`, `performance`, and `powersave`) and nd out how it is impacted according to speci c application characteristics.

To achieve the goal, in this paper, we run the PAMPAR[4] suite [8] changing the governor settings. PAMPAR consists of 13 parallel benchmarks developed to evaluate the performance and energy consumption of PPIs. Despite the serial code, each benchmark is implemented using well-known PPIs in the Computational Science eld, such as PThreads, OpenMP, MPI-1, and MPI-2 (dynamic process spawn). The authors of the suite ran experiments to measure the performance and energy consumption of PAMPAR benchmarks to estimate the power and energy consumption of the PPIs without regard to the CPU-enabled governor [9,10]. In this paper, we explore this gap.

The remainder of this work is organized as follows. In Section 2, we introduce the governors for frequency scaling and present more details about the benchmark selection. Section 3 shows how our experiments were structured and bring some information to a better understanding of the results. Section 4 discusses the results. The related works are discussed in Section 5 and, nally, Section section 6 draws the nal considerations and future works.

2.1 CPU Frequency Scaling Governors

The CPU frequency scaling allows the operating system to increase or decrease the CPU frequency to save power [15]. CPU frequencies can be scaled automatically depending on the system load, in response to ACPI (Advanced Con guration and Power Interface) events, or manually by the user. CPU frequency

[4] https://github.com/adrianomg/PAMPAR

scaling is implemented in the Linux kernel, and the infrastructure is called `CPUFreq`. Since kernel 3.4, the required modules are loaded automatically, and the `ondemand` governor is activated by default. However, other governors can be enabled, such as `powersave`, `performance`, `userspace`, or `conservative`. Each governor has its unique behavior, purpose, and suitability in terms of workload [18]. In this paper, we evaluate performance, power and energy consumption. In this way, we investigate the `ondemand`, `performance`, and `powersave` governors.

The `ondemand` is a dynamic governor that uses CPU load as a metric to select the CPU frequency. It measures the time elapsed between consecutive invocations of its worker routine and computes the fraction of that time in which the given CPU was not idle. This way, it can estimate the current CPU load. The ratio of the active time (non-idle) to the total CPU time is taken as an estimate of the load. In our work, this governor is attached to a policy shared by multiple CPUs, so the load is estimated for all of them, and the best result is used as the load estimate for the entire policy.

To achieve the highest possible clock frequency by the CPU, the governor `performance` can be enabled. Once enabled, the highest frequency will be set statically and will not change. It is indicated for cases in which the CPU will deal with a heavy workload or will be rarely idle, at least. That is because this particular governor is opposed to the power saving bene t.

Contrary to `performance`, the `powersave` governor forces the processor to use the lowest possible clock frequency. Once activated, the operation of this governor is similar to `performance`. However, in this case, the lowest frequency will be statically adjusted and will not change. Therefore, as expected, this particular governor o ers maximum power savings, but at the cost of lower CPU performance.

2.2 Benchmark

The objective of this work is to investigate the impact of di erent CPU governors and PPIs on power and energy consumption. However, it is di cult to nd a well-known benchmark suite that o ers a diverse set of benchmarks implemented with several PPIs for general-purpose architectures. The most parallel suites do not o er a set of benchmarks representing diverse domains and fully parallelized in many PPIs (e.g., NPB, PARSEC, Rodinia, etc.). Most of them implement no more than a couple of PPIs or only small subsets in di erent PPIs, and these subsets do not always match the same benchmarks.

This way, we looked for any other benchmark suite that could be suitable for our goals. Thus, we found PAMPAR. It is a new parallel benchmark suite with 13 C/C++ benchmarks from many domains, such as physics, engineering, chemistry, image processing, pattern recognition, biological simulation, linear algebra, etc. The suite consists of 3 micro benchmarks, seven kernels, and three pseudo-applications. They were developed to establish a relationship between performance and energy consumption in embedded systems and general-purpose architectures [9]. The main factor that makes PAMPAR suitable for our work is

that all benchmarks on the suite are parallelized in 4 PPIs: PThreads, OpenMP, MPI-1, and MPI-2. These PPIs are also the target of this work because they are the most widespread in the Computational Science eld. Besides this, they are supported by most multicore based infrastructures, both embedded and general-purpose.

The experiments were carried out on a computer equipped with 2 Intel® Xeon® Silver 4116 processors. Each processor has 12 physical cores operating at the standard 2.1 GHz frequency and 3 GHz turbo frequency. Its memory system consists of three levels of cache: a 32 KB L1i and 32 KB L1d cache and a 1 MB cache L2 for each core. Level L3 has a 16.5 MB cache shared between all cores using Smart Cache technology. The main memory (RAM) is 96 GB in size and DDR4 technology. The operating system is Linux Debian kernel version 4.19.0-8 using GNU GCC 9.3 compiler with `-O2` optimization ag, OpenMP 4.5, and OpenMPI 3.1.3. This machine represents a node in a large supercomputing infrastructure.

PAMPAR benchmark 1was used for the experiments. We selected the Medium workload class. This class includes 2048×2048 input matrices for benchmarks that use matrices, for instance. To increase the accuracy of experiments, the results presented in section 4 are the average of 10 executions of each benchmark. The results graphs show the average energy consumption values and the 95% con dence intervals according to the Student's t-distribution [19]. During the experiments, the computer remained locked to ensure that other applications did not interfere with the results.

The benchmarks deployed with MPI-2 begin the execution with a single process. Then, this process (parent process) invokes new child processes that do not need to be identical to the parent. After creating a child process, it will belong to an intra-communicator, and the communication between parent and child will occur through this communicator.

The Intel® Performance Counter Monitor (PCM) 2.0 toolkit was used to measure energy consumption [13]. It has a tool to monitor the power states of the processor and DRAM memory. This way, the total energy consumption is the sum of the energy required by the DRAM modules and core domains (CPU and cache memories). For the execution time, the time in the beginning and the end of each benchmark's main function was measured using the GNU time library, and the di erence of these values was used.

4.1 Power and Energy consumption

In this section, we present the energy consumption results of PAMPAR benchmarks and their PPIs impact over di erent governors for the CPUs. Figure 1,

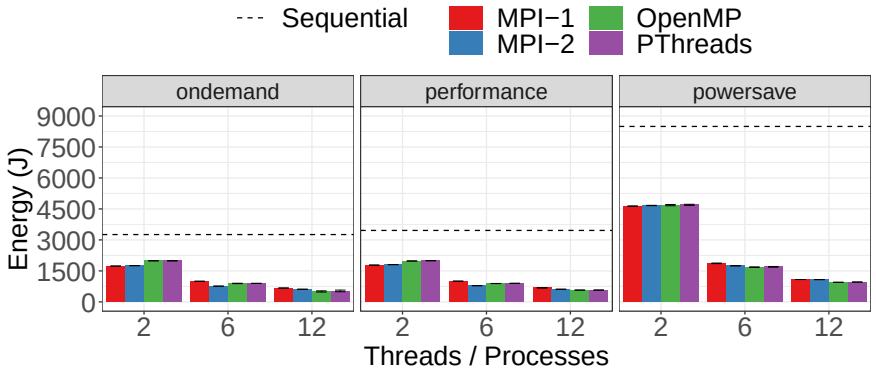
Table 1 and Figure 2 show the results for different benchmarks. The figures are charts, where facets are different governors, and bars are the energy consumption in joules for each PPI. Each chart displays the results by benchmarks individually. These results refer to running using two, six, and twelve parallel threads/processes for each case. Also, a dashed horizontal line represents the sequential result of the respective benchmark.

In Figure 1 and Table 1 are presented results for the CPU-bound benchmarks. The exception is DJ, JA, and MM benchmarks that are not fully CPU-bound, as they do a bit more memory access, but not enough to be classified as highly memory-bound. These results show that `powersave` governor consumes the triple of energy than `ondemand` and performance, on average. For all the results that present this behavior, this difference is reduced as the number of parallel tasks increases. However, since the execution time of `performance` and `ondemand` are low, they present higher power consumption results. Moreover, it shows that `performance` governor consumes 4.8% more energy than `ondemand` due to its high frequency used per core. For CPU-bound benchmarks using `ondemand` governor on this infrastructure, great power and energy consumption savings are not expected because this kind of application requires high computing power in most of its execution time.

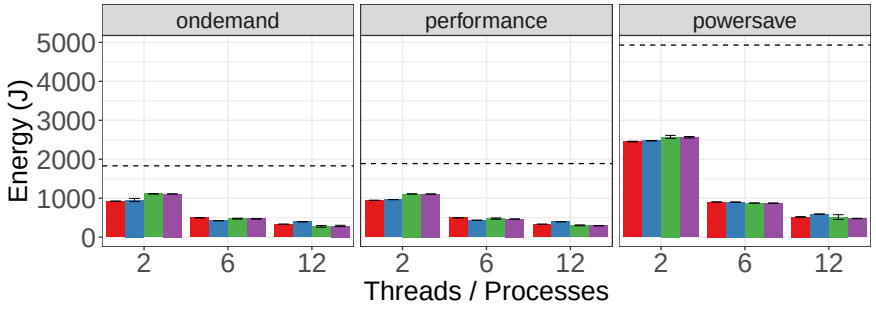
Moreover, looking more closely at PPI behavior, it is possible to observe that `ondemand` has less impact on OpenMP and PThreads than MPI-1 and MPI-2. For OpenMP and PThreads the `performance` governor consumes 1.4% and 2.2% more energy, while for MPI-1 and MPI-2, it is 4.8% and 9.7%. The difference in these PPIs can be explained in the context of threads and processes. Threads are often a lighter type of process for the system, while processes are heavier. A thread shares with other threads its code area, data, and operating system resources. Because of this sharing, the operating system needs to deal with less scheduling costs and thread creation, when compared to context switching by processes|all of these factors impact performance and, consequently, on energy. Also, in a larger infrastructure using distributed processing, it would impact much more.

The DJ benchmark shows the highest energy savings using `ondemand` and `performance` over the `powersave` governor, about three times less. Using 12 threads, `performance` governor consumed 12.5% more energy for OpenMP than `ondemand`. We can also see that the tendency of `powersave` to consume triple the energy with two threads, is not confirmed with 12 threads, where the consumption does not reach double. For DFT and HA the `performance` energy consumption overhead varies from 2.7% to 5.1%. The MM presents an unusual behavior, that is, for this benchmark `powersave` governor consumes less than twice the energy consumed by `ondemand` and `performance`. We hypothesize that once this benchmark mixes computation with memory operations, which would execute in a low frequency, the `ondemand` governor takes a wrong decision, decreasing cores frequency when the maximum frequency would be more effective.

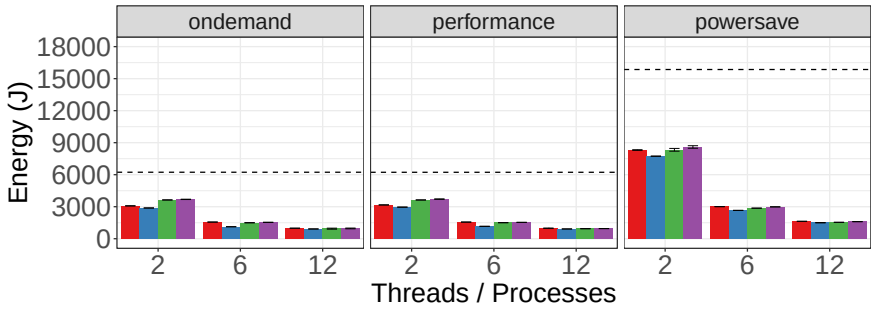
The DP benchmark using `ondemand` and `performance` governors show a different PThreads behavior regarding the other PPIs with low number of parallel



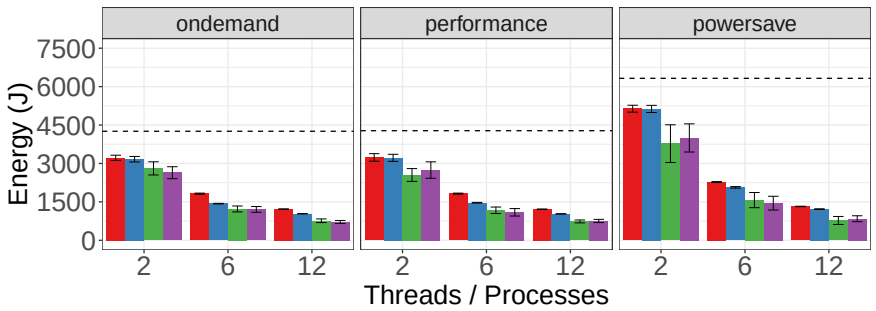
(a) Dijkstra - DJ.



(b) Disc. Fourier Transf. - DFT.



(c) Harmonic Sums - HA.



(d) Matrix Multiplication - MM.

Fig. 1. Energy consumption for CPU-bound benchmarks.

Table 1. Energy consumption for CPU-bound benchmarks (joules).

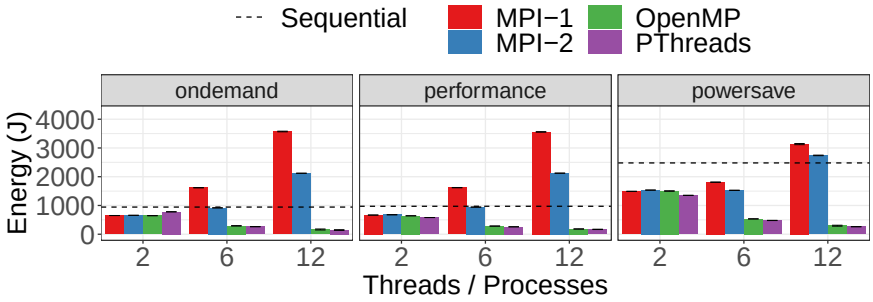
		MPI-1		MPI-2		OpenMP		Pthreads	
Threads/Processes		6	12	6	12	6	12	6	12
DP	Ondemand	402.27	265.15	360.08	369.75	319.62	170.74	341.62	211.31
	Performace	403.75	267.18	368.12	367.18	349.15	181.30	352.35	224.13
	Powersave	729.65	412.03	738.71	502.54	572.84	304.51	681.68	362.36
NI	Ondemand	236.80	161.16	253.27	266.56	207.56	122.93	189.58	112.40
	Performace	237.99	161.36	262.37	267.10	208.15	134.47	190.25	120.06
	Powersave	447.07	260.23	474.71	356.52	417.31	222.03	380.96	202.09
PI	Ondemand	415.34	268.92	382.37	380.02	387.84	218.55	387.76	229.41
	Performace	413.46	269.70	390.60	383.34	384.41	228.39	386.86	233.60
	Powersave	789.08	443.19	792.24	519.20	755.86	405.23	759.45	405.47

tasks. With six and twelve threads, PThreads follows the pattern that is seen in most CPU-Bound benchmarks. However, using two threads, this consumption exceeds the consumption of other PPIs. The other three PPIs follow the pattern by increasing the number of threads, where MPI-2 consumes less energy than MPI-1 and OpenMP. This behavior is not the same for `performance` and `powersave`.

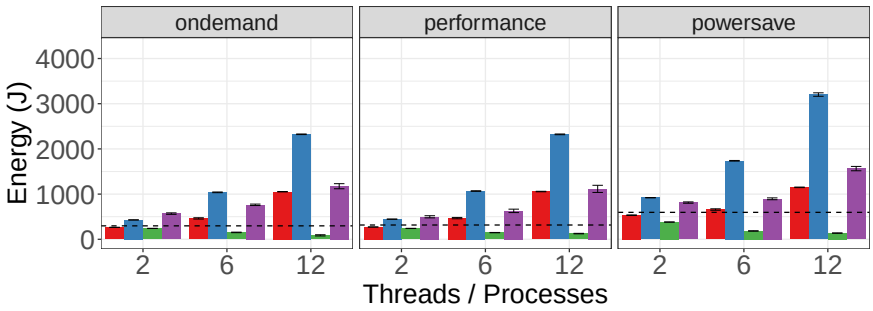
Table 1 shows the results of energy consumption of DP, NI and PI benchmarks. They are three CPU-bound micro-benchmarks that perform simple iterative operations. We present the values in a table because the graphs were very similar for di erences to be noticed. In this table it can be seen that `ondemand` and `performance` consumed almost the same amount of energy with MPI-1 and MPI-2. But OpenMP and PThreads showed di erences between these two governors. In DP Pthreads with 6 threads, `ondemenand` consumed 3.2% less energy than the `performance` governor. Using 12 threads,`performance` consumed 6.2% more energy. Regarding NI OpenMP with 12 threads,`performance` consumed 9.8% more energy than`ondemand`. The `powersave` governor showed the same behavior seen in the previous benchmarks.

MM shows similar behavior to DP in `performance` and `ondemand`, but for both MPI PPIs. The consumption of these two PPIs grows at a higher rate than the other PPIs as the number of threads increases. The di erence in this benchmark is that OpenMP also uses more power than both MPI PPIs. For two threads, the rate of increase is not as high as PThreads, but for six and twelve, it consumes the same or more. Also, using MPI-1 and MPI-2, this benchmark was the one that most approached the consumption of the sequential version among the CPU-Bound benchmarks.

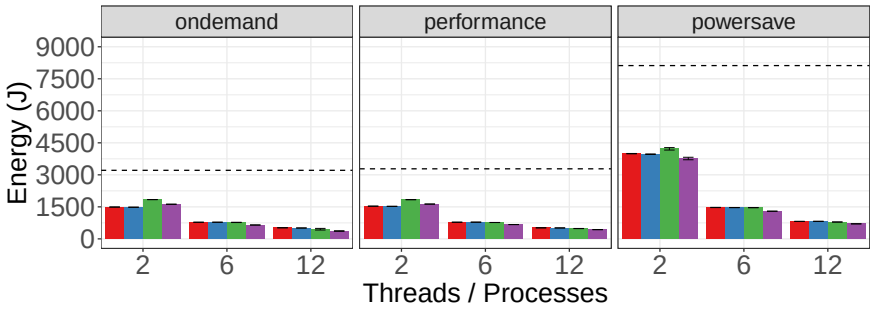
In the memory-bound benchmarks (Figure 2), it is possible to observe that `ondemand` has a high impact on Pthreads. For MPI-1, MPI-2, and OpenMP, `performance` governor consumes 1.8%, 2.7%, and 2.3% less energy, respectively, while for Pthreads, it consumes up to 14.3% less, which represents 27.8% of power



(a) Jacobi Method - JA.



(b) Gram-Schmidt - GS.



(c) Turing Ring - TR.

Fig. 2. Energy consumption for memory-bound benchmarks.

increase. The GS and JA benchmarks show the highest energy consumption for MPI. On average, `performance` governor consumes 13.8% and 11.4% more power, respectively, while for other memory-bound benchmarks, it represents less than 7.2%.`Powersave` governor has almost the same behavior as for CPU-bound benchmarks. It consumes 43.1% less power than `performance` governor and 37.9% less than `ondemand`. It is expected, as memory-bound benchmarks consume most of its time in-memory operations and the workloads we use to t into the infrastructure's cache memory. So these benchmarks generally do not need to access main memory, which would be more costly in terms of energy. This way, the `powersave` takes advantage through the decrease in the CPU frequencies while these memory operations are done.

The low power consumption by PThreads in memory-bound benchmarks does not represent that the total power consumed was lower in this PPI. As next section will show, the execution time and the energy consumption are higher than OpenMP. A low power consumption means that the benchmark consumed less energy over time, but that time was higher than OpenMP. It means that PThreads has a lower overhead caused by parallelization over OpenMP. In fact, for all memory-bound benchmarks, OpenMP uses about two and three times more memory than PThreads [9].

On the other hand, PThreads have approximately ten times more cache misses than other PPIs in memory-bound benchmarks [9]. In this way, the execution of PThreads takes more time, but the use of hardware in this period is less intense about the other PPIs, which implies in lower consumption of energy over time. This increase in execution time can be caused by busy waiting for PThreads.

JA and GS with OpenMP reach a high energy consumption with MPI. They are memory-bound benchmarks, so the overhead of communication and synchronization among threads begins to impact negatively. With MPI-1 and MPI-2, the results indicates that, despite the total energy, the power consumption are very similar to the results of CPU-bound benchmarks. The growth of power consumption as the number of parallel processes increases follows the same pattern previously observed. It is perceived that MPI-2 has a lower consumption than MPI-1 in most cases for both CPU and memory-bound. This small di erence may be caused by dynamic process creation. This causes processes to be created later in MPI-2.

Another observed factor is that PThreads access less the memory system during synchronization. This means that for memory-bound programs parallelized using PThreads, this processor we used is a good choice since it provides considerable performance improvements at the same price in energy consumption. For CPU-bound programs, the power consumption for each PPI is very similar. The impact of particular characteristics of each communication model on the memory system is reduced as the benchmarks use more CPU.

4.2 Performance

Regarding performance, the benchmark showed similar behavior for the three governors across all CPU-bound benchmarks, and the same occurred for memory-bound benchmarks. Thus, we only present representative tests from HA, TR, and MM using twelve parallel threads/processes for the sake of space. They represent CPU-Bound (CPU-B), memory-bound (WMEM-B), and weakly memory-bound (WMEM-B) benchmarks, respectively. The results are presented in Figure 3.

In CPU-bound benchmarks the governor `performace` achieved a maximum performance of only 3% over the `ondemand`. That happens because these benchmarks will run most of the time on the CPU. It means that the `ondemand` governor can perform well because it will operate most of the time at the highest frequency. Regarding the `powersave` governor, it is possible to draw a relationship with the result of power and energy. The `powersave` consumed about three times the energy compared to other governors. The same behavior occurs the other way around with the execution time, which shows that `powersave` spends approximately the triple of the time running CPU-bound benchmarks.

The memory-bound benchmarks are represented by TR (MEM-B on Figure 3). In this type of benchmark the `ondemand` governor does not perform as closely as `performance` for PThreads, as seen in CPU-bound benchmarks. This shows the impact that is switching on frequency levels causes each time a benchmark goes into memory operations. This pattern is seen in weakly memory-bound benchmarks, as well. MPI-2 presents the same behavior of CPU-bound benchmarks, with `powersave` spending around the triple of time to compute. What causes this behavior is the need for communication among threads by TR. In MPI-2, these communications go through an inter-communicator that links the main process with the dynamically created ones. It increases the cost of each communication operation. TR is characterized as one of the benchmarks that do most communication operations [16]. Each process does one communication operation for each element of the input vector. In our test case, that means 96 thousand exchange data operations using a 2048×2048 input matrix and twelve threads. Therefore, all these characteristics together causes the `ondemand` to lose performance over the `performance` governor. Regarding threads, it impacts more in PThreads than OpenMP, which uses directives to improve communication. For MPI-2, the need for an inter-communicator also impacts negatively.

For the last, MM is representing weakly memory-bound benchmarks (WMEM-B on Figure 3). These benchmarks are weakly memory-bound because they spend most of their time on the CPU but still make a considerable amount of memory accesses. The other two benchmarks in this category are DJ and JA. DJ was not much a ected by the change of governors and showed similar behavior to the CPU-bound benchmarks. That is because the DJ looks for paths between nodes in a sparse matrix and does not have to read and write for each value. MM, on the other hand, accesses three arrays to do read and write operations, needing to do more accesses than DJ. Thus, the matrix access pattern in the benchmark source code shows that MM has an access pattern that increases cache misses (by changing the j and k indexes). The overhead presented by MPI-1 and MPI-2 in

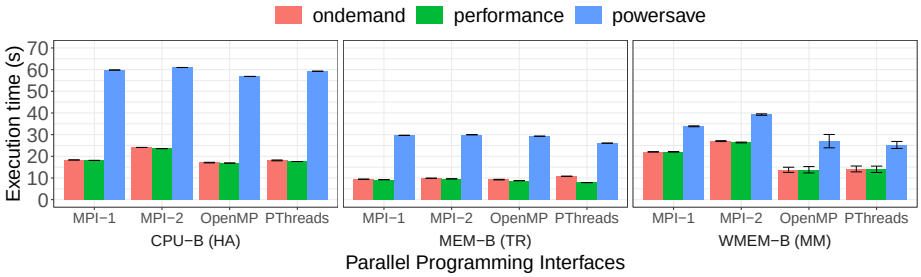


Fig. 3. Execution time using twelve parallel threads/processes.

this benchmark shows that the power governor did not increase execution time over the other two in MPI-2. In other PPIs, the di erence was also much smaller compared to that seen in CPU-bound benchmarks. Therefore MM's pattern of memory access has had a signi cant impact on execution time with `ondemand` and `performance` governors.

The performance results show that the amount of memory access and communication operations are the biggest in uences on the performance using di erent CPU governors. For benchmarks that do many memory-accesses, CPU governors do not de ne speci c behavior and vary depending on how memory-bound the benchmark is. While `ondemand` and `performance` achieved almost 70% performance over `powersave` in CPU-bound and memory-bound benchmarks, for the weakly memory-bound benchmarks this performance gain is around 30%. Thus, using frequency scaling only on the CPU may not make much di erence when using highly memory-bound benchmarks. To improve the performance of this type of benchmark through governors, it is necessary to balance the use of frequency scaling in memory. This scenario would probably change a lot if we used a larger supercomputing infrastructure. The MPI benchmarks would require to communicate through the network, which is a costly operation for the infrastructure.

The impact of frequency scaling governors is widely discussed. However, few studies evaluate the performance, power and energy consumption of parallel benchmarks using di erent governors.

Dzhagaryan and Milenkovic [7] evaluated how the number of threads and frequency scaling impact the energy consumption of multicore based infrastructure. The authors used PThreads benchmarks from the PARSEC benchmark for the experiments. They concluded that for an Intel® Xeon® processor 1240 v2 a frequency between 2.8 and 3.0 GHz gives the best trade-o between performance, power and energy consumption.

Jiang C. [14] attempted to nd a relationship between multicore processor frequency levels and application performance. The author tested a couple of

benchmarks (PI Calculation and File Compression) with di erent workloads. However, he did not evaluate parallel benchmarks, which reduces the contribution of evaluating a multicore based infrastructure.

Ibrahim et al. [12] investigated the impact of dynamically scaling the frequency of compute nodes on the performance and energy consumption of a Hadoop cluster infrastructure. They ran the PUMA benchmark and another couple of distributed programs using di erent governors. The paper does not exploit parallelism.

Catalan et al. [3] evaluated the energy e ciency of dense linear algebra routines using low-power multicore processors and analyzed whether the potential energy reduction achieved when scaling the processor to operate at a low voltage compensates the cost of integrating a fault tolerance mechanism. The authors used matrix-vector and matrix-matrix multiplication kernels using the BLIS framework. The authors did not exploit TLP or di erent PPIs.

Teng et al. [17] propose a set of algorithms that use compile-time information to achieve energy e ciency by frequency scaling control at run-time. The authors concluded that in a power-saving con guration, the memory-intensive benchmarks achieved better performance over CPU-intensive.

Chadha and Gerndt [4] implemented an energy-aware tuning plugin for DVFS based on a neural network. This neural network was trained using various OpenMP, MPI, and hybrid benchmarks. The authors did not exploit TLP or PPIs. In our work, we evaluate the impact of each application's characteristics, but we also assess the impact of di erent PPIs on power, energy and performance.

Oliveira et al. [2] proposed an automatic and non-intrusive framework to optimize parallel applications implemented with OpenMP, at the static time, by selecting the ideal number of threads and CPU frequency level to execute each parallel region. This framework consists of an optimization algorithm based on a genetic algorithm that optimizes the trade-o between performance and energy consumption. For the experiments, the authors used eight benchmarks parallelized with OpenMP, ve of them from the NAS Parallel Benchmarks. The paper does not exploit the impact of the characteristics of the benchmarks and evaluates only OpenMP.

Almatouq et al. [1] propose an optimization technique that balances performance and energy consumption by applying a joint control of core, resource, and frequency scaling. The technique was validated using benchmarks from the PARSEC benchmark. Although the benchmarks are evaluated individually, the authors did not exploit TLP or PPI characteristics.

It is assumed that execution time and energy consumption behave in a non-linear manner concerning frequency scaling. Based on that, Rauber and Rünger [21] proposed a scheduling process to independent tasks assignment and frequency scaling selection to improve e ciency. The experiments were done for the SPEC CPU benchmarks. The authors only exploited parallelism for independent sequential tasks.

Sheikh et al. [22] use genetic algorithms to nd the best trade-o s for energy, performance, and temperature using a DVFS-based algorithm. The experiments

were carried over five parallel benchmarks, and the authors evaluated the application characteristics and multiple TLP. However, they did not investigate the PPI impact on the computing infrastructure.

Marques et al. [6] investigated how multidimensional frequency scaling (CPU, RAM, and L2 cache) can improve Energy-Delay Product (EDP) in multicore embedded systems. They used nine parallel benchmarks from different benchmark suites, but do not mention any particular PPI.

Lorenzon et al. [16] ran the same benchmarks that make up the PAMPAR suite to find significant trade-offs between performance and energy in different architectures. The results showed that there is no single best case with higher performance and lower energy consumption. However, the author evaluated divided the benchmarks into CPU-bound and memory-bound and evaluated these two groups as if all the benchmarks were one, presenting unified results for each group. In this work, we evaluate each benchmark individually and also evaluate the impact of CPU governors.

All of the aforementioned related work evaluates the relationship between performance and power in multicore processors. However, the works which use parallel applications in experiments do not necessarily focus on exploring details of parallelism, such as varying the number of parallel threads or addressing a particular PPI. Nevertheless, none of them investigated the impact of using different PPIs. Also, some of them do not investigate the impact of each benchmark characteristics properly. In our work, we do an energy consumption assessment of 4 PPIs using 10 parallel benchmarks and varying the number of parallel threads/processes. We evaluated the impact of three different governors for frequency scaling and how these governors behaved according to the benchmarks and PPI characteristics.

In this paper, we evaluated the power and energy consumption and performance of different CPU frequency scaling techniques such as `ondemand`, `performance`, and `powersave` on multicore based computer infrastructure. We applied the techniques in the PAMPAR parallel benchmark, a set of benchmarks to evaluate the performance and energy consumption of PPIs. We show the power and energy consumption of 10 benchmarks written in PThreads, OpenMP, MPI-1, and MPI-2, popular PPIs in the Computational Science area, varying the number of threads/processes and the CPU governors.

Our experimental results showed that the power and energy consumption has an increasing rate proportional to the number of threads/processes used in parallel. Moreover, we demonstrated that `powersave` consumes the triple of energy and up to 43.1% less power than `performance` and `ondemand` governors. The `performance` governor consumes 9.7% more power than `ondemand` for CPU-bound and 27.8% for memory-bound benchmarks. Another important factor to observe is that the way each benchmark communicates in MPI could make the power and energy consumption to be higher in large computing infrastructure.

This impact shows up on the performance, as well. Benchmarks that do many memory-accesses or exchange data operations tend to reduce the performance gains with the `powersave` governor. Although these factors impact less on the `performance` and `powersave` governors, there is no trade-o between performance, energy, and power by using these governors because they are too strict for only one goal.

In the future, we intend to verify how the distribution of threads/processes to di erent cores and processors a ects our experiments. Experiments using more nodes and a more extensive distributed supercomputing infrastructure would be necessary to improve the analysis over the MPI PPI. We also consider evaluating real-world benchmarks and other PPIs such as Intel TBB or UPC, for instance, the NAS Benchmarks [11].

1. Almatouq, M.: Performance and Power Optimization for Multi-core Systems using Multi-level Scaling. Ph.D. thesis, University of California, Irvine (2019)
2. Cardoso De Oliveira, C., Lorenzon, A.F., Beck, A.C.S.: Automatic tuning tlp and dvfs for edp with a non-intrusive genetic algorithm framework. In: 2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC). pp. 146{153 (Nov 2018). https://doi.org/10.1109/SBESC.2018.00029
3. Catalan, S., Herrero, J.R., Quintana-Ort, E.S., Rodrguez-Sanchez, R.: Energy balance between voltage-frequency scaling and resilience for linear algebra routines on low-power multicore architectures. Parallel Computing 73, 28{39 (2018)
4. Chadha, M., Gerndt, M.: Modelling dvfs and ufs for region-based energy aware tuning of hpc applications. In: 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 805{814 (May 2019). https://doi.org/10.1109/IPDPS.2019.00089
5. Dongarra, J., Meuer, H., Strohmaier, E.: Green500 Supercomputer: June 2019 (June 2019), https://www.top500.org/green500/ , Accessed in: 28 Jun. 2019
6. dos Santos Marques, W., de Souza, P.S.S., Lorenzon, A.F., Beck, A.C.S., Beck Rutzig, M., Diniz Rossi, F.: Improving edp in multi-core embedded systems through multidimensional frequency scaling. In: 2017 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 1{4 (May 2017). https://doi.org/10.1109/ISCAS.2017.8050515
7. Dzhagaryan, A., Milenkovic, A.: Impact of thread and frequency scaling on performance and energy in modern multicores: a measurement-based study. In: Proceedings of the 2014 ACM Southeast Regional Conference. p. 14. ACM (2014)
8. Garcia, A.M.: Towards a Benchmark for Performance and Power Consumption Evaluation of Parallel Programming Interfaces. Master's thesis, Universidade Federal do Pampa (2016)
9. Garcia, A.M., Schepke, C., Girardi, A.: PAMPAR: A new parallel benchmark for performance and energy consumption evaluation. Concurrency and Computation: Practice and Experience (2019). https://doi.org/10.1002/cpe.5504, e5504 cpe.5504
10. Garcia, A.M., Schepke, C., Girardi, A.G.: A new parallel benchmark for performance evaluation and energy consumption. 13th International Meeting on High Performance Computing for Computational Science (VECPAR) (2018)

11. Griebler, D., Lo , J., Mencagli, G., Danelutto, M., Fernandes, L.G.: E cient nas benchmark kernels with c++ parallel programming. In: 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP). pp. 733{740 (Mar 2018)
12. Ibrahim, S., Phan, T.D., Carpen-Amarie, A., Chihoub, H.E., Moise, D., Antoniu, G.: Governing energy consumption in hadoop through cpu frequency scaling: An analysis. Future Generation Computer Systems 54, 219{232 (2016)
13. Intel: Intel Performance Counter Monitor - A better way to measure CPU utilization (2012), http://www.intel.com/software/pcm , Accessed in: 12 Mar. 2019
14. Jiang, C.: System level power characterization of multi-core computers with dynamic frequency scaling support. In: 2012 IEEE International Conference on Cluster Computing Workshops. pp. 73{79. IEEE (2012)
15. Le Sueur, E., Heiser, G.: Dynamic voltage and frequency scaling: The laws of diminishing returns. In: Proceedings of the 2010 international conference on Power aware computing and systems. pp. 1{8 (2010)
16. Lorenzon, A.F., Cera, M.C., Beck, A.C.S.: Performance and Energy Evaluation of Di erent Multi-Threading Interfaces in Embedded and General Purpose Systems. Journal of Signal Processing Systems 80(3), 295{307 (2014)
17. Lu, T., Pande, P.P., Shirazi, B.: A dynamic, compiler guided dvfs mechanism to achieve energy-e ciency in multi-core processors. Sustainable Computing: Informatics and Systems 12, 1{9 (2016)
18. Mittal, S.: A survey of techniques for improving energy e ciency in embedded computing systems. International Journal of Computer Aided Engineering and Technology 6(4), 440{459 (2014)
19. Ott, R.L., Longnecker, M.T.: An introduction to statistical methods and data analysis. Nelson Education (2015)
20. Rauber, T., Runger, G.: Parallel programming: For multicore and cluster systems. Springer Science & Business Media (2010)
21. Rauber, T., Runger, G.: A scheduling selection process for energy-e cient task execution on dvfs processors. Concurrency and Computation: Practice and Experience 31(19) (10 2019). https://doi.org/10.1002/cpe.5043
22. Sheikh, H.F., Ahmad, I., Arshad, S.A.: Performance, energy, and temperature enabled task scheduling using evolutionary techniques. Sustainable Computing: Informatics and Systems 22, 272 { 286 (2019). https://doi.org/10.1016/j.suscom.2017.10.002
23. Silveira, D.S., Moro, G.B., Cruz, E., Navaux, P.O., Schnorr, L.M., Bampi, S.: Energy consumption estimation in parallel applications: an analysis in real and theoretical models. In: XVII Simposio em Sistemas Computacionais de Alto Desempenho. pp. 134{145 (2016)
24. Solana, A.: Europe's greenest supercomputer: Why energy-e cient HPC is on the rise (2019), https://www.zdnet.com/article/europes-greenest-supercomputer-why-energy-efficient-hpc-is-on-the-rise/ , Accessed in: 14 Jun. 2019