CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2017; 29: e3839
Published online 12 May 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3839

Modeling and simulation of global and sleep states in ACPI-compliant energy-efficient cloud environments

Miguel G. Xavier[1,*,†], Fábio D. Rossi[1], César A. F. De Rose[1], Rodrigo N. Calheiros[2] and Danielo G. Gomes[3]

[1]*Pontifical Catholic University of Rio Grande do Sul, Brazil*
[2]*The University of Melbourne, Australia*
[3]*Federal University of Ceará, Brazil*

SUMMARY

The more large-scale data centers infrastructure costs increase, the more simulation-based evaluations are needed to understand better the trade-off between energy and performance and support the development of new energy-aware resource allocation policies. Specifically, in the cloud computing field, various simulators are able to predict and measure the behavior of applications on different architectures using different resource allocation policies. Yet, only a few of them have the ability to simulate energy-saving strategies, and none of them support the complete advanced configuration and power interface (ACPI) specification. ACPI defines a terminology for all possible power states of a machine and their associated power rate. The hardware industry has relied on ACPI to provide up-to-date standard interfaces for hardware discovery, configuration, power management, and monitoring, enabling a better understanding of the energy consumption level of different hardware states, referred to as ACPI G-states, S-states, and P-states. In this paper, we improve the modeling and simulation of the ACPI G/S-states and show not only that these states offer different energy-saving levels but also that state transitions consume energy. In addition, we model the latency to transit between two states and the effects on the turnaround time when the transitions are not performed conservatively. Furthermore, the equations provide essential information to quantify the trade-off between energy consumption and performance and assist in the analysis/decision on which strategy fits better in the environment and how it could be refined. Our expanded energy model was implemented in CloudSim and validated with simulation-based experiments with a very high level of accuracy, with a standard deviation of at most 6%. Copyright © 2016 John Wiley & Sons, Ltd.

Received 1 April 2015; Revised 8 December 2015; Accepted 10 March 2016

KEY WORDS: Green IT; cloud computing; modeling and simulation; resource management

1. INTRODUCTION

Cloud computing delivers infrastructure, platform, and software as a service under the pay-as-you-go model on an unprecedented scale [1]. Companies and developers do not need to make large investments in hardware and maintenance services, allowing them to focus more on innovation and improving business enterprises. These new opportunities increased the popularity of cloud computing because of its reliability, security, availability, fault tolerance, scalability, and sustainability.

Although there is already a variety of cloud computing systems, there is still no standard for evaluating these environments. An appropriate alternative is the use of simulation tools through which systems not physically available can be characterized. Simulation is a useful technique for

*Correspondence to: Miguel Xavier, Faculty of Informatics, Pontifical Catholic University of Rio Grande do Sul, Ipiranga Avenue 6681, 90619-900 Porto Alegre, RS, Brazil.
†E-mail: miguel.xavier@acad.pucrs.br

computer system analysis to reproduce tests, evaluate hypotheses, and compare several scenarios. Therefore, simulation-based experiments may be preferred over real experiments because they allow the alternatives to be compared under a wider variety of workloads and environments. Furthermore, it helps to detect bottlenecks and trade-offs before the deployment of the solution on a physical infrastructure[‡]

Identification of the impact of energy on performance in large-scale cloud data centers is one of today's most studied topics on energy-efficient cloud computing [2–4]. The green IT-related key issues involve reducing carbon emissions, which implies efficient management of energy usage, with the reduction of equipment and rethinking business practices that cause the least impact to the environment. In recent years, the US data centers consumed an estimated 91 billion kilowatt-hours (kWh) of electricity, which is equivalent to the annual output of 34 large (500-megawatt) coal-fired power plants [5]. Currently, this scenario has become worse, and the numbers are even higher, especially at the end of the year when cloud services have a high demand for resources (retail has large demand spikes around Christmas).

This worrying scenario has stimulated many studies proposing strategies for energy conservation, aiming at reducing the impact on the environment. Niyato *et al.* [6] proposed an ideal power management based on a Markov model to adjust the number of active servers for maximum energy-savings. Beloglazov and Buyya [7] presented a heuristic for virtual machines allocation in a cloud with the goal of saving energy. Duy *et al.* [8] presented the design, implementation, and evaluation of a scheduling algorithm integrated with a predictor that uses neural networks to optimize the energy consumption of servers in a cloud. Alvarruiz *et al.* [9] proposed a management system for clusters and clouds that saves energy by turning off idle nodes across the network. Isci *et al.* [10] showed that there is an opportunity for energy-saving strategies in these environments using the concept of sleep states.

Such strategies are based on the fact that not all physical hosts are overutilized all the time (the hosts usage rate differs substantially in a large-scale data center), leading to energy-savings policies, such as

- **reduce the processors frequencies:** processor frequencies might be scaled up/down on demand based on their utilization rate. This capability provides a high degree of energy conservation while putting the processors to operate at the lowest frequency. This is the most straightforward and usual policy because it is set once and remains unchanged during the operating system (OS) run-time;
- **put idle hosts to sleep (sleep states):** consolidation of virtual machines in cloud data centers is arguably a well-established approach to reduce costs and make better use of the resources. The higher the number of consolidated virtual machines, the greater the number of idle physical hosts consuming energy needlessly. Based on this, it is possible to put idle hosts into a lower-power state by turning off its internal hardware components.

Many strategies have adopted such policies to increase energy efficiency, but they are not feasible if the underlying hardware does not provide standard interfaces for power management of its internal components. To this end, the hardware industry has implemented such standard interfaces, enabling robust OS-directed motherboard device configuration and power management of both devices and entire systems. These standard interfaces have been defined since 1996 in the specification referred to as advanced configuration and power interface (ACPI) [11]. In broad terms, the specification defines, among many other things, a terminology for all possible power states of a machine and their associated energy consumption levels. These definitions in ACPI are implicitly cited when talking about processor power states (P-states) and machine power states (G/S-states) in any energy-saving strategy. With the continuous evolution of standards in its collections, the ACPI has become a de facto standard in the industry and has been widely supported by companies such as Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba [11].

Putting ACPI-compliant machines into a G-/S-state contributes significantly to energy-savings but imposes an additional latency to enter or leave a state. This latency might delay resource availability

[‡]Cloud computing environments composed of physical resources, such as servers, switches, and routers.

DOI: 10.1002/cpe

and affect the scheduling performance (job turnaround time). There have been many studies focusing on identifying the trade-off between energy consumption and performance to sidestep this disruptive effect. Our proposal is complementary; we focus on the **modeling** and **simulation** of the ACPI G/S-states, considering the trade-off to switch a host from one state to another. We correlate power rate and latency to identify the trade-off (Section 4). We validate our model in CloudSim, allowing it to simulate a handful of energy-saving strategies previously impossible on a robust and well-validated platform (Sections 5 and 6). As we shall see, the cost to switch a host between two states plays an important role in our model, because the transitions imply different power rates and latencies to return a host to the G0 state (a state in which the host might be used for virtual machine provisioning) (Section 7).

2. BACKGROUND AND STATE OF THE ART

This section presents a background on the terms used in energy-saving solutions and the state-of-the-art concepts that are relevant for energy-aware studies. In order to drive the reader with the concepts intrinsic to the work, we would like to distinguish them as follows:

- Power model: is a real-world scenario description using mathematical terms and languages for this purpose. In the context of energy consumption, power models might be understood as being an equation or a set of equations that describe the power rate of a system, in which the effects of its components, as well as the predictions of its behavior, are explained. The energy consumption is obtained through the integral of the power model equations.
- Energy-saving strategy [12]: represents actions performed by a given entity upon a system with the sole purpose of reducing the overall energy consumption. Strategies normally benefit from power models to reach their goals;
- Energy-saving policy [12]: denotes the behavior of a system when an energy-saving strategy acts on it.

The challenge to be tackled in simulation methods is how to develop a model that most closely represents a real environment. Hence, models should be implemented/validated in simulators, and accuracy is achieved when the simulated results are close to real results. Otherwise, the simulation produces no useful or misleading results. The first part of this section covers work that present power models for cloud computing environments. Then, we summarize studies that implement power modules in simulators to evaluate strategies with policies involving scheduling, placement, consolidation, scaling of processors frequencies (P-states), and shutdown hardware components (G/S-states).

2.1. Energy-aware modeling

Some recent work in energy-efficient cloud computing benefits from virtualization capabilities, such as load balancing, resource allocation, and virtual machine scheduling, to make the environment more sustainable [13]. Another study proposes power models to quantify the energy consumption of different workloads [14]. Beloglazov and Buyya [15] balance the service-level agreement (SLA) metrics and energy constraints, describing the energy consumption through a linear model.

Bohra and Chaudhary [16] proposed a model that considers the utilization of CPU, RAM, cache, and disk. The model consists of a four-dimensional linear regression, allowing a computational resource usage prediction with 82% accuracy. This model can be seen in the following, where P_{CPU}, P_{cache}, P_{DRAM}, and P_{disk} are the explanatory variables that denote computational resources and C0, C1, C2, C3, C4 are weights that calculated the workflow.

$$P_{total} = C_0 + C_1 P_{CPU} + C_2 P_{cache} + C_3 P_{DRAM} + C_4 P_{disk} \quad (1)$$

This model predicts the power rate of a single core host but cannot determine the individual consumption of each virtual machine. Thus, it was decomposed into a linear system for this purpose, as follows:

DOI: 10.1002/cpe

$$P_{total} = P_{(baseline)} + \sum_{k=0}^{N} P_{(domain(k))} \quad (2)$$

where $P_{(domain(k))}$ corresponds to the power rate implied by an active domain (a domain is one of the virtual machines that run on the system) and N is the number of active domains including domain0. Domain0 is the first domain started by the Xen hypervisor at boot time. Based on this model, Chen *et al.* [17] proposed a new model that uses only two of the most power-consuming components: processor and disk. C_0 is replaced by P_{idle} as follows:

$$P_{total} = P_{idle} + C_1 P_{CPU} + C_2 P_{HDD} \quad (3)$$

The new model (equation 3) aims to infer the power rate based on just these two components, because cache and RAM are not significant enough to disrupt the results. The management of cloud computing environments is a complex task because of their layered structure and the heterogeneity of their resources.

Bruneo *et al.* [18] present several characteristics of these environments that can hinder management, such as allocating, migrating, and consolidating resources, in addition to managing when, how, and where to instantiate new virtual machines. To this end, they present a stochastic model (stochastic reward nets) that investigates the best strategy to manage cloud environments, focused on reducing energy consumption.

Salehi *et al.* [19] present a power module for Haizea [20] serving as a scheduler for cloud computing platforms such as OpenNebula. Simulation traces from a supercomputing center in San Diego were used and showed a reduction in energy consumption of 18% within 30 days.

Although current models are quite accurate regarding power rate of individual components, they do not take into account the trade-off to put a machine in a power state. The model proposed in this work overcomes this limitation, allowing for a more realistic cost-benefit analysis of energy-saving strategies based on ACPI states.

2.2. Energy-aware simulation

This section presents implementations of power models in simulators to support energy-saving strategy evaluations.

CloudSim [21] consists of a general system and extensive simulator, enabling modeling, simulating, and testing infrastructure in the emerging cloud computing applications and services. Some advantages can be identified, such as requiring less effort for test implementation while allowing large-scale environment simulation. Also, it allows flexibility in the choice and implementation of new policies for using resources and services. The power module has been introduced in CloudSim in a partial implementation of ACPI focused on dynamic voltage and frequency scaling (DVFS) [22] by Guérout *et al.* [23].

The GreenCloud simulator [24] extends the functionalities of the NS-2 network simulator [25] to measure the power rate of communication components and packet-level patterns for data centers. Also, it includes specific functionalities of virtualized data centers, such as virtual machine migration. All topologies supported by NS-2 can be used, allowing power measurement in several TCP operations, routing, and protocols.

ICanCloud simulator [26] is focused on Amazon's cloud environment, allowing flexibility in the choice of the hypervisor and providing large-scale simulation. The aim of this simulator is to predict the trade-offs between energy consumption and performance of applications running on a specific hardware. To this end, the simulated scenarios allow for the use of various data center components, such as disk, network, memory, and file system environments, including their behavior.

The MDCSim [27] presents a power module based on the workload execution time and CPU usage rate. MDCSim represents a typical three-tier environment (application, web server, and database). The simulator allows multiple scenario configurations varying the network latency, cluster size, and workload. Based on this information, the user can scale the cloud environment to suit

DOI: 10.1002/cpe

performance metrics, such as response time or number of transactions per minute. The power module implemented in MDCSim uses a linear function based on three components: a fixed CPU's frequency, the energy consumption of the hosts, and the application throughput.

There have been many cloud computing simulators that support energy-saving strategy evaluations, but none of them implements the complete ACPI specification. Moreover, recent studies lack a proper understanding of the trade-off between energy consumption and performance when dealing with the simulation of the ACPI states.

3. ADVANCED CONFIGURATION AND POWER INTERFACE

The ACPI is a specification that provides an open standard for OS power management. It was designed to allow OSs to configure and control each hardware component, replacing both the predecessors plug and play (PnP) energy management and the advanced power management (APM). In modern hosts, the firmware-level ACPI is implemented in the ACPI BIOS code, which provides tables containing information on hardware access methods. OSs use this information for tasks like assigning interrupts or (de)activating hardware components. As this management is performed by the OS, there is greater flexibility regarding energy-saving modes for CPU and several other devices present in the hardware. This section outlines how ACPI is organized and how its components relate to each other.

3.1. Advanced configuration and power interface architecture

The ACPI architecture can be seen in Figure 1, where the upper part represents user-mode applications and threads dispatched by OS. The communication between OS and hardware platform is performed by a device driver. Likewise, power management is carried out by the ACPI driver through a communication between OS and the hardware platform.

The ACPI driver manages three different components: ACPI Tables, ACPI Registers, and ACPI BIOS. ACPI Tables contain hardware descriptions managed through ACPI, including machine-independent byte-code used to perform hardware management operations. ACPI Registers provide low-level hardware management operations. Finally, during the hardware designing, additional registers are implemented to be accessed through the byte-code stored in the device-specific part of the ACPI tables, referred to as ACPI BIOS.

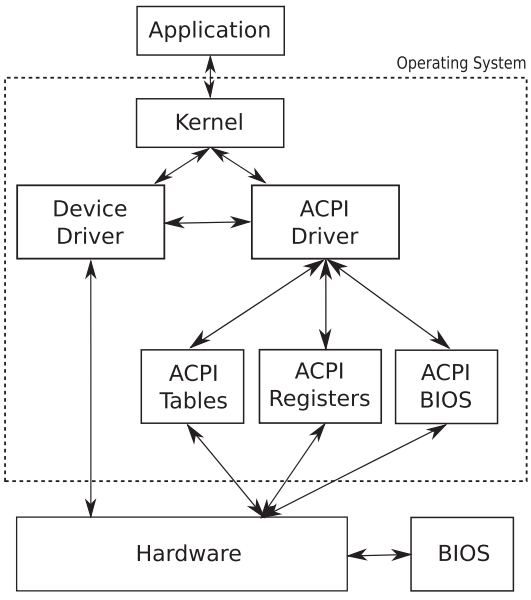

Figure 1. The advanced configuration and power interface architecture [11].

DOI: 10.1002/cpe

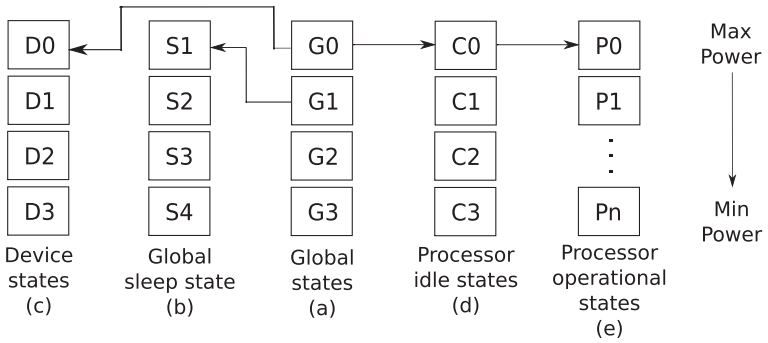

Figure 2. Aadvanced configuration and power interface states based on [11].

In computers that support ACPI, before the OS is loaded, the ACPI BIOS puts the ACPI Tables in memory. Thus, when the OS is started, it searches for a small data structure with the valid signature within the BIOS and uses a pointer to the ACPI Tables to find the definition of the hardware blocks. The ACPI Registers store changes that are made in the ACPI Tables. By ACPI, the OS has the ability to put devices in and out of low-power states. Devices that are in use can be turned off. Similarly, it uses information from applications and user settings to put the system as a whole into a low-power state.

Although ACPI is a standard used in today's computers, several legacy architectures remain in use in data centers. Most of these architectures have their resources managed directly by the firmware. The main firmware-based approach is the system management mode (SMM). As computer architectures evolved, conflicts started to be observed between information obtained via SMM and OS [28]. Because in SMM, the processor's states are stored in the system management RAM, ACPI can read these legacy states and translate them to supported ACPI states.

3.2. Advanced configuration and power interface states overview

From a user-visible level, the system can be thought of as being in one of the power states presented in Figure 2. Moreover, the arrow indicates the depth of energy savings provided by each state. ACPI specifies different power state levels, which are global states, sleep states, device states, and processor states. Some of these levels comprehend IT resources, such as computers, hard disks, and graphic cards, in addition to other peripherals, such as the processor chip.

The global states (Figure 2(a)) denote the entire system and are visible only to the user. Sleep states (Figure 2(b)) are power states derived from the G1 state and are visible only to the system. When the user has pressed the power button, for example, the power states of a particular device (Figure 2(c)) are usually not visible to the user. For instance, devices may be turned off while the system keeps working. Finally, processor states (Figure 2(d)) are power states within the G0 state (working state). It means the processor states may vary if the computer is processing something. Besides those mentioned states, DVFS [22] is the name given by the industry to P-states (Figure 2(e)). Each level denotes one of all available modern CPU's frequencies, which in conjunction with the ACPI-based firmware allows on-the-fly adjustment based on the CPU load. Table I shows the depth levels of the mentioned states, as well as their descriptions. The deeper the state, the lower the power rate and the higher the latency for returning to the working state.

3.3. Synthesis and discussion

The first most widely used power state in energy-saving strategies is called standby (S1), which turns off the screen, hard drives, and fans. Because all open programs are kept stored in RAM, the memory remains active, requiring little power for maintaining user data until some external event occurs and turn the subsystems back on. The advantage of this state is the short time required for the computer to be on again. This is fundamental in situations where the computer must be awakened

DOI: 10.1002/cpe

Table I. Description of the advanced configuration and power interface states.

Global states	
G0	the system and user threads are running (working)
G1	the system consumes a small amount of power, user mode threads are not running, the system appears as if turned off, and the system context is saved (sleeping)
G2	the system consumes a minimal amount of power, user mode threads and system processes are not running, and the system context is not saved (Soft Off)
G3	the system is turned off (Mechanical Off)
Sleep states	
S1	no system context is lost
S2	CPU and system cache context are lost
S3	CPU, system cache, and chip set context are lost
S4	powered off all devices
Device states	
D0	device is turned on and running
D1	low-power state when the device context may or may not be lost
D2	low-power state when the device context may or may not be lost, and the power supply of the bus is reduced
D3	device is turned off and not running
Processor states	
C0	CPU is on
C1	CPU is not executing instructions
C2	CPU main internal clocks are stopped
C3	deep sleep
Processor operational states	
P0	maximum processor performance capability and may consume maximum power
P1	the processor performance capability is limited below its maximum and consumes less than maximum power
Pn	the processor performance capability is at its minimum level and consumes minimal power while remaining in an active state

DOI: 10.1002/cpe

to all possible events or do so very quickly. As the context of the OS is stored in a volatile memory that requires power to keep up the data, there is a disadvantage when instabilities occur in the power grid.

Another lower-power state adopted in a variety of energy-saving strategies is called hibernate (S3). In this state, the computer is completely turned off, and the application execution context is stored as a file into the hard disk. When an external event interrupts hibernation, the computer is turned on, and the original state is loaded from the hard drive into memory. Computers consume less power in this state because most of the hardware components are turned off. The drawback is that the computers in this state incur a higher latency for getting ready because of the cost of moving the context from disk to memory.

Besides these two power states, another way to save energy is by turning computers off without worrying about the OS state, application contexts, or user data. This behavior refers to the global state G2 in ACPI. The difference of this state compared with hibernation is that it does not keep settings in memory.

These states can be controlled locally by ACPI commands, but in some systems, the ACPI might also be remotely managed using Wake-on-LAN (WoL) [29]. WoL consists of a standard developed by Advanced Micro Devices (AMD) for computers connected to a network to manage energy information. For this, the network card and the motherboard must support WoL.

4. MODELING THE ADVANCED CONFIGURATION AND POWER INTERFACE STATES

While most previous studies are related to at least one ACPI state, none of them explore the cost to go from one state to another. For example, if a given computer goes into the G2 state; that is, it is turned off to meet an energy-saving strategy, how long does it take to turn back on? And how much energy is spent during reboot?

Figure 3 depicts power-agnostic states in a computer and their transitions from a holistic view of a data center, where the requested computers are *busy* and unused computers are *idle*. Under these constraints, it is possible to infer several polices to decide on conditions to enter or leave ACPI states to save energy. However, the trade-off concerning energy consumption and performance to change power states is not considered in current simulators, even though they are fundamental to answer the aforementioned optimization questions.

From the ACPI's point of view, when a computer becomes *busy*, its CPU might be put into one of the load-driven power states (P-states). On the other hand, the computer might enter into an S-state when it becomes idle, reducing the energy consumed by hardware components in both states. This is a very common scenario in laptops and mobile devices to conserve energy. In a more generic sense, the set of ACPI G-/S-states can be correlated as shown in Figure 4.

It is worth noting that all states at some point converge at the G0 state, when the computer is busy in processing a workload. It occurs because any transition must pass through G0, as per the ACPI specification. The weights l and p denote, respectively, the latency (seconds) and power rate (watts) required for a state to be reached. Moreover, the power rate while the computer remains in each state is represented as a transition to the state itself.

Although transitions between states can be measured at discrete time intervals and are represented by different reachable states, they do not present a probabilistic behavior. State transitions are *deterministic based on well-defined and controlled events* within a limited set of states. This deterministic behavior of ACPI's state transitions makes them unsuitable to be modeled via stochastic processes, such as Markov chains. Therefore, according to graph theory, we express this behavior

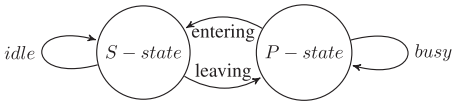

Figure 3. Power-agnostic states of a computer.

DOI: 10.1002/cpe

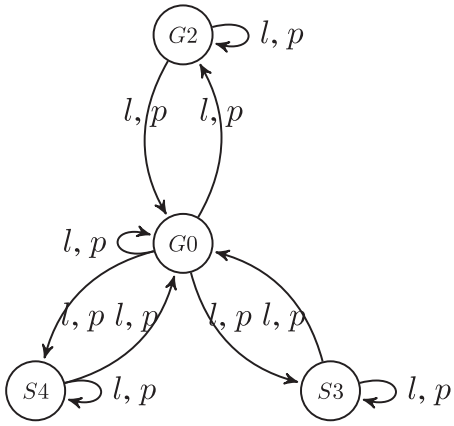

Figure 4. Correlation of advanced configuration and power interface S-/G-states. Edges represent reachability between states and weights the latency and power rate to switch between them.

Table II. Assessments of the power states and their transitions in a computer.

ϵ	$l(\epsilon)$	$p(\epsilon)$	σ_l	σ_p
(G0,G2)	59	108	1.1	0.7
(G2,G0)	81	69	2.1	0.4
(G0,S3)	25	51	1.1	0.5
(S3,G0)	5	91	0.6	0.5
(G0,S4)	101	86	4.9	0.2
(S4,G0)	79	79	1.3	0.9
{G0,G0}	∞	190	N/A	N/A
{G2,G2}	∞	6	N/A	N/A
{S3,S3}	∞	9	N/A	N/A
{S4,S4}	∞	11	N/A	N/A

N/A means the computer does not process any load in sleep mode.
The symbol ∞ means the computer may stay indefinitely in a state.
Latency and power are represented in seconds and watts, respectively.

as $G = (V, A)$, where $V = \{v_i, ..., v_n\}$ is the vertix-set and $A = \{\epsilon_i, ..., \epsilon_m\} \subseteq \{(x, y), \{x, x\} \mid x, y \in V\}$ is the edge-set. From the ACPI standpoint, we define $V = \{G0, G2, S3, S4\}$ and $A = \{(G0, G2), (G2, G0), (G0, S3), (S3, G0), (G0, S4), (S4, G0), \{G0, G0\}, \{G2, G2\}, \{S3, S3\}, \{S4, S4\}\}$. Power ($p$) and latency ($l$) are the weights of each $\epsilon \in A$ such that $F(\epsilon) = ((x, y), \{x, x\})$.

In order to discuss our claims, we conducted a set of preliminary tests in a physical computer to identify relationships between state transitions under different load conditions. We assessed the power rate while the computer was changing from/to G0, G2, S3, and S4 states. In the case of (G0,G0), in which the computer is busy and the CPUs might enter in a load-driven P-state, we scaled them up to the maximum frequency for peak energy consumption. The testbed consists of a computer equipped with two 2.27 GHz Intel Xeon E5520 processors (with 8 cores each), 8M of L3 cache per core, 16 GB of RAM and one NetXtreme II BCM5709 Gigabit Ethernet adapter. The instantaneous power rates were measured via a digital power meter connected directly to the computer's power supply, and the latency was obtained by inspecting power fluctuation during transitions. The measurements for each state and its transitions are shown in Table II.

Latencies and power rates may vary during state transitions. Computers with a large amount of data loaded in memory will probably take more time to be powered off than idle computers. To reflect this, several measurements of latency and power were performed while the computer was

DOI: 10.1002/cpe

under different load conditions in terms of memory and CPU usage, and the measurements were based on the average with their respective standard deviations σ. The total amount of memory allocated was from 10% to 100%, and the CPU load was increased in a core basis, starting from 1 to 16 using the Linpack benchmark [30]. The highest standard deviation was observed in $(G0, S4)$. All data were moved from the memory to disk before entering the S4 state. This caused the latency to vary unpredictably because of the amount of data in memory. We believe this is not an issue because energy-saving strategies in most cases do not change a computer's power states until it is unallocated, so that memory content is cleaned up and space is returned to the OS before any policy is triggered.

Based on our observations, the energy consumed by a computer during a given state transition ϵ is calculated by the integral of the power rate using the first instant time t_0 and the amount of time the computer is changing to a state $t_l(\epsilon)$ as limits:

$$E_\epsilon = \int_{t_0}^{t_l(\epsilon)} p(\epsilon) Dt \tag{4}$$

Given that a computer may have its power state changed many times governed by an energy-saving strategy, then we need a discrete equation to sum the energy consumed by a set of executed transitions. Thus, let $S : S \subseteq E$ be a subset of transitions executed for a period. The total energy consumed by transitions in S is given by

$$E_T = \sum E_\epsilon, \forall_\epsilon \in S \subseteq \{(x, y), \{x, x\} \mid x, y \in V\} \tag{5}$$

Additionally, we also considered the consumption while the computer is in the G0 state, that is, executing some task. The CPU's frequencies may vary dynamically to conserve power in the G0 state by entering into a P-state. Thus, we added to our definition the well-known linear power model proposed by Chen *et al.* [17], which considers CPU usage (α) as input to predict power in P-states:

$$E_{\{G0,G0\}} = \sum_{i=1}^{U} \cdot \int_{t_0}^{t_l} \left[(1-\alpha) P_{FreqIdle_i} + \alpha P_{FreqFull_i} \right] Dt \tag{6}$$

where the CPU power rate while it is idle and full utilization are denoted by $P_{FreqIdle}$ and $P_{FreqFull}$, respectively. The integral limits represent the amount of time the computer remained in G0, and U is the total amount of processing units. Finally, the total energy consumed by transitions in S including the $\{G0, G0\}$ transitions is denoted by

$$E = E_T + E_{\{G0,G0\}} \tag{7}$$

It should be noted that a transition occurs whenever a request arrives and a given computer must return to G0 to serve it. Thus, let $W : W \subseteq S$ be a subset of transitions performed over this condition, w is the user's workload, and t is its execution time. The equation that represents the total workload execution time when the requested computer is not in G0 and the user must wait for a transition ϵ before having its workload placed on that computer is as follows:

$$ET = t(w) + l(\epsilon), \epsilon \in W \tag{8}$$

Finally, the latency-related performance degradation incurred by transitions in W is given by

$$L_T = \sum l(\epsilon), \forall_\epsilon \in W \tag{9}$$

It is remarkable that there is a trade-off between the total energy consumption and performance. The impact of this relationship on real-world scenarios now becomes much clearer. An in-depth study reveals that current energy-saving strategies do not consider this trade-off and that there are

DOI: 10.1002/cpe

environments where these transitions would have a huge influence on energy consumption and user's Service Level Agreement (SLA), such as those that the turnaround time is critical and should never be exceeded. We claim that these influences would be better comprehended through simulation-based evaluations that enable an analysis without any disturbance of the environment.

5. MODEL SIMULATION

Evaluation by simulation allows researchers to evaluate different ACPI states efficiently without being affected by the environment. To this end, we expanded the current CloudSim's power engine to enable more generic energy-saving strategy simulations of cloud computing environments considering host's power states and their transitions. CloudSim's DVFS package provides part of the ACPI architecture (e.g., P-states), which can be expanded to encompass other power states, such as S-states and G-states. We did not see opportunities to implement D-states support, as CloudSim does not implement devices in its physical substrate. Also, the energy consumed by devices, such as hard drives, CD-ROM, and LCD display, is not a concern in cloud computing environments, and the energy-saving strategies usually do not consider this layer of physical components. Nevertheless, equation 5 could be easily adapted for all types of devices.

5.1. Model implementation in CloudSim

The proposed energy model was added to CloudSim's DVFS package developed by Guérout *et al.* [23]. The package already implements and offers part of the ACPI architecture (e.g., P-states), which allowed us to expand from a well-validated power engine. The DVFS package provides a framework to simulate strategies involving only processor features. With new capabilities, it is possible to simulate a wider set of strategies. Therefore, we now have the ability to simulate not only processor states but also strategies involving sleep and global states. In fact, we included subsidies that allow simulation of ACPI G/S-states, as presented in the specification document and described in Section 3.2.

Actually, CloudSim's power engine core is composed of the following entities and objects: *PowerDataCenterBroker*, *PowerDataCenter*, *PowerHost*, *PowerModel*,*VirtualMachine*, and *Cloudlet*. The class *PowerDataCenterBroker* models the broker, which is responsible for mediating between users and service providers. *PowerDataCenter* models the core infrastructure level services (hardware and software) offered by resource providers in a cloud computing environment. Moreover, *PowerHost* models physical hosts. *PowerModel* measures power based on the CPU load. Also, *VirtualMachine* models a virtual machine instance. Lastly, *Cloudlet* is the cloud-based application services (workload). The relationship between the power engine and the new modeled entities and objects can be observed in the diagram in Figure 5.

Regarding object classes, the class *ACPIStateDatas* was modeled to represent the ACPI state data structure. The *power_entering* and *time_entering* attributes refer respectively to the power rate and latency for entering into a state. The *power_leaving* and *time_leaving* attributes refer respectively to the latency and power rate to leave a state. Finally, the latency and the power rate while the host remains in a state are stored in the *power_remaining* and *time_remaining* attributes.

For simulating the incurred latency to go from one state to another and to change the internal power state of the hosts, we designed the *PowerDatacenterACPI* entities. The entity *PowerDatacenterACPI* contains concrete methods conceived to provide ACPI capabilities and abstract methods to be overridden by a derived class, as is the case of the class *PowerDatacenterEnergySavingStrategy*, which in turn implements the desired energy-saving strategy on the *PowerHosts*.

A straightforward strategy would change the host's power state when it becomes idle or busy. The *processChangeHostACPIState* method is responsible for the state transitions; thereby it is called whenever the states of hosts change. For instance, if a state of a host changes from G0 to G2, then the method is called, and the latency associated with the state is used as the delay for creation or destruction of a virtual machine.

User mistakes are controlled by the *ACPITranstionException* class. State transitions are validated per host, and users are notified when a state is unreachable. For instance, a host cannot change

DOI: 10.1002/cpe

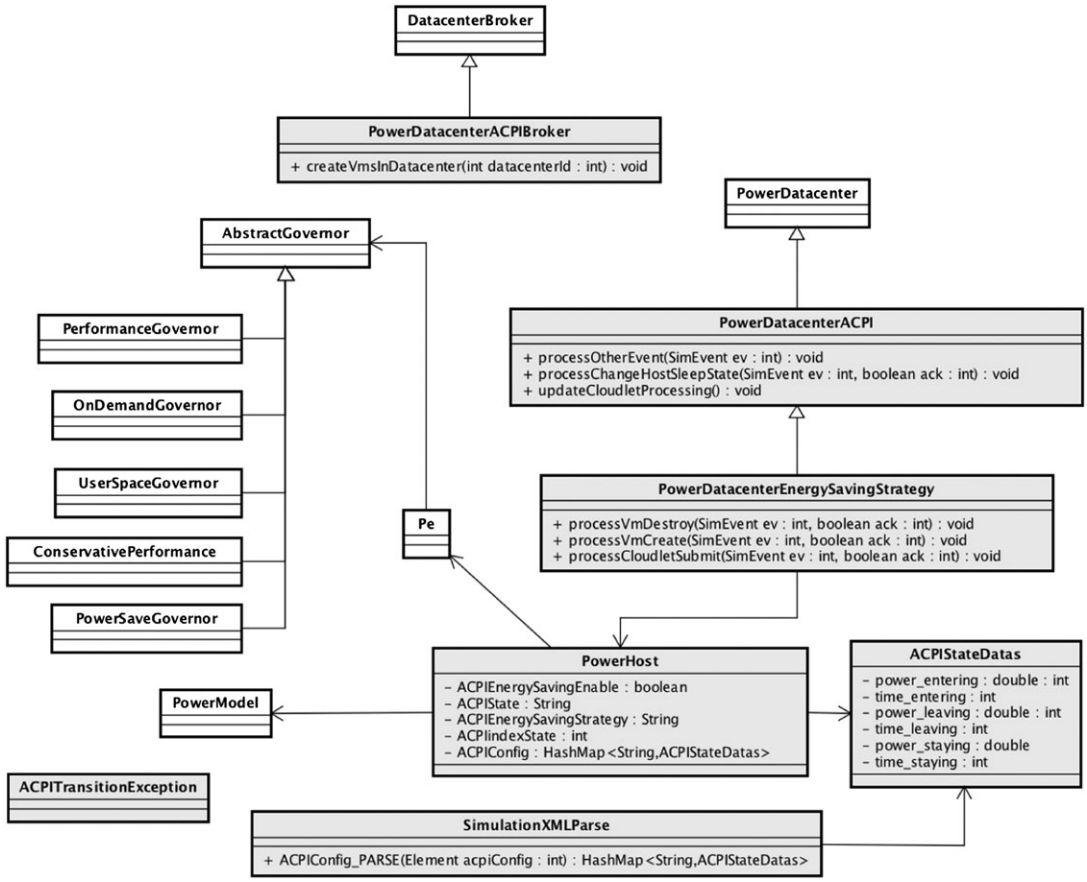

Figure 5. Advanced configuration and power interface implementation in CloudSim's power engine. The new designed entities are highlighted in grey.

between G2 and S3 or should be in the G0 state before entering into a P-state. These requirements are defined in the ACPI specification and were also exhibited previously in Figure 2.

On this platform, it is possible to implement and simulate a wider variety of strategies involving not only those at the processor level (e.g. P0 and P1) but also at host levels, such as G1, G2, and G3.

5.2. Simulation configuration

After implementing the strategy using the designed classes, the user must rely on calibration steps to make the simulator as close as possible to the real environment—the environment that is being simulated. In fact, three steps are essentials to do so, which are as follows:

(1) Identify the set of frequencies the hosts' CPU supports and the power rate (p) for each frequency. The DVFS's userspace governor might be used to scale manually the frequencies while power is measured. Then, power rates given by the hosts at 0% and 100% of CPU utilization, called *FreqIdle* and *FreqFull* for each frequency, should be measured. These results are inputs for the P-states' power model, as defined in Section 4, equation 6;
(2) Measure the power rate (p) to switch from one power state to another. For example, if the strategy considers hosts that enter into the G2 state, then the power rate while the host is shutting down and starting up should be measured. As discussed in Section 3.3, ACPI states can be controlled remotely by WoL, so that the power rate can be measured by a power meter while the transitions are triggered remotely. This calibration is quite important; otherwise, the simulation might not express reliable and precise results;

DOI: 10.1002/cpe

(3) Measure the latency (l) of each transition identified in item 2. For example, if the strategy considers hosts that enter into the G2 state, the hosts' boot time must be measured. A simple way to do it is by analyzing power rate fluctuations during transitions. When the host reaches the G0 state, the power rate tends to stabilize and latency can be measured by analyzing the time while the power rate varied. Another way is by using ICMP packets to identify the exact time that the network subsystem starts to reply with ICMP messages. However, this approach does not apply to all ACPI states, because not all of them turn off the network subsystem. Finally, the user can measure latency through a handmade script installed into the system through which the system's uptime is collected as soon as it becomes available. This calibration is quite essential, because the state changes take some time to complete, as shown in Section 4.

The *ACPIConfigPARSE* method was created in *SimulationXMLPARSE* class to load calibration values from a XML file and make the simulator easier of programming. It prevents the user from designing codes in CloudSim core aimed at not breaking the build.

6. MODEL VALIDATION

Energy-saving strategies employ policies to switch host's power states in different ways. Most large-scale data centers are governed by policies that put hosts into the most common ACPI states: S3, S4, and G2. In this section, the calibration phase necessary to simulate these states is presented. Next, the accuracy of the model incorporated to CloudSim is validated on a single host.

6.1. Setup configuration

All experiments were conducted on the machines presented in Table III. They are equipped with heterogeneous processor architectures and different resource capacities. All machines are interconnected by a Gigabit Ethernet switch. The ACPI and WoL capabilities were enabled in the machines' BIOS. WoL is the technique necessary to remotely manage the machine's power states during the experiments.

We deployed the OpenStack [31] (Havana release) platform on the 10 machines. OpenStack is an open-source cloud platform that has been largely adopted by the industry and has been continuously developed by a strong user community [32]. The simulated resources in CloudSim were analogous to the configurations described in Table III.

6.2. Simulation calibration

Among all of the hosts shown in Table III, there are five distinct architectures that vary in power rates, as can be seen in Figure 6. The trade-off formerly noted during the modeling steps now becomes more noticeable. The deeper the power state, the higher the latency to return from the state. In contrast, the deeper the power state, the lower the energy consumed in the state.

Table III. Configuration of the machines in our Cloud testbed.

Host	Processor	Cores	Clock (Ghz)	Cache (Mb)	RAM (Gb)
1	Intel dual core E5200	2	2.5	2	4
2	Intel dual core E5200	2	2.5	2	4
3	Intel core 2 duo E8400	2	3.1	6	2
4	Intel core 2 duo E8400	2	3.1	6	2
5	Intel Xeon E5520	16	2.7	8	16
6	Intel core i7 3770	8	3.4	8	16
7	Intel core i5 2400	4	3.1	6	8
8	Intel core i5 2400	4	3.1	6	8
9	Intel core i5 2400	4	3.1	6	8
10	Intel core i5 2400	4	3.1	6	8

DOI: 10.1002/cpe

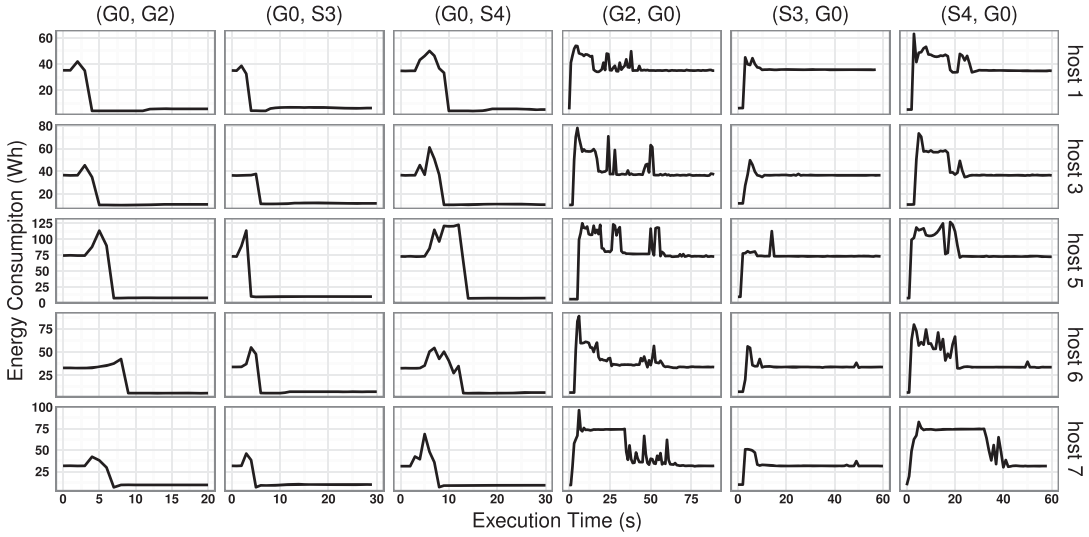

Figure 6. Energy consumption during advanced configuration and power interface state transitions on different architectures.

Table IV. Measurements of the energy consumption in each state for the five distinct hosts.

	Power (W)				
	(G0,G0)		(S4,S4)	(S3,S3)	(G2,G2)
Host	Idle	Full	*p*	*p*	*p*
1	32	70	12	15	5
3	36	82	8	11	5
5	110	190	24	12	5,5
6	32	95	9	13	6
7	31	92	10	13	6

Table V. Measurements of the energy consumption and latencies during state transitions for the five distinct hosts

	Latency (s) × power (W)											
	(G2,G0)		(G0,G2)		(S3,G0)		(G0,S3)		(G0,S4)		(S4,G0)	
Host	*l*	*p*	*l*	*p*	*l*	*p*	*l*	*p*	*l*	*p*	*l*	*p*
1	44	40	4	30	8	36	3	27	7	36	28	42
3	48	46	3	31	9	35	2	28	9	27	21	50
5	81	69	59	108	5	91	25	51	101	86	79	79
6	56	43	5	31	8	35	4	35	9	37	20	55
7	65	55	4	29	7	40	3	30	6	38	42	63

Based on this analysis, we must configure the simulated testbed in CloudSim to obtain more reliable, realistic, and accurate results compared with the real testbed. Therefore, in order to calibrate the simulator with the power rates p (in watts) and latency l (in seconds) values, we measured the ACPI states and their transitions in each architecture. The assessments collected from the hosts in each state are shown in Table IV, and the assessments from the transitions can be seen in Table V.

In addition to the energy consumption, Table IV also shows the power rate in G0 when the hosts' CPU becomes idle or goes up to a full load. The consumption while idle is essential to

DOI: 10.1002/cpe

reproduce/simulate strategies in which the host is unallocated and does not enter to a deeper power state immediately. On the other hand, the consumption while in the full load enables simulation of high-load host allocations. Decisions about energy-saving policies that lead to scenarios like these are influenced essentially by the trade-off we have presented. Furthermore, the latency values were suppressed just because a host can remain indefinitely in a state.

6.3. Aadvanced configuration and power interface validation on one host

To analyze CloudSim's accuracy in a real environment, we simulated a trace on a single host (host 6 in Tables V and IV) to validate each state individually. The trace driven by a strategy that puts idle hosts into the three states (S3, S4, and G2) is presented in Figure 7.

In the beginning, the host was idle and remained in that G/S-state until a request for provisioning a virtual machine was received. Slice (a) means there was a request, and the host must be ready to receive a virtual machine. In this case, a transition to the G0 state has occurred, and the host has begun leaving from its current state. While in G0, the host has started the virtual machine and its processors are no longer idle (CPUs are at the maximum power peak because DVFS is in performance governor mode). This step refers to the slice (b) in the figures. Finally, slice (c) means the virtual machine was unallocated, and the host became idle again; thereby, according to the energy-saving strategy policy, the host had its state switched back to either G-state or S-state. Thus, it has begun transitioning into a state (a state in which the host are consuming less power) and remained there until the next request.

All hosts maintained the same behavior regardless of the power state governed by the strategy, because the events had the same timeslice proportion among them. However, it is worth noting that there were substantial differences regarding energy consumption and latency among the states, as observed in Table VI.

Notable differences in run-time are due to the latency L_T associated with each state transition, and the differences in overall energy consumption E are influenced by the energy consumed by states throughout the trace, as noted by E_T. This simulation has shown an accuracy of 83% compared with the real results.

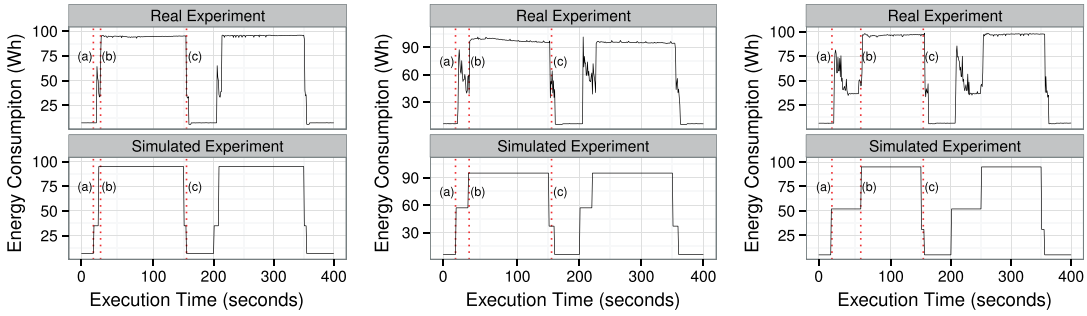

Figure 7. Comparison of the real and simulated energy consumption of all the three advanced configuration and power interface states: S3 (standby), S4 (hibernate), G2 (soft off). Only the first 400 s of the total execution time are shown; however, the same behavior is repeated seven times.

Table VI. Simulation results on one host in all of the three advanced configuration and power interface states.

v	E_T (Wh)	$E_{\{G0,G0\}}$ (Wh)	E	L_T (s)	Run-time (s)
S3	21	26	47	84	1406
S4	10	26	36	207	1529
G2	5	26	31	427	1751

DOI: 10.1002/cpe

7. MODEL EVALUATION

Strategies that put hosts into different power states were necessary to validate the new CloudSim capabilities for more realistic scenarios, and then we implemented and evaluated the straightforward strategy proposed by Alvarruiz *et al.* [9] that we referred to as Green energy-saving strategy. In addition, we also implemented and evaluated a timeout-based strategy widely employed in various works [33–35]. Both strategies reproduce different cloud computing data center behaviors regarding energy-saving policies.

A set of state transitions (denoted by S) was executed for each test case in real and simulated environments, and the simulator's accuracy was quantified by comparing the results. To reproduce the traces in the real OpenStack environment, we developed a set of scripts that implements the strategies by orchestrating hosts' power states according to transitions in S. Because we are only interested in the simulation of the G and S states, we assumed that the hosts CPUs operate at their highest frequency while running user's workloads, ensuring that the hosts are at power consumption peak. The Linux stress tool [36] was used to impose load on and stress the hosts.

Prior to experiments, some constraints have been defined before running the traces: (1) the virtual machines allocate the total amount of the hosts' resources while being careful not to exceed the maximum capacity; (2) a number of virtual machine requests are carried out throughout the traces. After the allocation time has expired, the virtual machine is unallocated and the host is released; (3) the traces start with all hosts in a power state different of G0; and lastly, (4) when the host becomes busy (provisioned virtual machine), the CPU-bound workload boosts the CPUs' frequency up to the maximum supported.

7.1. Green energy-saving strategy

This strategy is proposed by Alvarruiz *et al.* [9] and considers a cloud computing data center composed of hosts that become idle or busy under different periods of time. Under these conditions, the strategy takes into account the following policies that rely on two ACPI states ($V = \{G0, G2\}$):

(1) When a given host becomes idle (no provisioned virtual machine), then it enters into G2 state;
(2) The host remains in G2 until it is requested for a new virtual machine provisioning;
(3) The host returns immediately to the G0 state if it is requested.

Figure 8 shows the policies applied on one host and illustrates iteration among the aforementioned states.

The definition denoted by $E_{\{G0,G0\}}$ quantifies the total energy when the host is in the G0 state because the CPUs' frequencies are governed by DVFS. On the other hand, the host enters into the G2 state when it becomes idle and E_T is calculated. Finally, the testbed in this experiment consists of a cluster of four identical hosts (host 5 in Table III). We set the Kernel-based Virtual Machine (KVM) [37] as the hypervisor under the OpenStack platform.

The trace carried out to analyze CloudSim's accuracy is illustrated in Figure 9. We varied the number of transitions per minute during the 19 752 s to analyze the model's accuracy with several (G0,G2) and (G2,G0) transitions. Only the first 4000 s were plotted, but the interval lying between the second 0 and 2000 was performed 10 times repeatedly. This is why the interval from A to H was sliced only in this time interval.

Slice (A) represents (G2,G0) transitions triggered in all hosts. A set of virtual machine requests was received from the user that led all hosts to become ready to receive the workloads. All hosts were busy in slice (B). They were using the totality of resources, and their processors' units were

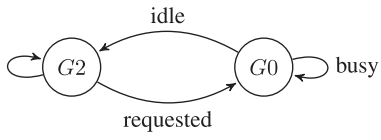

Figure 8. Green energy-saving strategy's policies employed on one host.

DOI: 10.1002/cpe

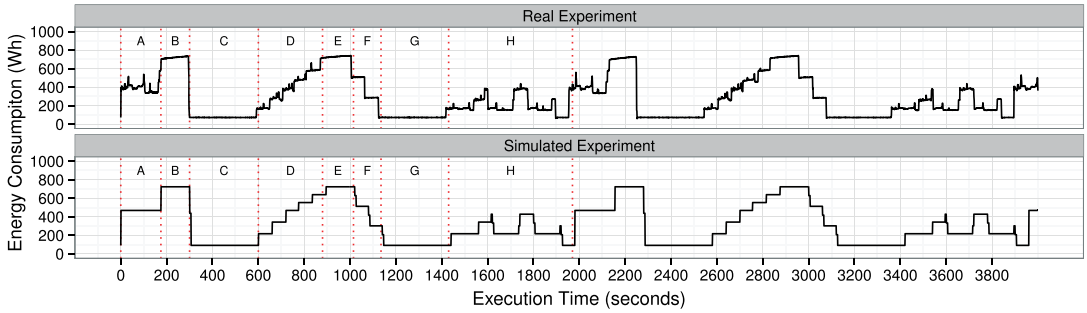

Figure 9. Comparison of energy consumption in real and simulated experiments for the Green energy-saving strategy.

Table VII. Green energy-saving strategy's simulation results.

E_T	$E_{\{G0,G0\}}$	E	L_T	Run-time
310 Wh	1221 Wh	1531 Wh	5127 s	19740 s

working at the highest power rate. The virtual machines were unallocated in slice (C), and the hosts went into G2 state to save energy. In slice (D), OpenStack started receiving virtual machine requests in 60-second intervals that made the hosts are gradually switched from the G2 to G0 state. The hosts were busy processing the workload in slice (E). They started entering to G2 state as the virtual machine allocation time begins to expire. Finally, the hosts become idle in slice (G).

The Green energy-saving strategy's simulation results had an accuracy of 95%. This shows that the simulator can correlate latency with power rate in the states and during their transitions. Finally, Table VII outlines the simulation results.

7.2. *Timeout strategy*

The Green strategy relies only on two power states ($V = \{G0, G2\}$). This new experiment expands our validations to a scenario that covers other states presented in this work. For this purpose, we evaluated a well-known timeout-based strategy [33–35] that uses four power states ($V = \{G0, S3, S4, G2\}$):

(1) When the host in the G0 state becomes idle, it enters into the S3 state;
(2) The host returns immediately to the G0 state if it is requested; and
(3) The host enters successively to a lower-power state if the timeout expires.

Figure 10 shows the policies applied on one host and illustrates iteration among the aforementioned states.

As we saw, the deeper the power state, the longer it takes to return from the state. On the other hand, the deeper the state, the lower the energy consumption. Thus, state transitions should be performed conservatively because it is less costly, in terms of latency, to 'wake up' a host from S3 than a host from G2. Hence, the strategy imposes a priority order to aid in deciding which host

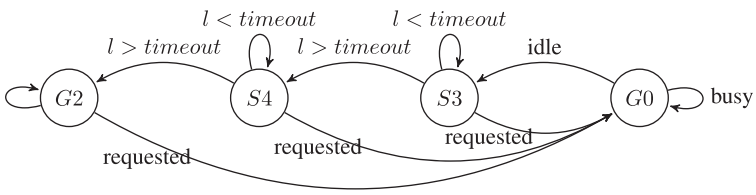

Figure 10. Timeout strategy's policies employed on one host.

DOI: 10.1002/cpe

should be returned to G0 when a virtual machine request arrives. The hosts are ordered by priorities based on their current power state. Deeper states have lower priority over shallow states. Based on this, when a new request arrives, hosts in S3 will be chosen for allocation followed by those in S4 and finally those in G2.

The simulation was carried out based on the same trace adopted in [38]. We deployed the OpenStack on the 10 hosts described in Table III. We also changed the underlying virtualization technology to experiment our model on an alternative virtualization architecture. Thus, we installed the Linux Container (LXC) container-based system [39] as a representative operating system-level virtualization system. The timeout value was set to 300 s based on the state-of-the-art works [40–42]. Finally, when idle hosts reach the timeout value, they are placed gradually in lower-power states as depicted in Figure 10. Figure 11 shows a comparison of the real versus the simulated experiments.

The simulation achieved an accuracy of 94%. We believe part of this difference is due to granularity in watts measured by the power meter when compared with power values configured in CloudSim. This difference becomes evident when we look at the variance spikes in the real experiment and a linear behavior when these spikes are represented in the simulated experiment.

Although Figure 11 shows the total energy consumed for each time interval, the state each host is in during each of these time intervals cannot be seen. Figure 12 shows the percentage of hosts in each state. This enables monitoring of the host state distributions throughout the simulated experiment and estimating the accumulated energy at every moment. Finally, we outlined the simulation results for 150, 300 and 600-s timeout values in Table VIII.

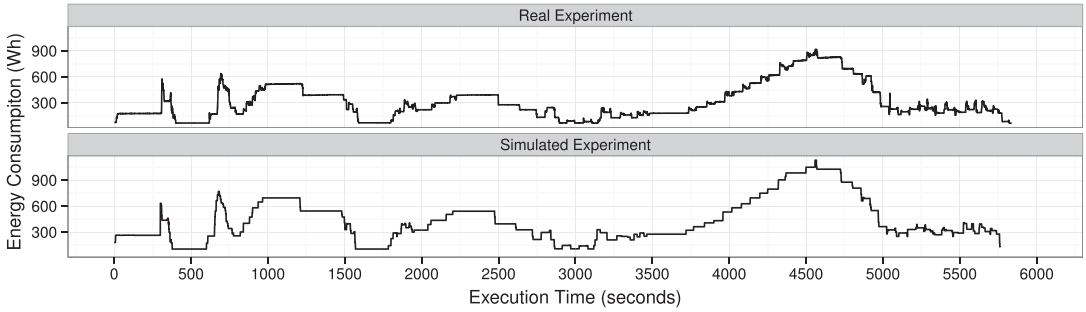

Figure 11. Comparison of energy consumption in real and simulated experiments for the timeout energy-saving strategy.

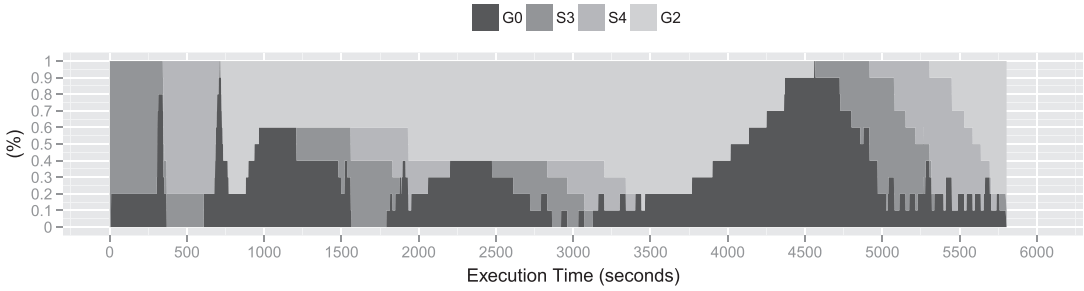

Figure 12. Percentage of hosts in each power state during the simulation.

Table VIII. Timeout strategy's simulation results.

Timeout (s)	E_T (Wh)	$E_{\{G0,G0\}}$ (Wh)	E (Wh)	L_T (s)	Run-time
150	118	418	536	973	5764
300	130	418	548	852	5764
600	182	418	580	609	5764

DOI: 10.1002/cpe

Differences in timeout values reflect strongly in energy consumption and latency. It occurs because the number of hosts reaching lower-power states varies considerably. It is easy to see that high-throughput clusters would suffer more impact on latency for lower timeout values. In counterpart, the energy consumption is reduced. This is a typical case study in which the strategy could be refined to improve scheduling performance. Based on the simulation results, we could suggest a timeout adaptive solution to balance the number of hosts in lower-power states based on the requests arrival time interval.

8. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we proposed an improved energy model for the ACPI power states and showed that not only these states offer different energy-saving levels but also that state transitions consume energy and impact on performance.

Evaluation of energy-saving strategies through simulation is of paramount importance before implementing them in a production data center. The definitions we have presented provide fundamental information to quantify the trade-off between the energy consumption and performance and assist in the analysis/decision on which strategy fits better in the environment based on the number of available hosts and the cloud provider's throughput.

As we have presented, the latency to create and destroy virtual machines might eventually impact the end-user's satisfaction and violate service-level agreements. In such cases, the S3 (*standby*) state would be the most suitable because it incurs a lower latency to return the hosts to the G0 state. On the other hand, if impact on energy consumption is a concern and cannot be disregarded, then an energy-saving strategy that uses the G2 (*soft off*) state should be taken into account. Yet, if the latency and energy consumption reflect end-user dissatisfaction and also concern in a green cloud computing environment, a strategy that uses the S4 (*hibernate*) state would be a good choice.

Our energy model was implemented in CloudSim and validated real-based and simulation-based. We also evaluate the implementation of two energy-saving strategies in CloudSim. In our preliminary results, we obtained a very high accuracy, with a standard deviation of at most 6%, compared with the same experiments running in a real testbed.

Therefore, we strongly believe that the inclusion of ACPI support in CloudSim with the implementation of our more precise energy model expands its applicability even more. This leads also to a better understanding of the cost-benefit trade-offs involved in changing states to save energy, thereby allowing a more accurate simulation and analysis of a wide range of energy-saving strategies in cloud environments, combining the consolidation of virtual machines, DVFS, and other less explored ACPI states. The CloudSim ACPI package is available in [43].

REFERENCES

1. Buyya R, Broberg J, Goscinski AM. *Cloud Computing Principles and Paradigms*. Wiley Publishing: Hoboken, NJ, 2011.
2. Rossi FD, Conterato M, Ferreto TC, De Rose CAF. Evaluating the trade-off between DVFs energy-savings and virtual networks performance. *Proceedings of the Thirteenth International Conference on Networks*, ICN '14, IARIA, Nice, France, 2014; 285.
3. Freeh VW, Lowenthal DK, Pan F, Kappiah N, Springer R, Rountree BL, Femal ME. Analyzing the energy-time trade-off in high-performance computing applications. *IEEE Transactions on Parallel and Distributed Systems* 2007; **18**(6):835–848.
4. Berl A, Gelenbe E, Di Girolamo M, Giuliani G, De Meer H, Dang MQ, Pentikousis K. Energy-efficient cloud computing. *The Computer Journal* 2010; **53**(7):1045–1051.
5. Nrdc: Data center efficiency assessment. (Available from: http://www.nrdc.org) [accessed on 29 October 2015].
6. Niyato D, Chaisiri S, Sung LB. Optimal power management for server farm to support green computing. *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, IEEE Computer Society, Washington, DC, USA, 2009; 84–91.
7. Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency Computing: Practice and Experience* 2012; **24**(13):1397–1420.

DOI: 10.1002/cpe

8. Duy TVT, Sato Y, Inoguchi Y. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010*, Atlanta, USA, April 2010; 1–8.
9. Alvarruiz F, de Alfonso C, Caballer M, Hernandez V. An energy manager for high performance computer clusters. *IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2012*, Springer US. Madrid, Spain, July 2012; 231 –238.
10. Isci C, McIntosh S, Kephart J, Das R, Hanson J, Piper S, Wolford R, Brey T, Kantner R, Ng A, Traore A, Frissora M. Agile, efficient virtualization power management with low-latency server power states. *SIGARCH Computer Architecture News* 2013; **41**(3):96–107.
11. Grover A. Modern system power management. *Queue* 2003; **1**(7):66–72.
12. Coutinho F, V. de Carvalho LA. Strategies based on green policies to the grid resource allocation. *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, PACT '12, ACM, New York, NY, USA, 2012; 487–488.
13. Yang CT, Wang KC, Cheng HY, Kuo CT, Hsu CH. Implementation of a green power management algorithm for virtual machines on cloud computing. *Proceedings of the 8th International Conference on Ubiquitous Intelligence and Computing*, UIC'11, Springer-Verlag, Berlin, Heidelberg, 2011; 280–294.
14. McIntosh-Smith S, Wilson T, Crisp J, Ibarra AÁ, Sessions RB. Energy-aware metrics for benchmarking heterogeneous systems. *ACM SIGMETRICS Performance Evaluation Review* 2011; **38**(4):88–94.
15. Beloglazov A, Buyya R. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, MGC '10, ACM, New York, NY, USA, 2010; 4:1–4:6.
16. Bohra A, Chaudhary V. Vmeter: Power modelling for virtualized clouds. *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, IEEE Computer Society, Atlanta, USA, April 2010; 1–8.
17. Chen Q, Grosso P, Veldt KVD, Laat CD, Hofman R, Bal H. Profiling energy consumption of VMS for green cloud computing. *Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, DASC '11, IEEE Computer Society, Washington, DC, USA, 2011; 768–775.
18. Bruneo D, Longo F, Puliafito A. Evaluating energy consumption in a cloud infrastructure. *Proceedings of the 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, WOWMOM '11, IEEE Computer Society, Washington, DC, USA, 2011; 1–6.
19. Salehi M, Krishna P, Deepak K, Buyya R. Preemption-aware energy management in virtualized data centers. *IEEE 5th International Conference on Cloud Computing (CLOUD), 2012*, IEEE Computer Society, Honolulu, Hawaii, USA, June 2012; 844–851.
20. Haizea: an open-source VM-based lease management architecture. (Available from: http://haizea.cs.uchicago.edu/) [accessed on 6 November 2015].
21. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 2011; **41**(1):23–50.
22. Kolpe T, Zhai A, Sapatnekar SS. Enabling improved power management in multicore processors through clustered dvfs. *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, IEEE Computer Society, Dresden, Germany, 2011; 1–6.
23. Guérout T, Monteil T, Costa GD, Calheiros RN, Buyya R, Alexandru M. Energy-aware simulation with DVFs. *Simulation Modelling Practice and Theory* 2013; **39**(0):76–91.
24. Kliazovich D, Bouvry P, Khan S. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing* 2012; **62**(3):1263–1283.
25. The network simulator ns-2. (Available from: http://www.isi.edu/nsnam/ns/) [accessed on 22 October 2015].
26. Núñez A, Váquez-Poletti J, Caminero A, Castañé G, Carretero J, Llorente I. icancloud: a flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing* 2012; **10**(1):185–209.
27. Lim SH, Sharma B, Nam G, Kim EK, Das C. Mdcsim: A multi-tier data center simulation, platform. *IEEE International Conference Oncluster Computing and Workshops, 2009. CLUSTER '09*, IEEE Computer Society, New Orleans, LA, August 2009; 1–9.
28. Delgado B, Karavanic K. Performance implications of system management mode. *IEEE International Symposium on Workload Characterization (IISWC), 2013*, Portland, OR, September 2013; 163–173.
29. Gil-Martinez-Abarca J, Macia-Perez F, Marcos-Jorquera D, Gilart-Iglesias V. Wake on LAN over internet as web service. *Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on*, IEEE Computer Society, Prague, Czech Republic, September 2006; 1261–1268.
30. Dongarra JJ, Luszczek P, Petitet A. The linpack benchmark: past, present and future. *Concurrency and Computation: Practice and Experience* 2003; **15**(9):803–820.
31. Openstack: the open-source software platform for cloud-computing. (Available from: https://www.openstack.org/) [accessed on 1 November 2015].
32. Pepple K. *Deploying Openstack*. "O'Reilly Media, Inc.": Sebastopol, CA, 2011.
33. Augustine J, Irani S, Swamy C. Optimal power-down strategies. *SIAM Journal on Computing* 2008; **37**(5): 1499–1516.

DOI: 10.1002/cpe

34. Meisner D, Gold BT, Wenisch TF. Powernap: Eliminating server idle power. *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, ACM, New York, NY, USA, 2009; 205–216.
35. Ponciano L, Brasileiro F. On the impact of energy-saving strategies in opportunistic grids. *2010 11th IEEE/ACM International Conference on Grid Computing (GRID)*, Brussels, Belgium, October 2010; 282–289.
36. Stress - tool to impose load on and stress test systems. (Available from: http://linux.die.net/man/1/stress) [accessed on 4 November 2015].
37. Habib I. Virtualization with KVM. *Linux Journal* 2008; **2008**(166):8.
38. Pucher A, Gul E, Wolski R, Krintz C. Using trustworthy simulation to engineer cloud schedulers. *IEEE International Conference on Cloud Engineering (IC2E), 2015*, Tempe, AZ, March 2015; 256–265.
39. Linux containers. (Available from: https://linuxcontainers.org/) [accessed on 4 November 2015].
40. Energy star* version 5.0 system implementation whitepaper. (Available from: http://www.energystar.gov) [accessed on 25 October 2015].
41. Reich J, Goraczko M, Kansal A, Padhye J. Sleepless in seattle no longer. *Proceedings of the 2010 USENIX Conference on Usenix Annual Technical Conference*, USENIXATC'10, USENIX Association, Berkeley, CA, USA, 2010; 17–17.
42. Lammie M, Brenner P, Thain D. Scheduling grid workloads on multicore clusters to minimize energy and maximize performance. *IEEE/ACM International Conference on Grid Computing, 2009 10th*, Las Vegas, Nevada, October 2009; 145–152.
43. Cloudsim's acpi package. (Available from: https://github.com/miguelxvr/cloudsimacpi) [accessed on 25 December 2015].

DOI: 10.1002/cpe