Empirical power consumption characterization and energy aware scheduling in data centers

Jonathan Muraña

Universidad de la República, Facultad de Ingeniería,
Montevideo, Uruguay, 11300,
jmurana@fing.edu.uy

Abstract

Energy-efficient management is key to reduce operational cost and environmental contamination in modern data centers. Energy management and renewable energy utilization are strategies to optimize energy consumption in high-performance computing. In any case, understanding the power consumption behavior of physical servers in datacenter is fundamental to implement energy-aware policies effectively. These policies should deal with possible performance degradation of applications to ensure quality of service. This manuscript presents an empirical evaluation of power consumption for scientific computing applications in multicore systems. Three types of applications are studied, in single and combined executions on Intel and AMD servers, for evaluating the overall power consumption of each application. The main results indicate that power consumption behavior has a strong dependency with the type of application. Additional performance analysis shows that the best load of the server regarding energy efficiency depends on the type of the applications, with efficiency decreasing in heavily loaded situations. These results allow formulating models to characterize applications according to power consumption, efficiency, and resource sharing, which provide useful information for resource management and scheduling policies. Several scheduling strategies are evaluated using the proposed energy model over realistic scientific computing workloads. Results confirm that strategies that maximize host utilization provide the best energy efficiency.

Keywords: Green Computing, Energy Efficiency, Multicores, Energy model, Cloud Simulation.

1 Introduction

Data centers are key infrastructures for developing and executing industrial and scientific applications. In the last decade, data centers have become highly popular for providing storage, computing power, hosting, middleware software, and other information technology services, available to researchers with ubiquitous access [1]. Energy efficiency of data centers has become one of the main concerns in recent years, having a significant impact on monetary cost, environment, and guarantees for service-level agreements (SLA) [2].

The main sources of power consumption in data centers are the computational resources and the cooling system [3]. Regarding power consumption of the computational resources, several techniques for hardware and software optimization can be applied to improve energy efficiency.

For example, software characterization techniques [4], which are applied to determine features that are useful to analyze the software behavior. This behavior analysis is an input to analyze and improve power consumption [5].

Energy consumption models are useful synthetic tools for analyzing issues related to energy efficiency of computing infrastructures, e.g., by predicting the power consumption for a given workload. Information resulting from the analysis is often used by decision-makers to take technical, monetary, or environmental decisions regarding the computing infrastructure. Energy models are based in the relation between power consumption and resource utilization and can be classified as hardware- or software-centric [2].

Models that follow complex approaches, i.e., by modeling the contribution of every hardware component and/or using machine learning methods for prediction are accurate, but they demand significant design and implementation effort. On the other hand, simple models, i.e., just based on overall server consumption and CPU utilization, demand significantly lower design and implementation effort, and can produce fairly

accurate results. In the review of related work, the use of simple models has been predominant, especially until 2013 (see, for example Dayarathna et al. [2] and Kurowski et al. [6]).

Simulations are widely utilized for evaluating performance and energy models, schedule techniques, and others features of the execution of scientific workloads on real infrastructures [7]. Using simulations, identical scenarios can be executed several times in order to perform statistical analysis. Moreover, simulators avoid the direct utilization of expensive hardware, allow a significant reduction in the execution time of the experiments [8]. Due to the aforementioned reasons, cloud simulation is a key component of the research in the area. Many cloud simulators with different characteristics and specializations have been proposed in the literature [9]. CloudSim is a simulation toolkit widely used in the literature. This tool allows modeling and simulating cloud computing infrastructures, provisioning environments, and applications [10]. In this work, a custom version of CloudSim [11] is extended by including new energy models, built considering the empirical power consumption evaluation, and different allocation policies.

In this line of work, this research focuses on the characterization of power consumption for applications over nowadays multicore hardware used in scientific computing platforms. Such characterization is useful for studying and understanding energy efficiency in data centers and for designing energy efficient scheduling strategies. Three synthetic benchmarks are studied over two physical servers from a real High Performance Computing (HPC) platform, registering their power consumption with a power meter device. Furthermore, the experimental analysis studies the power consumption of different applications sharing a computing resource via simultaneous execution. The proposed study is very relevant for nowadays data centers and HPC infrastructures that execute many applications, with an associated impact on both the energy efficiency and the quality of service (QoS) offered to the users of the platform. Furthermore, new energy models are built by applying polynomial regression on the empirical data. The new models are applied in simulation of data centers operation for predicting the power consumption and several scheduling strategies are evaluated in simulations performed considering real servers and scientific computing workloads.

The initial questions that have guided the research include the following: What is the relationship between the server load and its power consumption? Is power consumption different if the task is intensive in different computing resources? In that case, is it possible to save energy by executing tasks in the same server? What type of task is it convenient to execute together? To what extent is it possible to combine tasks without losing energy efficiency due to performance degradation caused by resource usage conflicts? What is the behaviour of the power consumption at the critical utilization level (about 100% of server capacity)?

This research focuses on answering the aforementioned questions using an empirical approach. The experiments are designed to cover the domain of the problem and gather relevant information to process it in later stages using appropriate computational techniques. The main contributions of this work are:

1. A cutting-edge review of related works regarding power characterization, modeling, and energy-aware scheduling in cloud computing and supercomputing systems. Also, a thorough review of available cloud simulation tools considering advantages and disadvantages of each one, is presented.
2. An empirical study of power consumption of the three main computing resources that most contribute to power consumption utilization (CPU, memory, and disk). The study considers isolated and combined computing resources, and different levels of server load. Two high-end multi-core servers (AMD and Intel architectures) are analyzed.
3. An empirical study of performance degradation in multi-core servers regarding the server load and the computing resource type.
4. Several energy models built from experimental power consumption data using computational intelligence techniques. Each model considers AMD and Intel architectures. Also, relevant metrics to assess the quality of each model are presented.
5. Energy evaluation of six scheduling strategies, considering realistic workloads and the developed models, through simulations.
6. The research follows the reproducible/replicable paradigm by using a Jupyter Notebook that shows in a clear and understandable manner the data processing from raw data. The reproducible/replicable paradigm allows to reduce errors and add new experiments data quickly. Also, the results and claims can be verified or extended by other researchers.

This manuscript summarizes the research developed in the thesis “Empirical characterization and modeling of power consumption and energy aware scheduling in data centers” (Universidad de la República, Uruguay, 2019). Part of the thesis content was previously published in a journal article [12]. This manuscript includes new scientific content not previously published: i) two energy models (for AMD and Intel hosts) that consider three types of computing resource, ii) a comparative analysis between the power consumption of CPU-bound tasks and memory-bound tasks, iii) analysis of the improvements of scheduling strategies

regarding business-as-usual planning, iv) four unpublished graphic analysis (Figures 5, 18, 19, and 20), and v) three unpublished numeric reports (Tables 1, 19, and 18).

The manuscript is organized as follows. Section 2 reviews related works on energy characterization in multicores and simulators for energy-aware data centers. Section 3 presents the route map of the research. The different stages, their inputs and outcomes are introduced. Also, the proposed methodology for energy characterization, the benchmarks, and the physical setup for experiments, are described. The experimental evaluation of power consumption and performance of different benchmarks, is reported and discussed in Section 4. Section 5 describes the details of the power consumption models and performance models, built using polynomial regression. These models quality is assessed and compared by utilization of relevant metrics. In addition, several scheduling strategies for data centers, based on well-known heuristic are presented. The strategies were compared according to its energy efficiency through a simulation tool, using realistic workloads. Finally, the conclusions and main lines for future work are formulated in Section 6.

2 Related Work

This section presents the review of related literature, that includes two parts. The first one presents the revised works related to characterization and modeling of power consumption of servers and energy-aware scheduling. Then, the second part presents a comprehensive review of the state-of-art in relation to cloud simulation tools.

In relation to the power characterization, modeling and scheduling, Iturriaga et al. [13] studied the multiobjective optimization problem to optimize power consumption and execution time in heterogeneous computers systems, considering uncertainty. Specific versions of well-known heuristic were proposed for scheduling on realistic scenarios, applying the power consumption model defined in Nesmachnow et al. [14] and considering only CPU-bound workloads. A model for uncertainty on power consumption was determined through empirical evaluations using three CPU-bound benchmarks. Regarding scheduling results, online heuristics computed better schedules than offline approaches. Results also confirmed that uncertainty has a significant impact on the accuracy of the scheduling algorithms. The power consumption behavior of CPU-bound benchmarks shown by [13] is consistent with the one reported in our research. Moreover, we propose a fully empirical power consumption characterization, also considering two additional types of benchmarks: memory bound and disk bound.

Srikantaiah et al. [15] studied workload consolidation strategies for energy optimization in cloud computing systems. An empirical study of the relationship between power consumption, performance, and resource utilization was presented. The experiments were executed on four physical servers connected to a power meter to track the power consumption, and resource utilization was monitored using the Xperf toolkit. Only two resources were considered in the study: processor and disk. The performance degraded for high levels of disk utilization, and variations in CPU usage did not result in significant performance variations. Energy results were presented in terms of power consumption per transaction, resources utilization, and performance degradation. Results also showed that power consumption per transaction is more sensitive to CPU utilization than disk utilization. The authors proposed a heuristic method to solve a modified bin packing problem where the servers are bins and the computing resources are bin dimensions. Results reported for small scenarios showed that power consumption of the solutions computed by the heuristic is about 5% from the optimal solution. The tolerance for performance degradation was 20%.

Du Bois et al. [16] presented a framework for creating workloads with specific features, applied to compare energy efficiency in commercial systems. CPU-bound, memory-bound, and disk-bound benchmarks were executed on a power monitoring setup composed of an oscilloscope connected to the host and a logging machine to persist the data. Two commercial systems were studied: a high-end with AMD processors and a low-end with Intel processors. Benchmarks were executed independently, isolating the power consumption of each resource. Results confirmed that energy efficiency depends on the workload type. Comparatively, the high-end system had better results for the CPU-bound workload, the low-end system was better for disk-bound, and both had similar efficiency for the memory-bound workload. Our work complements this approach by including a study of the power consumption behavior when executing different types of tasks simultaneously on specific architectures for high performance computing.

Feng et al. [17] evaluated the energy efficiency of a high-end distributed system, with focus on scientific workloads. The authors proposed a power monitoring setup that allows isolating the power consumption of CPU, memory, and disk. The experimental analysis studied single node executions and distributed executions. In the single node experiments, results of executing a memory-bound benchmark showed that the total power consumption is distributed as follow: 35% corresponds to CPU, 16% to physical memory, and 7% to disk. The rest is consumed by the power supply, fans, network, and other components. Idle state represented 66% of the total power consumption. In distributed experiments, benchmarks that are intensive

in more than one computing resource were studied. Results showed that energy efficiency increased with the number of nodes used for execution.

Kurowski et al. [6] presented a data center simulator that allows specifying various energy models and management policies. Three types of theoretical energy models are proposed: i) *static approach*, which considers a unique power value by processing unit; ii) *dynamic approach*, which considers power levels, representing the usage of the processing unit; and iii) *application specific approach*, which considers usage of application resources to determine the power consumption. Simulation results were compared with empirical measurements over real hardware to validate the theoretical energy models in arbitrary scenarios. All models obtained accurate results (error was less than 10% with respect to empirical measurements), and the dynamic approach was the most precise. Langer et al. [18] studied energy efficiency of low voltage operations in manycore chips. Two scientific applications were considered for benchmarking over a multicore simulator. The performance model considered for a chip was $S = a_k(\sum f_i) + b_k$, where S are the instructions per cycle, f_i is the i -th core frequency and a_k, b_k depend on k , the number of cores in the chip. A similar model is used for power consumption. On 25 different chips, an optimization method based on integer linear programming achieved 26% in energy savings regarding to the power consumption of the faster configuration.

There are different opinions in the related literature about the importance of network power consumption. On the one hand, Feng et al. [17] measured the power consumption of each computing resource in isolated manner using different scientific computing benchmarks, and network turned to be in the fourth place in terms of power consumption, behind CPU, memory, and disk. Moreover, a recent survey by Dayarathna et al. [2] presented a graphic distribution of power usage by components, based on results from the analysis of a Google data center by Barroso et al. [19], where network power consumption is behind the other computing resources. A comparison between two Intel hosts (with Xeon and Atom processors), based on results from Malladi et al. [20], showed that the network power consumption is less than CPU, memory, and disk consumption. On the other hand, Totoni et al. [21] presented a runtime power management for saving energy by turning on/off not used or underutilized links during the execution of applications. The authors justified the relevance of the study, regarding the power consumption of the system, based on the use of low frequency many cores in computing systems and the complexity of network design nowadays. However, the study was not specifically aimed at analyzing scientific workloads.

Regarding cloud simulation, Kurowski et al. [6] proposed DCworms, an event-driven data center simulator written in Java, which allows defining performance models, energy models, and scheduling policies. DCworms is built using GSSIM, a Grid simulator by Bak et al. [8]. Among other features, GSSIM allows simulating a variety of entities and distributed architectures. Regarding to power consumption, GSSIM provides several energy models classified in *static*, *resource load* and *application specific*. In the third type of model, the user can specify energy profiles that allow modeling the type of application and the type of resource. Energy information is logged for analyzing the simulation. Calheiros et al. [10] introduced CloudSim, a cloud simulator developed in Java. Among the main advantages of CloudSim are the virtualization layer, which allows defining an abstraction of different execution environments of the applications in virtual machines, and the support provided for implementing federated clouds, which allows modeling large distributed systems. The simulator provides functions for implementing custom energy models. In addition, new scheduling policies can be included, for instance the strategies oriented to optimize the power consumption developed in our work. The total power consumption of a simulation can be used to compare the efficiency of scheduling policies. Kliazovich et al. [22] presented GreenCloud, a simulator oriented to energy-aware data centers. The GreenCloud design allows measuring the power consumption of each component of the infrastructure (host, switch, etc) in a detailed manner. A set of energy efficiency strategies are provided, including DVFS and dynamic shutdown of the computing components. These GreenCloud characteristics allow implementing fine-grained energy-aware scheduling strategies.

Núñez et al. [23] introduced Icancloud, a Cloud Simulator which allows reproducing Amazon EC2 instances types. Also, it is able to customize the VM brokering due to the flexible design of the hypervisor model. The hosts can be single core or multicore, and the software is able to simulate long time periods (years). Regarding storage, local devices, NFS, and parallel storage can be simulated.

Several works in literature have focused on modeling and characterizing power consumption of scientific applications. However, to the best of our knowledge, there is no empirical research focused on the inter-relationship between power consumption and CPU, memory, and disk utilization. Also, there is no experimental analysis of critical levels of resource utilization (close to 100%) and its impact on power consumption and performance. This work contributes to this line of research, proposing an empirical analysis for both issues mentioned above. In turn, the review allows identifying several desirable features of the available cloud simulators that are useful for the analysis reported in this research, including support for virtualization, single and multicore hosts, distributed systems, energy and performance models available, scheduling policies available, ease of use, and flexible customization. CloudSim includes many of the aforementioned

useful characteristics and also has a detailed documentation. For these reasons, we selected CloudSim as the simulator to use in the power consumption evaluation reported in this work.

3 Methodology for power consumption evaluation

This section describes the proposed methodology for power consumption evaluation, the benchmarks and architectures studied, the power evaluation setup and the experiments designed.

3.1 Overview of the proposed methodology

The experiments design was oriented to characterize the power consumption of the most important computing resources from the point of view of energy efficiency: CPU, memory and disk [19, 16, 17, 13, 20]. The importance of network power consumption has increased in the last years, mainly in modern data centers that offer distributed services, which heavily rely on network communications [24, 25]. However, network power consumption analysis was not included in the characterization, due to two main reasons: i) scientific computing applications in multicore architectures, on which the study of this research is centered, do not necessarily have a great use of the network. In these applications, the multithreaded paradigm allows solving complex scientific computing problems, without using the network extensively, since they mainly apply shared memory communication methods and ii) the complexity of the experiments, since to performing measures of network power consumption implies the consideration of several components such as switches, network interfaces, network topology, and cards, etc. Furthermore, the relevance of network power consumption is subject of debate, as we already acknowledged in the review of related literature in Section 2.

The main goal of the analysis was studying the holistic behavior of the power consumption of a host, considering the following elements, which are correlated: the utilization level of each resource; the total utilization level of the host; and the types of resources involved. In consequence, experiments were designed to execute synthetic benchmarks that make intensive utilization of different computing resources, which allow capturing the features of different scientific computing applications in multicore computers. Benchmarks were executed in both isolated and combined manner, at different levels of resource utilization. Analyzing the power consumption of hosts at levels close to maximum utilization (100%) has received low consideration in related bibliography, thus it is included as one of the main relevant issues to study. Thus, all type of experiments performed consider the critic level of utilization.

Since the downside of reducing energy consumption is the degradation of system performance, it is necessary to complement the characterization of energy consumption with performance experiments. In addition to the relationship between system load and power consumption, the relationship between system load and execution time is studied. This way, it is possible to make trade-off decisions according to the context demand, such as energy prices, service levels, etc. In performance experiments, the same benchmarks were executed with a fixed computational effort as stopping criterion, instead of the wall-time considered in power consumption experiments. For example, the criterion for stopping the CPU-intensive benchmark is when the loop counter to find prime numbers is greater than 20000.

3.2 Benchmarks for power consumption characterization

Simple benchmarks (i. e., benchmarks that are intensive in a single computer resource) were used for the analysis, in line with several articles reviewed in the literature related to the evaluation of power consumption. For example, Iturriaga et al. [13] considered a power consumption evaluation of LINPACK [26], a well-known CPU-intensive benchmark used for ranking supercomputers of the TOP500 list. The same benchmark were used by Kurowski et al. [6], together with Abinit [27], a software to calculate properties of material within density functional theory (which can be CPU or memory intensive depending on the configuration of parameters), NAMD [28], a CPU-intensive software for biomolecular system simulation, and CPUBURN [29], a software that allows stressing the CPU. The approach that uses simple benchmarks was followed since analysis is oriented to characterize power consumption regarding the utilization of each computing resource, both isolated and when combining the utilization of several resources. For this reason, specific programs with intensive utilization of each one of the three studied computing resources (CPU, memory, disk) were needed. The chosen synthetic benchmarks allow isolating the utilization of each resource, in order to perform the characterization.

A set of benchmarks included in the Sysbench toolkit [30] were used in the analysis and characterization of power consumption. Sysbench is a cross-platform software written in C that provides CPU, memory, and disk intensive benchmarks for performance evaluation. The components used in the experiments include a CPU-bound benchmark, a memory-bound benchmark, and a disk-bound benchmark. The main details of each benchmark are described next.

CPU-bound benchmark. The CPU-bound benchmark is an algorithm that calculates $\pi(n)$ (the prime counting function) using a backtracking method. The algorithm in the benchmark includes loops, square root, and module operations, as described in Algorithm 1.

Algorithm 1 CPU-bound benchmark program

```

1:  $c \leftarrow 3$ 
2: while  $c < \text{MaxPrime}$  do
3:    $t \leftarrow \text{sqrt}(c)$ 
4:    $l \leftarrow 2$ 
5:   while  $l < t$  do
6:     if  $c \pmod t = 0$  then
7:       break
8:     end if
9:     if  $l > t = 0$  then
10:       $n \leftarrow n + 1$ 
11:    end if
12:     $l \leftarrow l + 1$ 
13:  end while
14:   $c \leftarrow c + 1$ 
15: end while
16: return  $n$ 
```

Algorithm 2 Memory-bound benchmark program

```

1: while  $\text{buf} < \text{end}$  do
2:    $*\text{buf} \leftarrow \text{tmp}$ 
3:    $\text{buf} \leftarrow \text{buf} + 1$ 
4: end while
```

Algorithm 3 Disk-bound benchmark program

```

1: while  $\text{ReqCount} < \text{MaxReqs}$  do
2:    $\text{Req} \leftarrow \text{generate\_rnd\_request}()$ 
3:   if  $\text{is\_read}(\text{Req})$  then
4:      $\text{read}(\text{Req.file})$ 
5:   end if
6:   if  $\text{is\_write}(\text{Req})$  then
7:      $\text{write}(\text{Req.file})$ 
8:   end if
9:    $\text{ReqsCount}++$ 
10: end while
```

Memory-bound benchmark. The memory-bound benchmark is a program that executes write operations in memory, as described in Algorithm 2, where the *buf* variable is an array of integers. The cells of the array are overwritten with the value of *tmp* until the last position of the array, i.e., the value of the *end* variable.

Disk-bound benchmark. The disk-bound benchmark is a program that reads/writes content in files. Read or write requests are generated randomly and executed until a given number of requests (*MaxReqs*) is reached, as described in Algorithm 3.

3.3 Multicore hosts and power monitoring setup

Experiments were performed on high-end servers from Cluster FING, the HPC platform from Universidad de la República, Uruguay [31]. Two hosts were chosen according to their features and availability: HP Proliant DL385 G7 server (2 AMD Opteron 6172 CPUs, 12 cores each, 72 GB RAM), and HP Proliant DL380 G9 server (2 Intel Xeon E5-2643v3 CPUs, 12 cores each, 128 GB RAM). The considered hardware testbed covers the two most important architectures for high-end computers and high performance computing data centers nowadays. Table 1 presents the specification of both hosts.

Table 1: Specification of servers used in experiments

<i>feature</i>	<i>host server</i>	
	AMD	Intel
model name	AMD Opteron 6172	Intel Xeon CPU E5-2643 v3
architecture	x86_64	86_64
cores	24	24
CPU frequency	2.1 GHz	3.4 GHz
threads per core	1	1
cores per socket	12	12
sockets	2	2
NUMA nodes	4	2
total memory	70 GB	126 GB

To measure the power consumption of a server, two approaches are found in literature: software-based metering (i.e., in-band) and hardware-based metering (i.e., out-of-band) [2, 32]. Software-based meters estimate the power consumption of some server components by consulting internal counters [33]. An example of a software-based meter is *likwid* [34], which reads the counter Running Average Power Limit (RAPL) of Intel architectures. On the one hand, software-based approaches have low cost and high scalability and, on the other hand, the power measurement is an estimate and is partial, since the CPU counters do not consider the consumption of server components such as fans, motherboard and others. Besides, counters and sensors are not available in many high-end server models, so the software-based approaches are restricted to one

group of servers. Hardware-based approaches [35, 36] (the server is connected to an external power meter, which is connected to the power outlet) have the advantage that the power measurement is independent of the server model. Also, since the total energy consumption is considered, the measurements are accurate.

On the downside of hardware-based approaches, the economic cost is greater than the measurement of the software and can not be scaled, since including a new host for power monitoring involves buying and installing new power meters. In addition, power meters installation is not possible if physical access to the IT equipment is restricted.

Because this research focuses on the overall power consumption of the server and studies its holistic behavior, the use of external power meters is more appropriate than software-based meters. Figure 1 presents the power monitoring setup applied in the research developed in this research. The presented setup allows reducing the measurements noise, since the processing that is not related to the benchmark executions (polling, log writing, etc.), is executed in an external machine.

Figure 1: Power monitoring setup

The applied setup is similar to the one used in related works [13, 16, 15]. Benchmarks were executed in a host connected to the power source via a Power Distribution Unit (PDU) to register the instant power consumption. The PDU used is CyberPower PDU20SWHVIEC8FNET model. The PDU allows accessing the power consumption indicators through a web service. In an secondary machine, a polling demon logged data for post-processing.

3.4 Design of experiments

In order to consider the general remarks of the key steps in the design of experiments suggested by [37], the design of the experiments in this work is a consequence of initial questions about the factors that affect the host PC of multi-cores (see Introduction 1). In the experiments, the average power consumption (PC) of each host was computed considering the measures obtained in 20 independent executions of each benchmark (in single benchmark experiments) or combination of benchmarks (in the combined benchmarks experiments), to obtain statistically significant values. Average and standard deviation of PC are reported for each benchmark in the experimental analysis. The average idle power consumption (IC), i.e., the average consumption of the host when it is not executing any workload, is computed considering the measures for 20 independent executions of a program with null operations. Both PC and IC are used to compute the effective consumption (EC) for a given benchmark or workload as $EC = PC - IC$.

In order to consider an incremental approach to the experiment design that facilitates understanding, the experiments are described in three stages: single experiments, combined experiments and performance experiments. The stages are described in the next paragraphs.

First stage: single benchmarks. In a first set of experiments, benchmarks were evaluated isolated (i.e., independently from each other, analyzing only one resource). Utilization level (UL) is defined as the percentage of processors being used regarding the total number of processors in the host.

Eight ULs were considered for single benchmark experiments: 12%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5%, and 100%. Figure 2a shows an example of 37.5% UL: the host has 24 processors of which 9 execute instances

of the CPU-bound benchmark. The remaining processors are in *idle* state.

Second stage: combined benchmarks. In a second stage, the simultaneous execution of benchmarks was evaluated, analyzing several combinations of resource utilization at the same time. Two and three benchmarks were executed together in different ULs. In this case, UL is a vector where each entry represents the percentage of processors being used by each type of benchmark considered in the study (CPU-bound, memory-bound, and disk-bound, in that order). The chosen ULs were (25%,25%), (25%,50%), (25%,75%), (50%,25%), (50%,50%), (75%,25%) in pair executions, and (25%,25%,25%), (25%,25%,50%), (25%,50%,25%), (50%,25%,25%) in triple executions. The considered ULs cover the domain of possible combinations with a suitable granularity. Figure 2b shows CPU-bound and memory-bound executing together with UL (25%,50%) and Figure 2c shows the three benchmarks executing combined with UL (25%,25%,50%).

Each instance of the memory-bound benchmark was configured to use $(100/N)$ percent of the available memory, where N is the number of processors of the host. This configuration allows using 100% of the memory in full utilization mode. Each instance of the disk benchmark was configured to use 4 GB of disk size in AMD experiments and 2 GB of disk size in Intel experiments. These disk sizes were chosen taking into account the available disk capacity in each host. Instances were executed and monitored for 60 seconds. These duration allows ensuring the steady state of the benchmark execution, avoiding biasing the measurements to the initialization stages.

Figure 2: Examples of single and combined utilization levels

Third stage: performance evaluation. Finally, the impact on performance was analyzed. One of the most relevant metrics in performance analysis of applications is the makespan, defined as the time spent from the moment when the first task in a batch or bag of tasks begins execution to the moment when the last task is completed [38]. In performance evaluation experiments, the makespan when executing multiple applications at the same time is compared with the makespan of single executions. In all cases, the average makespan values computed over 20 independent executions of each benchmark or combination of benchmarks are reported, in order to provide statistical significance.

Summarizing, the design of experiments aims at studying the factors that affect the power consumption of servers, such as server load and the type of computing resources used by tasks, among others. The independent variable is the UL, i.e., the percentage of processors used with respect to the total number of processors on the server. In combined experiments UL is a vector where each entry represents the percentage of processors being used by each type of benchmark. In addition, in power experiments the dependent variable is the power consumption of the server. In turn, in performance experiments the dependent variable is the completion time (makespan) of the bag-of-tasks job.

4 Power and performance evaluation results

This section presents the results obtained in experiments that studied the power consumption. Details of idle power consumption evaluation are presented in Subsection 4.1. Subsection 4.2 shows and discusses the

results of the power consumption evaluation considering benchmarks execution independently. Then, the results of combined benchmark executions are presented in Subsection 4.3.

4.1 Idle power consumption evaluation

The average idle power consumption in both AMD and Intel hosts was calculated by performing 20 independent executions of a null program, i.e, a program that executes a sleep function for 60 seconds. The number of independent executions was chosen to obtain results with statistical validity. The average values obtained for idle power consumption ($\pm$ standard deviation) were 183.4 ± 1.3 W for the AMD host and 57.0 ± 0.9 W for the Intel host. These average values are considered as the idle power consumption, of the corresponding host, in all the experiments reported in this section. For example, if the measured overall consumption of one experiment in Intel host was 200 W, the effective power consumption of the experiment was calculated as 200 W less 57 W, this is, 143 W.

4.2 Results of single benchmark executions

CPU-bound benchmark. Table 2 reports PC and EC values for the CPU-bound benchmark for Intel and AMD hosts. The PC is reported as the average more less the standard deviation of the independent executions of the experiment. The PC results indicate that the AMD host demands more power than the Intel host for CPU-bound workload. The PC difference between hosts is in the interval [68 W, 73 W], which means that the AMD host consumes approximately 60% more power than the Intel host. However, it is not possible to conclude about the hosts energy efficiency without a performance analysis.

According to peak PC reported by Table 2 (264.1 W on AMD and 194.6 W, that occurs at UL 100%), the IC represents the 69% on AMD and 29% on Intel, of the maximum power consumption.

Table 2: PC and EC results for the CPU-bound benchmark on AMD and Intel hosts

<i>AMD</i>			<i>Intel</i>		
<i>UL</i>	<i>PC</i>	<i>EC</i>	<i>UL</i>	<i>PC</i>	<i>EC</i>
12.5%	194.4 ± 0.5	11.0	12.5%	121.6 ± 1.7	64.6
25.0%	205.0 ± 0.7	21.6	25.0%	136.6 ± 2.0	79.6
37.5%	215.1 ± 0.5	31.7	37.5%	142.5 ± 2.1	85.5
50.0%	225.1 ± 0.8	41.7	50.0%	152.8 ± 2.9	95.8
62.5%	235.5 ± 1.6	52.1	62.5%	163.2 ± 2.7	106.2
75.0%	246.0 ± 1.4	62.6	75.0%	175.8 ± 1.8	118.8
87.5%	254.7 ± 1.0	71.3	87.5%	185.8 ± 2.3	128.8
100.0%	264.1 ± 1.8	80.7	100.0%	194.6 ± 4.3	137.6

Figure 3: EC comparison for the CPU-bound benchmark on AMD and Intel hosts

Figure 3 presents a comparison of EC values in both hosts, which shows an average EC difference of 56 W between Intel and AMD hosts for all ULs. However, this difference is not relevant in order to evaluate which of both host consumes more power, because the absolute value of EC is not representative of the overall power consumption of the host. In the graphic analysis, the relevant indicator is the variation of EC regarding the UL and also the difference (of the EC variation) between hosts (i.e., a comparison of its derivatives). The almost linear behavior of EC in Figure 3 indicates that EC is proportional to the UL. Furthermore, curves for Intel and AMD are almost parallel, indicating that the UL of CPU-bound applications has a similar behavior in both hosts. EC on Intel host shows a remarkable increase when moving from 0 to 12.5% UL, which is not observed for the other ULs. This increase is explained by the dynamic handling of chip power, awaking, or increasing voltage of its components according to usage demand, which is notorious on Intel architecture. In critic UL (%100), it is not observed any different behavior regarding the power consumption, which allows concluding that in CPU-bound workloads, resource conflict does not implies an increase in power consumption.

Memory-bound benchmark. Table 3 reports the PC and EC values for the memory-bound benchmark for Intel and AMD hosts. The comparison between PC values of memory experiments and the CPU (presented above) allows concluding that memory use has impact in power consumption, because at same level

and host, the power consumption of memory experiments is greater than CPU experiments. As seen in the CPU experiments, the AMD host consumes more power than Intel. The biggest difference in PC (89.6 W) occurs at the minimum UL (12%) and the lowest (45.7 W) occurs at 100%, indicating that Intel has an efficient management of the energy with respect to the use memory.

Table 3: PC and EC results for the memory-bound benchmark on AMD and Intel hosts

<i>AMD</i>			<i>Intel</i>		
<i>UL</i>	<i>PC</i>	<i>EC</i>	<i>UL</i>	<i>PC</i>	<i>EC</i>
12.5%	215.5±1.8	32.1	12.5%	125.9±6.4	68.9
25.0%	238.7±1.1	55.3	25.0%	156.3±4.8	99.3
37.5%	257.7±1.9	74.3	37.5%	178.7±4.8	121.7
50.0%	272.3±2.7	88.9	50.0%	191.8±6.8	134.8
62.5%	278.5±6.4	95.1	62.5%	213.0±3.9	156.0
75.0%	279.9±5.9	96.5	75.0%	225.1±5.4	168.1
87.5%	290.8±5.6	107.4	87.5%	239.2±4.0	182.2
100.0%	294.7±3.0	111.3	100.0%	249.0±9.3	192.0

Figure 4: EC comparison for the CPU-memory benchmark on AMD and Intel hosts

Figure 4 presents a graphic comparison of the EC values in both hosts. Results show a significant increase of EC with regard to CPU-bound executions for all ULs (104% for the AMD host and 36% for the Intel host, on average). A logarithmic behavior is observed for both PC and EC, which does not occur in CPU-bound case. This behavior may be mainly due to the bottleneck in the access to the main memory that reduces the CPU usage. No significant increase is detected on high/critical ULs, possibly by effective resource contention by the operating system for solving conflicts over access to shared resources.

In order to analyze the PC difference between CPU and memory bound workload, Figure 5 presents ΔPC , defined as the additional percentage of power that memory experiments consumes regarding CPU experiments (i.e., $\Delta PC = (PC_{MEM} - PC_{CPU}) \times 100 / PC_{CPU}$). Since the CPU-bound benchmark is designed to consume more CPU cycles than the memory-bound benchmark, ΔPC is related to the power consumption of the memory usage. The graphic comparison of ΔPC allows observing a remarkable increase of ΔPC on Intel at medium and high ULs, when compared with low UL of the same host. On AMD, there is not a notorious ΔPC increment. This difference between hosts suggests that the power management on Intel architecture is better than on AMD architecture, since the power consumption increases according to the percentage of used memory (the percentage of the system memory used by the benchmark is equal to the UL, for example, 50% of the host total memory is used at UL 50%).

Figure 5: PC difference between of CPU-bound and memory-bound experiments

Disk-bound benchmark. Table 4 reports PC and EC values for the disk-bound benchmark. The PC is almost constant and notoriously less than CPU and memory experiments. Figure 6 presents a comparison of EC values in both hosts. The maximum EC variation through ULs is 4W in Intel and 2W in AMD. The reported low power variations indicate that disk usage has low impact in power consumption in comparison with CPU and memory.

Table 4: PC and EC results for the disk-bound benchmark on AMD and Intel hosts

<i>AMD</i>			<i>Intel</i>		
<i>UL</i>	<i>PC</i>	<i>EC</i>	<i>UL</i>	<i>PC</i>	<i>EC</i>
12.5%	188.1±0.4	4.7	12.5%	67.5±0.9	10.5
25.0%	189.3±0.5	5.9	25.0%	68.0±0.6	11.0
37.5%	189.5±0.6	6.1	37.5%	67.7±0.6	10.7
50.0%	189.3±0.3	5.9	50.0%	68.9±0.7	11.9
62.5%	189.3±0.3	6.5	62.5%	70.5±0.7	13.5
75.0%	190.1±0.4	6.7	75.0%	70.7±0.8	13.7
87.5%	190.7±1.6	7.3	87.5%	71.7±0.6	14.7
100.0%	190.4±0.6	7.0	100.0%	72.4±1.0	15.4

Figure 6: EC comparison for the disk-memory benchmark on AMD and Intel hosts

4.3 Results of combined benchmark executions

CPU- and memory-bound benchmarks. Table 5 reports PC and EC for the simultaneous execution of the CPU- and memory-bound benchmarks on AMD host. The results shows that the peak PC on AMD is 312.7 W and it occurs at UL (25%,75%). This peak is the maximum PC on AMD host registered considering all experiments in this research, indicating that the IC on AMD (183.4W) represents 59% of the total PC of the host.

Table 5: PC and EC results for the CPU- and memory-bound benchmarks combined on AMD host

<i>UL</i> (CPU,memory)	<i>PC</i>	<i>EC</i>	ΔEC
(25%,25%)	259.4±2.7	76.0 =	0.77
(25%,50%)	290.3±1.6	106.9 ↓	3.59
(25%,75%)	312.7±3.3	129.3 ↑	-11.30
(50%,25%)	277.1±2.2	93.7 ↓	3.21
(50%,50%)	310.3±2.6	126.9 ↓	3.69
(75%,25%)	295.8±6.6	112.4 ↓	5.53

Figure 7: Combined CPU- and memory-bound EC on AMD host

Symbol $\uparrow$ indicates that the EC of the combined benchmarks is higher than the sum of the ECs of each benchmark executed independently, i.e., the combined execution is less efficient than the independent execution. Symbol $\downarrow$ indicates the opposite, that is, the combined execution is more efficient. Symbol $=$ indicates that the values are equal, considering a threshold of 1 W of difference. Column ΔEC reports the difference between the EC of the combined execution and the sum of the EC of the independent executions (in W). The ΔEC values reported in Table 5 shows that the combined execution on AMD host allows reduce the EC compared to independent executions. In UL with high memory use, the independent executions presents lower EC values that combined. The 3D graph on Figure 7 presents the EC values of the combined execution of CPU and memory on AMD host. The color scale of the graph allows observing that the dark sectors, which correspond to a high EC, are accentuated near the memory axis.

Table 6 reports PC and EC for the simultaneous execution of the CPU- and memory-bound benchmarks

on Intel host. The peak PC (255.3 W) corresponds to UL (50%,50%), thus, the IC of the Intel host represents the 22% of the overall PC. The low percentage of IC is due to Intel host reduces the IC when the host is not used (at UL almost 0%), however, in other ULs the IC increases (as is observed and discussed in Section 3, a significant increment of PC occurs in the first UL, which does not occurs in later ULs). The combined executions of CPU and memory benchmarks in Intel host achieve the best results with regard to EC in pair combined experiments. In this case, a significant gain (around 47 W) is observed for all ULs. The highest gain occurs when CPU use is low and memory use is low, simultaneously.

Table 6: PC and EC results for the CPU- and memory-bound benchmarks combined on Intel host

<i>UL</i> (CPU,memory)	<i>PC</i>	<i>EC</i>	ΔEC
(25%,25%)	178.2 $\pm$ 5.0	121.2 $\downarrow$	57.65
(25%,50%)	226.2 $\pm$ 3.6	169.2 $\downarrow$	45.10
(25%,75%)	254.5 $\pm$ 5.6	197.5 $\downarrow$	50.08
(50%,25%)	199.9 $\pm$ 3.5	142.9 $\downarrow$	52.23
(50%,50%)	255.3 $\pm$ 3.2	198.3 $\downarrow$	32.28
(75%,25%)	225.0 $\pm$ 3.2	168.0 $\downarrow$	50.09

Figure 8: Combined CPU- and memory-bound EC on Intel host

The 3D graph on Figure 8 reports EC values of the combined execution of CPU and memory on AMD host. The graph allows observing the difference in derivative of UL 0 and the other ULs and the dark sectors near memory axis. Results of combined CPU and memory benchmarks show that for Intel host, combined executions reduce EC compared to single executions. In AMD host, however, combined executions of these types of benchmarks have not such notorious improvement when compared with single executions.

CPU- and disk-bound benchmarks. Table 7 reports PC and EC values for the simultaneous execution of the CPU- and disk-bound benchmarks on AMD host. ΔEC values indicate that the combined execution has not a significant gain or loss regarding independent execution.

Table 7: PC and EC results for the CPU- and disk-bound benchmarks combined on AMD host

<i>UL</i> (CPU,memory)	<i>PC</i>	<i>EC</i>	ΔEC
(25%,25%)	210.0 $\pm$ 1.1	26.6 =	0.83
(25%,50%)	211.5 $\pm$ 1.3	28.1 =	-0.67
(25%,75%)	212.4 $\pm$ 1.2	29.0 =	-0.71
(50%,25%)	229.8 $\pm$ 2.6	46.4 $\downarrow$	1.14
(50%,50%)	234.3 $\pm$ 4.9	50.9 $\uparrow$	-3.38
(75%,25%)	248.4 $\pm$ 3.5	65.0 $\downarrow$	3.49

Figure 9: Combined CPU- and disk-bound EC on AMD host

The 3D graph on Figure 9 presents EC values on AMD host, for different combinations of CPU and disk ULs. The graph shows that EC values only change significantly in the direction of the CPU axis, which indicates that the disk-bound load has not significant impact on EC.

Table 8 reports PC and EC values for the simultaneous execution of the CPU- and disk-bound benchmarks on Intel host. Results show an average gain of 10 W of combination regarding to independent execution. The maximum gain is 13.06 W and it occurs at UL (25%,75%). The 3D graph on Figure 10 presents EC values on Intel host, for different combinations of CPU and disk ULs. A similar result to the one obtained in AMD is observed, as EC changes significantly only in CPU axis direction. The aforementioned remarkable EC increase when moving from 0 to the next UL is notorious on CPU axis direction, but not on disk axis direction. This results indicate that the increment is related to CPU use.

Table 8: PC and EC results for the CPU- and disk-bound benchmarks combined on Intel host

<i>UL</i> (CPU,memory)	<i>PC</i>	<i>EC</i>	ΔEC
(25%,25%)	136.4 $\pm$ 2.3	79.4 $\downarrow$	11.19
(25%,50%)	139.0 $\pm$ 1.9	82.0 $\downarrow$	9.40
(25%,75%)	137.2 $\pm$ 3.2	80.2 $\downarrow$	13.06
(50%,25%)	154.2 $\pm$ 2.6	97.2 $\downarrow$	9.64
(50%,50%)	156.5 $\pm$ 2.8	99.5 $\downarrow$	8.19
(75%,25%)	177.8 $\pm$ 2.9	120.8 $\downarrow$	8.96

Figure 10: Combined CPU- and disk-bound EC on Intel host

Memory- and disk-bound benchmarks. Table 9 reports PC and EC values for the simultaneous execution of the memory- and disk-bound benchmarks on AMD host. The combined executions consume less energy at ULs (50%,25%) and (50%,50%), i.e., when memory-bound load is 50%. In the other ULs, there are not significant improvements on power consumption.

Table 9: PC and EC results for the memory- and disk-bound benchmarks combined on AMD host

<i>UL</i> (CPU,memory)	<i>PC</i>	<i>EC</i>	ΔEC
(25%,25%)	243.9 $\pm$ 1.5	60.05 =	0.61
(25%,50%)	241.8 $\pm$ 5.5	58.4 $\downarrow$	2.81
(25%,75%)	245.9 $\pm$ 2.0	62.5 =	-0.54
(50%,25%)	268.2 $\pm$ 9.1	84.8 $\downarrow$	10.01
(50%,50%)	269.5 $\pm$ 12.9	86.1 $\downarrow$	8.71
(75%,25%)	287.0 $\pm$ 4.0	103.6 $\uparrow$	-1.19

Figure 11: Combined memory- and disk-bound EC on AMD host

The 3D graph on Figure 11 presents EC values on AMD host for memory and disk-bound benchmarks executing in combination. The graph allows observing the same behaviour noted in the analysis of Figure 10 regarding the low increment of EC in disk axis direction, which confirms the almost negligible power consumption of disk-bound benchmark.

Table 10 reports PC and EC values for the simultaneous execution of the memory- and disk-bound benchmarks on Intel host. The results show that single executions consume less EC than combined, except for those experiments with high load of the memory-bound benchmark. The 3D graph on Figure 12 presents EC values on Intel host for memory and disk-bound benchmarks executing in combination. A similar results to the one obtained in AMD is observed, where the disk-bound benchmark has not impact in EC.

Table 10: PC and EC results for the memory- and disk-bound benchmarks combined on Intel host

<i>UL</i> (CPU,memory)	<i>PC</i>	<i>EC</i>	ΔEC
(25%,25%)	174.1 $\pm$ 8.3	117.1 $\uparrow$	-6.70
(25%,50%)	176.2 $\pm$ 5.9	119.2 $\uparrow$	-8.01
(25%,75%)	172.0 $\pm$ 7.6	115.0 $\uparrow$	-2.00
(50%,25%)	206.4 $\pm$ 6.0	149.4 $\uparrow$	-3.66
(50%,50%)	208.6 $\pm$ 4.2	151.6 $\uparrow$	-4.93
(75%,25%)	228.5 $\pm$ 5.0	171.5 $\downarrow$	7.51

Figure 12: Combined memory- and disk-bound EC on Intel host

CPU-, memory-, and disk-bound benchmarks. Table 11 reports the PC and EC values obtained when executing the CPU-, memory- and disk-bound benchmarks combined, on AMD and Intel hosts.

Table 11: PC and EC results for the CPU-, memory- and disk-bound benchmarks combined

<i>UL</i> (CPU,mem.,disk)	<i>AMD</i>		<i>Intel</i>	
	<i>PC</i>	<i>EC</i>	<i>PC</i>	<i>EC</i>
(25%,25%,25%)	266.3±4.5	82.9 =	177.4±4.1	120.4 ↓
(25%,25%,50%)	266.9±4.7	83.5 =	178.7±3.5	121.7 ↓
(25%,50%,25%)	303.2±3.3	119.8 ↑	220.9±5.5	163.9 ↓
(50%,25%,25%)	288.1±1.7	104.7 ↑	195.0±5.0	138.0 ↓

Results in Table 11 show that the combined execution on the AMD host has a higher EC compared to their independent execution in ULs (25%,50%,25%) and (50%,25%,25%). This behavior indicates that the three-combined execution is not efficient for high load of CPU-bound and memory-bound benchmarks. However, on Intel host, combined executions reduce EC compared to independent executions for all ULs. For independent executions on Intel host, the difference on PC between consecutive ULs is larger at the first UL (12.5%). Because of this difference, combined executions achieve better efficiency regarding independent execution on Intel host.

4.4 Performance evaluation

This subsection analyzes the performance evaluation experiments performed for each benchmark.

Table 12 reports the makespan of the CPU-bound benchmark and Figure 13 presents a graphic comparison for both hosts.

Table 12: Makespan results for the CPU-bound benchmark on Intel and AMD hosts

<i>AMD</i>		<i>Intel</i>	
<i>UL</i>	<i>makespan</i>	<i>UL</i>	<i>makespan</i>
12.5%	44.4±0.5	12.5%	35.8±8.6
25.0%	44.9±0.4	25.0%	30.4±5.7
37.5%	45.0±0.0	37.5%	36.2±9.7
50.0%	45.0±0.0	50.0%	30.8±3.7
62.5%	45.0±0.0	62.5%	29.5±0.6
75.0%	45.0±0.0	75.0%	29.9±0.8
87.5%	45.0±0.0	87.5%	34.0±9.4
100.0%	50.5±4.5	100.0%	47.4±11.0

Figure 13: Makespan comparison for the CPU-bound benchmark on Intel and AMD

Results show that increasing the UL does not impact significantly the completion time, due to the absence of resource competition. However, both hosts present a slight degradation for UL 100%, possibly due to conflicts with operating system processes.

Table 13 reports the makespan of the memory-bound benchmark and Figure 14 presents a graphic comparison for both hosts. Performance degrades in AMD; there is a gap of 400 seconds between the lowest and the highest UL. For Intel the difference is only 48 seconds. The difference in gaps is possibly explained by the specific memory features of each host, such as cache size and transfer speed: the Intel host processor has a significantly larger L3 cache size than the AMD host processor (20MB vs 12MB) and a faster bus clock speed (4.800 MHz vs 3.200 MHz).

Table 13: Makespan results for the memory-bound benchmark on Intel and AMD hosts

<i>AMD</i>		<i>Intel</i>	
<i>UL</i>	<i>makespan</i>	<i>UL</i>	<i>makespan</i>
12.5%	90.2±4.7	12.5%	34.6±4.7
25.0%	136.8±7.4	25.0%	26.6±0.9
37.5%	179.8±14.2	37.5%	41.1±2.2
50.0%	220.8±12.3	50.0%	41.8±1.1
62.5%	259.1±12.9	62.5%	67.5±4.3
75.0%	326.7±11.4	75.0%	62.2±1.7
87.5%	408.2±14.6	87.5%	74.5±1.5
100.0%	490.6±16.2	100.0%	82.5±2.0

Table 14 reports the makespan of the disk-bound benchmark and Figure 15 presents a graphic comparison for both hosts. The disk-bound case presents a significant degradation in performance when increasing UL when compared with other benchmarks, because of concurrency has a bigger impact in the disk access than memory access and CPU access.

Table 14: Makespan results for the disk-bound benchmark on Intel and AMD hosts

<i>AMD</i>		<i>Intel</i>	
<i>UL</i>	<i>makespan</i>	<i>UL</i>	<i>makespan</i>
12.5%	234.7±5.0	12.5%	36.0±0.9
25.0%	547.7±29.0	25.0%	74.0±1.5
37.5%	839.4±29.2	37.5%	114.7±2.4
50.0%	1397.8±184.5	50.0%	154.5±3.2
62.5%	1644.0±184.9	62.5%	195.9±3.8
75.0%	2006.3±182.7	75.0%	239.0±5.1
87.5%	2514.9±367.8	87.5%	279.1±25.8
100.0%	2571.5±147.2	100.0%	313.6±4.0

4.5 Energy efficiency analysis

This subsection analyzes the energy efficiency from the collected measurements. The energy efficiency metric (*eff*), defined in Equation 1, allows comparing the PC results for different ULs and hosts, while taking into account the execution time of an application. For comparing the energy efficiency of each host, the metric focuses on PC because it represents the real power consumption of each host. The lower the metric value, the higher energy efficiency of the host (lower power consumption and lower makespan). Results from the study indicate that the CPU-bound benchmark is more efficient at high ULs, the memory-bound benchmark is more efficient at medium ULs and the disk-bound benchmark is more efficient at low ULs. These results hold for both hosts. Overall, Intel host is more efficient than AMD for all ULs and all types of benchmarks. This observation is coherent with reported comparison of these processors [39], where Intel host processor is presented as 10 times more efficient than AMD host processor. However, our energy analysis is more comprehensive, because it considers more components of each host, not only the processor. Finally, the high-critic UL (100%) is less efficient than the high-medium UL (87.5%) in all cases, except for disk-bound benchmark executions. For disk-bound benchmarks there are small variations on the PC values between ULs. Thus, the proposed efficiency metric improves when the number of instances increase.

$$eff = \frac{PC \times makespan}{number\ of\ instances \times 3600} \quad (1)$$

Table 15 reports the average energy efficiency for all ULs and both hosts. The more efficient UL for a given host and type of benchmark is presented in bold.

Figure 14: Makespan comparison for the memory-bound benchmark on Intel and AMD

Figure 15: Makespan comparison for the disk-bound benchmark on Intel and AMD

Table 15: Efficiency for different ULs on AMD and Intel hosts

<i>UL</i>	<i>AMD</i>			<i>Intel</i>		
	<i>CPU</i>	<i>mem.</i>	<i>disk</i>	<i>CPU</i>	<i>mem.</i>	<i>disk</i>
12.5%	0.798	1.800	4.090	0.403	0.403	0.225
25.0%	0.426	1.510	4.800	0.192	0.192	0.233
37.5%	0.299	1.430	4.910	0.159	0.227	0.240
50.0%	0.234	1.390	6.130	0.109	0.185	0.246
62.5%	0.196	1.340	5.780	0.089	0.266	0.256
75.0%	0.171	1.410	5.890	0.081	0.216	0.265
87.5%	0.152	1.570	6.340	0.083	0.236	0.265
100.0%	0.154	1.670	5.670	0.107	0.238	0.263

4.6 Concluding remarks

The empirical study was aimed at extracting the characteristics of the power consumption of multi-cores and their relationship with the type of workload, considering all the components involved holistically. Single executions allowed concluding that EC vary in different ways depending on the computing resource considered. In particular, EC is directly proportional to the UL in the CPU-bound workload. For the memory-bound workload, the single executions showed a deceleration in EC variation as the UL increases.

The combined execution of different types of benchmarks indicated that EC reduces with respect to single executions, by tasks consolidation, taking advantage of the optimum UL (regarding EC) of computing resources. Results of combined experiments of two benchmarks showed a remarkable gain in Intel host, where the difference in EC reached 57.65 W at UL (25%, 25%). This gain represents the 32% of the EC of single executions. The three combined executions presented a EC gain on Intel host for all ULs regarding to single executions. However, on AMD host the EC is equal at (25%, 25%, 25%) and at (25%, 25%, 50%), and it is greater than EC of single executions at (50%, 25%, 25%) and at (25%, 50%, 25%). On the one hand, the performance analysis indicated that the CPU-bound workload showed not degradation as UL increases. On the other hand, results showed performance degradation of memory-bound and disk-bound workloads, specially in AMD host.

The energy efficiency study, which combines results of power consumption and performance experiments, showed that the optimum UL depends on the type of workload: CPU-bound workload presented best efficiency at high UL, memory-bound workload at medium UL, and disk-bound workload at low UL. The difference between host efficiency observed in results was consistent with the published specifications of the hosts. Numerical results of the presented power characterization can be used to build power consumption models. In addition, it is possible to consider the conclusions of the analysis as a guide to develop energy-aware scheduling strategies based on heuristics. Next section presents the proposed power consumption models, several scheduling strategies, and the simulations performed for their evaluation.

5 Energy model and simulation

This section presents the utilization of power and performance characterization results, by building a wide variety of models, which are used in simulations to evaluate energy aware scheduling strategies.

5.1 Energy model

According to the literature reviewed, power consumption models can be classified on three categories: *static*, where the consumption of the host is a constant value; *dynamic*, where the value of the power consumption depends of the utilization level of the host; and *application specific*, where the consumption is associated to the characteristics of each application [6]. Models with higher complexity can be built considering the competition for computing resources between those applications running at the same time in a host [40].

The proposed power consumption models are versions of specific application models. They are based on partitioning the applications into equivalence classes according their bounding computing resource (two applications are of the same class if they are intensive in the same computing resource). The identified variables that affect the energy consumption are: the computer resource, the load of the computer resource, and the total load of the host. These variables are synthesized in the definition of UL, an expression $(x_1, x_2, \dots, x_n)$. The proposed models seek the form of a function $f: [0,1]^n \rightarrow \mathbb{R}$, $f(x_1, x_2, \dots, x_n)$. The models built from independent executions focused in the effective consumption (EC) because it contains more synthetic information than power overall power consumption (PC), since PC is just EC plus a constant.

To build the energy models, experimental data were adjusted by polynomial regression [41]. Four functions were studied: linear, piece-wise, quadratic, and cubic. Two intervals were considered for the piece-wise model: $[0,50]$ and $[50,100]$. The complete data processing and modeling is available in a Jupyter Notebook in <https://www.fing.edu.uy/~jmurana/msc/>. The analysis follows the paradigm of reproducible research [42]. All raw data, processing tools, and processed results are provided for verification by researchers. Complete results and design decision are explained and the process of analysis is clearly documented, thus the published Jupyter Notebook allows reproducing the complete data processing and modeling, making easy to correct errors as well as extending the research by considering new experimental data and/or new models, without significant effort. This is an important contribution of our research, since the reproducible/replicable approach is not found often in related research areas [43].

Two relevant metrics to assess the quality of statistical models were applied to analyze the results: *coefficient of determination* (R-squared, R^2) and *adjusted R-squared* ($\bar{R}^2$). Both metrics evaluate the capabilities of the studied model to forecast future values in a given temporal series. $\bar{R}^2$ is an extension of R^2 proposed to avoid spurious increasing when using a larger number of independent variables [44].

Graphs in Figure 16 presents the four EC models for CPU-bound workload, on AMD and Intel hosts. Graph legends show the equations for each curve, where x is the independent variable, i.e. UL. The ancillary variable x' in the piece-wise model is zero when $x < 50$ and $x - 50$ in other case. A linear behavior of the models is observed. Besides, quadratic and cubic models have no-linear coefficients close to zero, which indicates the linear dependency too. Both hosts have a similar rate of increase. For example, in linear models, the slope is 0.8 in AMD host and 0.827 in Intel host. This similarity indicates that the proportional dependency between UL and EC is independent of the architecture, for CPU-bound workloads.

Figure 16: CPU energy models

Graphs in Figure 17 presents the EC models for memory-bound workloads on AMD and Intel hosts. The comparison of slopes of linear models shows that the EC on Intel increases faster than AMD, by a factor of 1.6 (1.368/0.846). The piece-wise graph for AMD shows a noticeable decrease of the first derivative in UL 50%, possibly due to the degradation of the performance caused by the waits in the access to memory, which implies less use of memory and, consequently, less consumption of energy.

Figure 17: Memory energy models

Results of the statistical analysis, reported in Tables 16 and 17, indicate that the cubic model provides the best fit for CPU on AMD host and memory on both host. Piecewise model provides the best values for CPU experiments on Intel host (values close to 1 means the better fit). According to the obtained results, the aforementioned cubic and piecewise models were considered for the simulations.

Equation 2 presents the PC model built applying the linear combination for CPU, memory and disk workload for AMD host, based on the best independent models. Since the independent models correspond to the EC modeling, the host IC is added in the PC models. In turn, Equation 3 presents the PC model built applying the linear combination for Intel host for CPU, memory, and disk workload. The ancillary variable α' is zero when $\alpha < 50$ and $\alpha - 50$ in other case. In both equations, α is the percentage of CPU-bound workload, β is the percentage of memory-bound workload, and γ is the percentage of disk-bound workload. The linear combination models assume that CPU-bound, memory-bound and disk-bound jobs arrive to the host (to be processed) with equal probability (0.33).

Table 16: Statistics for CPU energy models

Model	<i>AMD</i>		<i>Intel</i>	
	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.9031	0.9031	0.9749	0.9707
Piece	0.9925	0.9895	0.9929	0.9901
Quad	0.9843	0.9780	0.9966	0.9953
Cubic	0.9948	0.9909	0.9977	0.9960

Table 17: Statistics for memory energy models

Model	<i>AMD</i>		<i>Intel</i>	
	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.9993	0.9992	0.9958	0.9951
Piece	0.9996	0.9995	0.9959	0.9943
Quad	0.9997	0.9997	0.9958	0.99417
Cubic	0.9998	0.9997	0.9958	0.9927

$$\begin{aligned}
PC_{AMD} = & 0.78 \times \alpha + 1.2 \times 10^{-3} \times \alpha^2 - 1.124 \times 10^{-5} \times \alpha^3 \\
& + 3.238 \times \beta - 3.637 \times 10^{-2} \times \beta^2 + 1.555 \times 10^{-4} \times \beta^3 \\
& + 7.011 \times 10^{-2} \times \gamma - 7.077 \times 10^{-4} \times \gamma^2 + 3.055 \times 10^{-6} \times \gamma^3 \\
& + ((1.124 - 3.761 + 3.055 \times 10^{-6})/3) + 183.4
\end{aligned} \tag{2}$$

$$\begin{aligned}
PC_{INTEL} = & 0.802 \times \alpha + 0.042 \times \alpha' \\
& + 2.902 \times \beta - 2.107 \times 10^{-2} \times \beta^2 + 7.644 \times 10^{-5} \times \beta^3 \\
& - 7.645 \times 10^{-2} \times \gamma + 2.506 \times 10^{-3} \times \gamma^2 - 1.330 \times 10^{-5} \times \gamma^3 \\
& + ((56.362 + 36.89 + 11.20)/3) + 57
\end{aligned} \tag{3}$$

The proposed energy models were implemented in the version of Cloudsim developed by CICESE. Cloudsim handles the following main entities: hosts, which correspond to the physical host, virtual machines (VM), which are the processing units and assigned to the hosts, and jobs, which correspond to the workload. The jobs are assigned to the VMs to be processed.

Two Java classes were modified in Cloudsim: i) `PowerModelJobType`, located in package `cicese.cloudbus.cloudsim.power.models`, which is used by Cloudsim for scheduling tasks; and ii) `PowerModelJobType` located in package `cicese.cloudbus.cloudsim.util.power`, which is applied for computing the total power consumption in a post-processing stage, using the simulation results. In each Java class, the method `getEnergy(double  $\alpha$ , double  $\beta$ )` was extended to include the main features of each energy model developed in this research. This method returns the current power consumption of a VM that executes a task. The method `getEnergy(double  $\alpha$ , double  $\beta$ )` was overrode to include the empirical coefficients from the linear interpolation and the corresponding value of IC for each host.

5.2 Schedulers evaluation

This section describes the simulations performed for comparing different scheduling strategies regarding energy efficiency. The proposed model is used for calculating the power consumption for each scheduler.

5.2.1 Simulation details

Two different power consumption models were considered, modeling multicore hosts with AMD and Intel architectures (eight cores each), which correspond to versions of the linear combination models presented on Equations 2 and 3. Since the simulation only considered two type of workload (CPU-bound and memory-bound), the term of variable γ (disk-bound workload) was eliminated from the equations.

Thirty different workloads were considered in the evaluation for each architecture, thus a total number of 60 simulations were performed (without considering the independent executions of the stochastic algorithms). In order to focus the simulations in the energy consumption study, some simplifications were considered: i) the physical hosts within a same simulation are identical; ii) only one VM is created per physical host, which can execute as many tasks as cores available in the host; and iii) each job has only one task, so the terms job and task both denote the atomic workload unit to schedule. These simplifications are realistic in the context of scientific computing infrastructures, for example Cluster FING [31].

During a simulation, N independent jobs arrive in different times (considering realistic distributions, see Subsection 5.2.2 for details) and the scheduler must deliver the job to one VM from M active VMs. Each VM can execute up to eight jobs at the same time and a job requires only one core to execute (1/8 of the VM capacity). When all active VMs are full, a new VM is turned on and this VM is considered active from that moment. It is assumed that a VM always has enough memory, bandwidth, and disk to execute any job.

5.2.2 Workloads

The workloads used for the experimental evaluation are based on traces of real applications taken from HPC Parallel Workloads Archive executed on parallel supercomputers, clusters, and grids. The workloads include traces from DAS2-University of Amsterdam, DAS2-Delft University of Technology, DAS2-Utrecht University, DAS2-Leiden University, Royal Institute of Technology (KTH), DAS2-Vrije University Amsterdam, High Performance Computing Center North (HPC2N), Cornell Theory Center (CTC), and (Los Alamos National Laboratory) LANL. A detailed study of the workloads were presented by [45]. The information included in these workloads allows evaluating allocation strategies over real scenarios and applications.

Workloads are specified in a format that extends the Standard Workload Format (swf). The extended format, introduced by [11], adds two new fields to the format: *codec utilization* and *job type*. Fields of the extended swf format used in this work include:

- *Job id*: is the identifier of the job.
- *Submitted time*: is the arrival time of the job, in seconds. A job cannot begin execution before its submitted time.
- *Job length*: is a measure of the needed resources to complete the job, expressed in MIPS. When instantiated over a specific host, the duration of the job is given by the quotient of the job length and the power processing of the job (i.e., a job with length of 800 MIPS executing in a core of 100 MIPS will execute for eight seconds).
- *Codec utilization*: represents the percentage of a VM defined over the host that is requested to be used by a job. In the context of this research, codec represents the number of cores required by a job.
- *Job type*: this field allows specifying the type of the job (CPU-bound, memory-bound, disk-bound, or other relevant type). In the experiments performed in this article, type 0 represents a CPU-bound job and type 1 represents a memory-bound job.

5.2.3 Scheduling heuristics

The scheduling strategies were compared in experiments performed considering the two power consumption models over the 30 different workloads studied (W01, W02, ..., W30). Each workload accounts for one week of operation of a real HPC platform, as described in subsection 5.2.2. The computing platform simulated in the experiments is composed of 150 multicore hosts with eight cores each.

Figure 18: Scheduling approach

An on-line scheduling approach is applied. On-line scheduling is a model in which the scheduling decisions are taken immediately after a job arrives or is released [46]. Although they are based on local and often sub-optimal decisions, on-line schedulers provide some advantages over static scheduling methods including a more realistic model of the user-system interaction and less information is needed to perform the resource assignment. In the experiments, the destination VM is chosen by the cloud scheduler by applying a specific strategy when a new job arrives to the system. Figure 18 illustrates the scheduling approach used in simulations. The proposed scheduling strategies hold a list of active VMs (the active list), i.e., VMs that are currently executing jobs. Adding a VM to the active list corresponds to turning on a physical host. A VM is capable of executing a job if the number of free processors of the VM is greater than (or equal) the number of processors that this job requires. Strategies are implemented using well-know heuristics for scheduling problems. The proposed heuristics are oriented to fulfill the followings goals: minimize the host utilization, maximize the host utilization, balance the utilization, and minimize the power consumption of each host.

The studied heuristics include:

- *Random* (RD). The purpose of the heuristic is to assign the jobs randomly. The destination VM for an arriving job is selected randomly from the list of active VMs, applying a uniform distribution.
- *Round Robin* (RR). The purpose of the heuristic is to distribute equitably the jobs between the VMs, in a rational order [38, 47]. The destination VM for an arriving job is selected in circular order. VMs are considered ordered by VM identifier and when the last active VM is assigned, the VM with identifier equal to 0 is assigned again. Algorithm 4 explains the Round Robin strategy, where *vm_idx* is the index of the VM where the last job was assigned.

Algorithm 4 Round Robin Strategy

```

1: first_vm_idx  $\leftarrow$  vm_idx
2: assigned  $\leftarrow$  false
3: repeat
4:   vm  $\leftarrow$  active_vm_list[vm_idx];
5:   if is_capable(vm, arrived_job) then
6:     assign(vm, arrived_job)
7:     assigned  $\leftarrow$  true
8:   end if
9:   vm_idx = vm_index + 1 (mod size(active_vm_list))
10: until assigned or first_vm_idx = vm_idx

```

- *First Fit* (FF) assigns jobs to the first possible VM [48]. The destination VM for an arriving job is the first VM in the active list with enough free resources (in this work, cores) as requested by the arriving job. The active list is considered to be ordered by VM identifier, ascendant.
- *Minimum Energy* (mE). assigns jobs to the possible VM that consumes less power, according to the current assignment. The destination VM for an arriving job is the VM whose corresponding host has the lowest energy consumption. Algorithm 5 shows mE strategy.
- *Minimum Utilization* (mU) assigns jobs to the possible VM that have the most resources available (in this work, available cores). The destination VM for an arriving job is the VM of the active list with the lowest percentage of utilization.
- *Maximum Utilization* (MU) assigns jobs to the possible VM that have the less resources available (in this work, available cores). The destination VM for an arriving job is the VM of the active list with the highest percentage of utilization.

Algorithm 5 Minimum Energy strategy

```

1: for all vm in active_vm_list do
2:   if is_capable(vm, arrived_job) then
3:     if is_not_assigned(arrived_job) then
4:       assign(vm, arrived_job)
5:     else
6:       if current_power_of_host(vm) < current_power_of_host(assigned_vm(job)) then
7:         assign(vm, arrived_job)
8:       end if
9:     end if
10:  end if
11: end for

```

5.2.4 Simulation Results

Tables 18 and 19 report the power consumption values for each scheduling strategy using the AMD and Intel energy models, respectively, in W/s ($\times 10^8$). All scheduling strategies are deterministic, except for RD, which is non-deterministic. For RD, the mean and standard deviation of power consumption values (calculated in 20 independent simulations performed for each workload) are reported. Results in Tables 18 and 19 indicate that strategies that maximize the utilization of hosts (i.e., MU and FF) achieved better energy efficiency. On the AMD host, the FF strategy achieves the best results in 6 scenarios, MU achieves the best result in 18 scenarios, and the results of both strategies are the same in 6 scenarios. In addition, on the Intel host, the FF strategy achieves the best results in 5 scenarios, MU achieves the best result in 15 scenarios, and the results of both strategies are the same in 10 scenarios.

Conversely, lower efficiency is achieved when the utilization is minimized (mU). Strategy mE, which is oriented to minimize the energy consumption also achieved worse results. The fact that both utilization model and energy model are linear explains the results for strategy Me, because minimizing energy corresponds to minimizing the utilization. Strategies that balance the utilization (RR, RD) achieved intermediate results regarding the other reported strategies.

Table 18: Total energy consumption (tot) and improvement over RR strategy (Δ_{RR}) of the studied scheduling strategies using the AMD energy model in simulation, considering 30 weeks of realistic workloads

<i>Workload</i>	<i>RR</i>	<i>FF</i>		<i>MU</i>		<i>RD</i>		<i>mU</i>		<i>mE</i>	
		tot	Δ_{RR}	tot	Δ_{RR}	tot	Δ_{RR}	tot	Δ_{RR}	tot	Δ_{RR}
W1	1.51	1.29	14.57%	1.28	15.23%	1.47±0.03	2.65%	1.73	-14.57%	1.66	-9.93%
W2	1.41	1.26	10.64%	1.27	9.93%	1.47±0.02	-4.26%	1.6	-13.48%	1.58	-12.06%
W3	1.88	1.73	7.98%	1.72	8.51%	1.92±0.02	-2.13%	2.04	-8.51%	2.06	-9.57%
W4	2.31	2.12	8.23%	2.11	8.66%	2.32±0.03	-0.43%	2.5	-8.23%	2.48	-7.36%
W5	2.06	1.91	7.28%	1.87	9.22%	2.08±0.03	-0.97%	2.37	-15.05%	2.34	-13.59%
W6	1.79	1.61	10.06%	1.61	10.06%	1.77±0.03	1.12%	1.98	-10.61%	1.9	-6.15%
W7	1.86	1.53	17.74%	1.55	16.67%	1.70±0.04	8.60%	1.98	-6.45%	1.95	-4.84%
W8	2.09	1.95	6.70%	1.95	6.70%	2.14±0.04	-2.39%	2.34	-11.96%	2.25	-7.66%
W9	2.17	2.02	19.31%	2.01	19.30%	2.19±0.02	-8.42%	2.29	0.50%	2.29	-13.37%
W10	1.79	1.58	11.73%	1.54	13.97%	1.73±0.03	3.35%	1.91	-6.70%	1.83	-2.23%
W11	2.21	2	9.50%	1.99	9.95%	2.22±0.03	-0.45%	2.52	-14.03%	2.43	-9.95%
W12	2.14	1.87	12.62%	1.86	13.08%	2.08±0.04	2.80%	2.31	-7.94%	2.28	-6.54%
W13	2.37	2.05	13.50%	2.02	14.77%	2.25±0.04	5.06%	2.65	-11.81%	2.59	-9.28%
W14	1.91	1.81	5.24%	1.8	5.76%	1.91±0.02	0.00%	2.06	-7.85%	2.08	-8.90%
W15	2.23	2.05	8.07%	2.05	8.07%	2.23±0.03	0.00%	2.38	-6.73%	2.36	-5.83%
W16	1.51	1.47	2.65%	1.45	3.97%	1.62±0.03	-7.28%	1.81	-19.87%	1.76	-16.56%
W17	1.85	1.69	8.65%	1.67	9.73%	1.89±0.03	-2.16%	2.13	-15.14%	2.08	-12.43%
W18	1.57	1.48	5.73%	1.46	7.01%	1.59±0.03	-1.27%	1.69	-7.64%	1.65	-5.10%
W19	2.06	1.85	10.19%	1.87	9.22%	2.01±0.03	2.43%	2.24	-8.74%	2.2	-6.80%
W20	2.4	2.08	13.33%	2.07	13.75%	2.37±0.05	1.25%	2.58	-7.50%	2.41	-0.42%
W21	2.37	1.99	16.03%	1.99	16.03%	2.35±0.05	0.84%	2.76	-16.46%	2.66	-12.24%
W22	2.32	1.86	19.83%	1.89	18.53%	2.32±0.05	0.00%	2.65	-14.22%	2.49	-7.33%
W23	1.5	1.39	7.33%	1.4	6.67%	1.51±0.02	-0.67%	1.66	-10.67%	1.62	-8.00%
W24	1.9	1.7	10.53%	1.7	10.53%	1.83±0.02	3.68%	1.95	-2.63%	1.97	-3.68%
W25	1.96	1.74	11.22%	1.69	13.78%	1.92±0.03	2.04%	2.06	-5.10%	2.04	-4.08%
W26	2.25	2.05	8.89%	2.02	10.22%	2.29±0.04	-1.78%	2.48	-10.22%	2.39	-6.22%
W27	2.08	1.82	12.50%	1.81	12.98%	1.97±0.03	5.29%	2.29	-10.10%	2.19	-5.29%
W28	2.12	1.96	7.55%	1.89	10.85%	2.13±0.04	-0.47%	2.3	-8.49%	2.32	-9.43%
W29	2.73	2.47	9.52%	2.47	9.52%	2.72±0.05	0.37%	3.08	-12.82%	3.06	-12.09%
W30	2.47	2.18	11.74%	2.21	10.53%	2.48±0.03	-0.40%	2.74	-10.93%	2.79	-12.96%

A relevant analysis regarding energy efficient is the comparison of the best schedulers against strategies that apply a Business-as-Usual approach, i.e., when energy efficiency is not considered to perform the job-to-resources allocation. RR is one of the most widely used strategies for scheduling and it is included as the default scheduler in popular resource managers such as Maui and Condor [49]. Maui and Condor is widely used by multiple computational platform such as cluster, data centers, distributed platforms, among others.

The percent improvements over RR strategy (Δ_{RR}) of each proposed scheduling strategies are also reported in Tables 18 and 19. Results indicate that FF is able to improve over RR up to 19.83% on AMD and 15.66% in Intel. In turn, MU improves over RR up to 18.53% in AMD and up to 15.06% in Intel. Moreover, considering average results, FF improves over RR a 10.63% on AMD and a 8.07% in Intel. In turn, MU improves over RR a 11.15% in AMD and a 8.05% in Intel.

For all workloads studied, AMD hosts consumed more energy than Intel hosts. Moreover, if a performance model is added (with the characteristics studied in subsection 4.4), it is expected that the difference is even greater. The improvements of each scheduling over RR on AMD host are graphically presented in Figure 19 and the improvements of each scheduling over RR on Intel host are graphically presented in Figure 20.

Overall, the reported results indicate that energy models built from power characterization and empirical data are important tools for the operation and management of HPC and data center infrastructures. Energy models allow evaluating energy-aware scheduling strategies with the main goal of reducing power consumption in large computing platforms.

Table 19: Total energy consumption (tot) and improvement over RR strategy (Δ_{RR}) of the studied scheduling strategies using the Intel energy model in simulation, considering 30 weeks of realistic workloads

Workload	RR	FF		MU		RD		mU		mE	
		obj	Δ_{RR}	obj	Δ_{RR}	obj	Δ_{RR}	obj	Δ_{RR}	obj	Δ_{RR}
W1	1.09	0.96	11.93%	0.96	11.93%	1.07±0.02	1.83%	1.22	-11.93%	1.21	-11.01%
W2	1.04	0.94	9.62%	0.95	8.65%	1.07±0.01	-2.88%	1.15	-10.58%	1.11	-6.73%
W3	1.39	1.31	5.76%	1.3	6.47%	1.42±0.02	-2.16%	1.49	-7.19%	1.5	-7.91%
W4	1.66	1.55	6.63%	1.54	7.23%	1.66±0.01	0.00%	1.77	-6.63%	1.76	-6.02%
W5	1.48	1.42	4.05%	1.39	6.08%	1.51±0.02	-2.03%	1.68	-13.51%	1.63	-10.14%
W6	1.27	1.16	8.66%	1.16	8.66%	1.26±0.02	0.79%	1.38	-8.66%	1.35	-6.30%
W7	1.35	1.15	14.81%	1.16	14.07%	1.26±0.03	6.67%	1.42	-5.19%	1.39	-2.96%
W8	1.54	1.46	5.19%	1.46	5.19%	1.57±0.01	-1.95%	1.69	-9.74%	1.62	-5.19%
W9	1.61	1.53	4.97%	1.53	4.97%	1.63±0.02	-1.24%	1.69	-4.97%	1.7	-5.59%
W10	1.3	1.18	9.23%	1.16	10.77%	1.27±0.02	2.31%	1.37	-5.38%	1.34	-3.08%
W11	1.63	1.51	7.36%	1.5	7.98%	1.64±0.02	-0.61%	1.81	-11.04%	1.72	-5.52%
W12	1.58	1.42	10.13%	1.41	10.76%	1.54±0.02	2.53%	1.68	-6.33%	1.66	-5.06%
W13	1.74	1.55	10.92%	1.53	12.07%	1.67±0.03	4.02%	1.9	-9.20%	1.85	-6.32%
W14	1.45	1.39	4.14%	1.38	4.83%	1.45±0.02	0.00%	1.54	-6.21%	1.55	-6.90%
W15	1.71	1.61	5.85%	1.6	6.43%	1.71±0.01	0.00%	1.8	-5.26%	1.79	-4.68%
W16	1.1	1.11	-0.91%	1.1	0.00%	1.19±0.02	-8.18%	1.3	-18.18%	1.29	-17.27%
W17	1.35	1.26	6.67%	1.25	7.41%	1.37±0.02	-1.48%	1.51	-11.85%	1.46	-8.15%
W18	1.17	1.11	5.13%	1.11	5.13%	1.18±0.01	-0.85%	1.24	-5.98%	1.24	-5.98%
W19	1.53	1.4	8.50%	1.42	7.19%	1.50±0.02	1.96%	1.64	-7.19%	1.62	-5.88%
W20	1.73	1.54	10.98%	1.54	10.98%	1.71±0.03	1.16%	1.83	-5.78%	1.85	-6.94%
W21	1.73	1.5	13.29%	1.5	13.29%	1.72±0.02	0.58%	1.95	-12.72%	1.91	-10.40%
W22	1.66	1.4	15.66%	1.41	15.06%	1.66±0.02	0.00%	1.86	-12.05%	1.77	-6.63%
W23	1.12	1.06	5.36%	1.06	5.36%	1.13±0.02	-0.89%	1.21	-8.04%	1.17	-4.46%
W24	1.42	1.3	8.45%	1.29	9.15%	1.37±0.01	3.52%	1.45	-2.11%	1.45	-2.11%
W25	1.47	1.34	8.84%	1.31	10.88%	1.45±0.01	1.36%	1.53	-4.08%	1.5	-2.04%
W26	1.69	1.57	7.10%	1.55	8.28%	1.71±0.02	-1.18%	1.83	-8.28%	1.75	-3.55%
W27	1.54	1.38	10.39%	1.38	10.39%	1.47±0.02	4.55%	1.66	-7.79%	1.59	-3.25%
W28	1.55	1.46	5.81%	1.42	8.39%	1.56±0.02	-0.65%	1.66	-7.10%	1.67	-7.74%
W29	2.04	1.88	7.84%	1.88	7.84%	2.03±0.03	0.49%	2.24	-9.80%	2.28	-11.76%
W30	1.86	1.68	9.68%	1.7	8.60%	1.86±0.02	0.00%	2.02	-8.60%	2.04	-9.68%

Figure 19: Percentage improvements in energy consumption over RR using AMD PC model

Figure 20: Percentage improvements in energy consumption over RR using Intel PC model

6 Conclusions and future work

This work presented an empirical analysis of the energy consumption of synthetic benchmarks in high-end multi-core servers. In addition, power consumption models were constructed and used in simulations to evaluate several energy-aware scheduling strategies. In the last decade, data centers have become a fundamental part of computers technology, since they are the backbone of the cloud, which is widely used by industrial and scientific applications worldwide. Furthermore, data centers are large consumers of energy, which implies a great challenge when it comes to consciously manage energy consumption, in order to reduce the environmental impact and the monetary costs. The first step towards achieving energy efficiency in data centers is to know the energy consumption of its different components, in particular, the physical hosts.

In this work, an exhaustive study of the inter-relationship among the power consumption of the main computing resources at different ULs was carried out over AMD and Intel architectures. Furthermore, all experiments were performed applying the reproducible research paradigm, which is not found often in this research area. All data and processing tools are publicly available for verification and extension by researchers. The experimental methodology consisted on executing synthetic benchmarks over high-end hosts connected to a PDU, considering different ULs and combinations of benchmarks with the goal of characterizing the power consumption of each computing resource (CPU, memory, and disk). The operations performed by the benchmarks include mathematical functions and read/write of main memory and disk. The study was complemented with performance experiments. A total number of 144 experiments were performed: 96 experiments evaluating power consumption and 48 evaluating performance. For each experiment, 20 independent executions were performed. Experimental results showed that in single executions, CPU utilization has a linear relation with power consumption. Memory utilization has a significant impact on power consumption when compared to CPU, up to 157% more EC for AMD and 46% more EC for Intel. On the other hand, disk utilization presented low EC variation for all ULs. Combined executions are able to reduce EC with regard to independent executions for CPU and disk combined execution, taking advantage of the optimum UL of computing resources. The results of two combined executions showed a remarkable energy reduction in the Intel host (32% of the EC of single executions). Efficiency analysis showed that different benchmarks performed more efficiently at different ULs: CPU at high ULs, memory at medium ULs, and disk at low ULs. The critic UL (100%) showed worse efficiency than high-medium UL (87.5%), except for disk.

Statistical tools were applied to built a realistic power consumption model for multicores. The models built provide a good fit to the empirical data according to relevant metrics for statistical models, and they are useful tools to assess the capabilities of scheduling strategies regarding energy efficiency. In order to evaluate energy-aware scheduling strategies, a total of 60 different simulations were performed considering linear energy models for AMD and Intel architectures and 30 realistic workloads based on traces of HPC Parallel Workloads Archive executed on real computing infrastructures. The simulation results showed that strategies which maximize the host utilization achieves better results, notably improving over traditional Business-as-Usual schedulers.

The research results confirm that power consumption models built from empirical data are a key tool for operation and management of data center infrastructures. They allow evaluating different scheduling strategies via simulations to provide accurate methods that account for environmental and monetary savings.

The work presented in this manuscript is the first step towards the construction of an energy-efficiency management of datacenters, guided by computational intelligence. The main lines for future work are related to including different benchmarks and real scientific computing applications. Regarding benchmarks, network-bound and GPU-based programs could be studied, mainly due to their relevance in modern datacenters and supercomputing infrastructures. The power characterization of non-synthetic benchmarks and programs from different scientific fields (e.g., molecular dynamics, quantum chemistry, computational fluid dynamics, climate modeling, etc.) will provide a useful insight for developers and scientist, which will certainly complement the analysis presented in this manuscript. Considering other high-end hosts with different chip designs is also a possible line for future work, in order to extend the hardware components on the study. More complex energy models can also be considered, including features of combined executions, and enhancing the planning capabilities by considering resource utilization models, also derived from empirical data. Extending the characterization of power consumption and building complex models can lead to designing powerful scheduling strategies. These strategies must consider energy and QoS together as an optimization objective, to meet the requirements of modern supercomputing facilities.

References

- [1] R. Buyya, C. Vecchiola, and S. Selvi, *Mastering Cloud Computing: Foundations and Applications Programming*. San Francisco, CA, USA: Morgan Kaufmann, 2013. [Online]. Available: <https://doi.org/10.1016/B978-0-12-411454-8.00020-6>
- [2] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 732–794, 2016. [Online]. Available: <https://doi.org/10.1109/COMST.2015.2481183>
- [3] S. Nesmachnow, C. Perfumo, and I. Goiri, "Holistic multiobjective planning of datacenters powered by renewable energy," *Cluster Computing*, vol. 18, no. 4, pp. 1379–1397, 2015. [Online]. Available: <https://doi.org/10.1007/s10586-015-0485-1>
- [4] A. Anghel, L. Vasilescu, G. Mariani, R. Jongerius, and G. Dittmann, "An instrumentation approach for hardware-agnostic software characterization," *International Journal of Parallel Programming*, vol. 44, no. 5, pp. 924–948, 2016. [Online]. Available: <https://doi.org/10.1145/2742854.2742859>
- [5] C. Brandolese, S. Corbetta, and W. Fornaciari, "Software energy estimation based on statistical characterization of intermediate compilation code," in *International Symposium on Low Power Electronics and Design*, 2011, pp. 333–338. [Online]. Available: <https://doi.org/10.1109/ISLPED.2011.5993659>
- [6] K. Kurowski, A. Oleksiak, W. Piatek, T. Piontek, A. Przybyszewski, and J. Weglarz, "DCworms—a tool for simulation of energy efficiency in distributed computing infrastructures," *Simulation Modelling Practice and Theory*, vol. 39, pp. 135–151, 2013. [Online]. Available: <https://doi.org/10.1016/j.simpat.2013.08.007>
- [7] S. Hernández, J. Fabra, P. Álvarez, and J. Ezpeleta, "Simulation and realistic workloads to support the meta-scheduling of scientific workflows," in *Simulation and Modeling Methodologies, Technologies and Applications*, Cham, 2014, pp. 155–167. [Online]. Available: https://doi.org/10.1007/978-3-319-03581-9_11
- [8] S. Bak, M. Krystek, K. Kurowski, A. Oleksiak, W. Piatek, and J. Waglarz, "GSSIM—a tool for distributed computing experiments," *Scientific Programming*, vol. 19, no. 4, pp. 231–251, 2011. [Online]. Available: <https://doi.org/10.1155/2011/925395>
- [9] R. Malhotra and P. Jain, "Study and comparison of various cloud simulators available in the cloud computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 9, pp. 347–350, 2013.
- [10] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011. [Online]. Available: <https://doi.org/10.1002/spe.995>

- [11] F. Armenta-Cano, A. Tchernykh, J. Cortes-Mendoza, R. Yahyapour, A. Drozdov, P. Bouvry, D. Kliazovich, A. Avetisyan, and S. Nesmachnow, “Min_c: Heterogeneous concentration policy for energy-aware scheduling of jobs with resource contention,” *Programming and Computer Software*, vol. 43, no. 3, pp. 204–215, 2017. [Online]. Available: <https://doi.org/10.1134/S0361768817030021>
- [12] J. Muraña, S. Nesmachnow, F. Armenta, and A. Tchernykh, “Characterization, modeling and scheduling of power consumption of scientific computing applications in multicores,” *Cluster Computing*, vol. 22, no. 3, pp. 839–859, 2019. [Online]. Available: <https://doi.org/10.1007/s10586-018-2882-8>
- [13] S. Iturriaga, S. García, and S. Nesmachnow, “An empirical study of the robustness of energy-aware schedulers for high performance computing systems under uncertainty,” in *High Performance Computing*, 2014, pp. 143–157. [Online]. Available: https://doi.org/10.1007/978-3-662-45483-1_11
- [14] S. Nesmachnow, B. Dorronsoro, J. Pecero, and P. Bouvry, “Energy-aware scheduling on multicore heterogeneous grid computing systems,” *Journal of Grid Computing*, vol. 11, no. 4, pp. 653–680, 2013. [Online]. Available: <https://doi.org/10.1007/s10723-013-9258-3>
- [15] S. Srikantaiah, A. Kansal, and F. Zhao, “Energy aware consolidation for cloud computing,” in *Conference on Power Aware Computing and Systems*, 2008, pp. 1–5.
- [16] K. Du Bois, T. Schaeps, S. Polfiet, F. Ryckbosch, and L. Eeckhout, “Sweep: Evaluating computer system energy efficiency using synthetic workloads,” in *6th International Conference on High Performance and Embedded Architectures and Compilers*, 2011, pp. 159–166. [Online]. Available: <https://doi.org/10.1145/1944862.1944886>
- [17] X. Feng, R. Ge, and K. Cameron, “Power and energy profiling of scientific applications on distributed systems,” in *19th IEEE International Parallel and Distributed Processing Symposium*, 2005, pp. 34–44. [Online]. Available: <https://doi.org/10.1109/IPDPS.2005.346>
- [18] A. Langer, E. Totoni, U. Palekar, and L. Kalé, “Energy-efficient computing for HPC workloads on heterogeneous manycore chips,” in *Proceedings of the 6th International Workshop on Programming Models and Applications for Multicores and Manycores*, 2015, pp. 11–19. [Online]. Available: <https://doi.org/10.1145/2712386.2712396>
- [19] L. Barroso, J. Clidaras, and U. Hözl, “The datacenter as a computer: An introduction to the design of warehouse-scale machines,” *Synthesis lectures on computer architecture*, vol. 8, no. 3, pp. 1–154, 2013. [Online]. Available: <https://doi.org/10.2200/S00516ED2V01Y201306CAC024>
- [20] K. Malladi, F. Nothaft, K. Periyathambi, B. Lee, C. Kozyrakis, and M. Horowitz, “Towards energy-proportional datacenter memory with mobile DRAM,” in *39th Annual International Symposium on Computer Architecture*, 2012, pp. 37–48. [Online]. Available: <https://doi.org/10.1109/ISCA.2012.6237004>
- [21] E. Totoni, N. Jain, and L. Kalé, “Toward runtime power management of exascale networks by on/off control of links,” in *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, 2013, pp. 915–922. [Online]. Available: <https://doi.org/10.1109/IPDPSW.2013.191>
- [22] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. Khan, “Greencloud: A packet-level simulator of energy-aware cloud computing data centers,” in *IEEE Global Telecommunications Conference*, 2010, pp. 1–5. [Online]. Available: <https://doi.org/10.1109/GLOCOM.2010.5683561>
- [23] A. Núñez, J. Vázquez-Poletti, A. Caminero, G. Castañé, J. Carretero, and I. Llorente, “ICanCloud: A flexible and scalable cloud infrastructure simulator,” *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, 2012. [Online]. Available: <https://doi.org/10.1007/s10723-012-9208-5>
- [24] Y. Zhang and N. Ansari, “HERO: Hierarchical energy optimization for data center networks,” *IEEE Systems Journal*, vol. 9, no. 2, pp. 406–415, 2015. [Online]. Available: <https://doi.org/10.1109/ICC.2012.6363830>
- [25] Z. Guo, S. Hui, Y. Xu, and H. Chao, “Dynamic flow scheduling for power-efficient data center networks,” in *IEEE/ACM 24th International Symposium on Quality of Service*, 2016, pp. 1–10. [Online]. Available: <https://doi.org/10.1109/IWQoS.2016.7590399>
- [26] J. Dongarra, “The linpack benchmark: An explanation,” in *Proceedings of the 1st International Conference on Supercomputing*. London, UK: Springer-Verlag, 1988, pp. 456–474. [Online]. Available: https://doi.org/10.1007/3-540-18991-2_27

- [27] X. Gonze, B. Amadon, P. Anglade, J. Beuken, F. Bottin, P. Boulanger, F. Bruneval, D. Caliste, R. Caracas, M. Côté, T. Deutsch, L. Genovese, P. Ghosez, M. Giantomassi, S. Goedecker, D. Hamann, P. Hermet, F. Jollet, G. Jomard, S. Leroux, M. Mancini, S. Mazevet, M. Oliveira, G. Onida, Y. Pouillon, T. Rangel, G. Rignanese, D. Sangalli, R. Shaltaf, M. Torrent, M. Verstraete, G. Zerah, and J. Zwanziger, “ABINIT: First-principles approach to material and nanosystem properties,” *Computer Physics Communications*, vol. 180, no. 12, pp. 2582 – 2615, 2009. [Online]. Available: <https://doi.org/10.1016/j.cpc.2009.07.007>
- [28] J. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. Skeel, L. Kale, and K. Schulten, “Scalable molecular dynamics with NAMD,” *Journal of Computational Chemistry*, vol. 26, no. 16, pp. 1781–1802, 2005. [Online]. Available: <https://doi.org/10.1002/jcc.20289>
- [29] P. Nielsen, “cpuburn,” <https://github.com/patrickmn/cpuburn>, 2012, online; accessed April 2019.
- [30] A. Kopytov, “Sysbench repository,” <https://github.com/akopytov/sysbench>, 2017, online; accessed June 01 2017.
- [31] S. Nesmachnow, “Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República,” *Revista de la Asociación de Ingenieros del Uruguay*, vol. 61, no. 1, pp. 12–15, 2010, text in Spanish.
- [32] R. Grant, J. Laros, M. Levenhagen, S. Olivier, K. Pedretti, L. Ward, and A. Younge, “Evaluating energy and power profiling techniques for HPC workloads,” in *Eighth International Green and Sustainable Computing Conference*, 2017, pp. 1–8. [Online]. Available: <https://doi.org/10.1109/IGCC.2017.8323587>
- [33] C. Isci and M. Martonosi, “Runtime power monitoring in high-end processors: methodology and empirical data,” in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, 2003, pp. 93–104. [Online]. Available: <https://doi.org/10.1109/MICRO.2003.1253186>
- [34] J. Treibig, G. Hager, and G. Wellein, “Likwid: A lightweight performance-oriented tool suite for x86 multicore environments,” in *39th International Conference on Parallel Processing Workshops*, 2010, pp. 207–216. [Online]. Available: <https://doi.org/10.1109/ICPPW.2010.38>
- [35] M. Rashti, G. Sabin, D. Vansickle, and B. Norris, “WattProf: A flexible platform for fine-grained HPC power profiling,” in *IEEE International Conference on Cluster Computing*, 2015, pp. 698–705. [Online]. Available: <https://doi.org/10.1109/CLUSTER.2015.121>
- [36] J. Laros, P. Pokorny, and D. DeBonis, “PowerInsight - a commodity power measurement capability,” in *International Green Computing Conference Proceedings*, 2013, pp. 1–6. [Online]. Available: <https://doi.org/10.1109/IGCC.2013.6604485>
- [37] D. Cox and N. Reid, *The theory of the design of experiments*. Chapman and Hall/CRC, 2000.
- [38] J. Leung, L. Kelly, and J. Anderson, *Handbook of scheduling: algorithms, models, and performance analysis*. Boca Raton, FL, USA: CRC Press, Inc., 2004.
- [39] CPUBOSS, “Intel Xeon E52643v3 vs AMD Opteron 6172 comparison,” <http://cpuboss.com/cpus/Intel-Xeon-E5-2643-v3-vs-AMD-Opteron-6172>, 2014, online; accessed March 29 2018.
- [40] Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan, and L. Liu, “Service level agreement based energy-efficient resource management in cloud data centers,” *Computers & Electrical Engineering*, vol. 40, no. 5, pp. 1621–1633, 2014. [Online]. Available: <https://doi.org/10.1016/j.compeleceng.2013.11.001>
- [41] A. Peckov, “A machine learning approach to polynomial regression,” Ph.D. dissertation, Jozef Stefan International Postgraduate School, Ljubljana, 2012.
- [42] A. Repko, *Interdisciplinary research : process and theory*. Los Angeles, USA: SAGE Publishing, 2008.
- [43] C. Begley, “Six red flags for suspect work,” *Nature*, vol. 497, no. 7450, pp. 433–434, 2013. [Online]. Available: <https://doi.org/10.1038/497433a>
- [44] H. Theil, *Economic forecasts and policy*. Amsterdam North-Holland Publishing Company, 1961.
- [45] D. Feitelson, D. Tsafir, and D. Krakov, “Experience with using the parallel workloads archive,” *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2967–2982, 2014. [Online]. Available: <https://doi.org/10.1016/j.jpdc.2014.06.013>

- [46] A. Tchernykh, L. Lozano, P. Bouvry, J. Pecero, U. Schwiegelshohn, and S. Nesmachnow, “Energy-aware online scheduling: Ensuring quality of service for IaaS clouds,” in *International Conference on High Performance Computing Simulation*, 2014, pp. 911–918. [Online]. Available: <https://doi.org/10.1109/HPCSim.2014.6903786>
- [47] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 9th ed. Wiley Publishing, 2012.
- [48] H. El-Rewini, T. Lewis, and H. Ali, *Task Scheduling in Parallel and Distributed Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1994.
- [49] D. Jackson, Q. Snell, and M. Clement, “Core algorithms of the Maui scheduler,” in *Job Scheduling Strategies for Parallel Processing*, 2001, pp. 87–102. [Online]. Available: https://doi.org/10.1007/3-540-45540-X_6