Security supportive energy-aware scheduling and energy policies for cloud environments

Damián Fernández-Cerero [a], Agnieszka Jakóbik [b], Daniel Grzonka [b], Joanna Kołodziej [b], Alejandro Fernández-Montes [a]

[a] Department of Computer Languages and Systems, University of Seville, Spain
[b] Department of Computer Science, Cracow University of Technology, Poland

HIGHLIGHTS

- We propose energy-efficiency strategies for task scheduling and hibernating VMs.
- We combine energy and time-based criteria in order to sleep idle resources.
- We take into account several security constraints in our model.
- The effectiveness of the proposed model has been confirmed by simulation experiments.

ABSTRACT

Cloud computing (CC) systems are the most popular computational environments for providing elastic and scalable services on a massive scale. The nature of such systems often results in energy-related problems that have to be solved for sustainability, cost reduction, and environment protection.

In this paper we defined and developed a set of performance and energy-aware strategies for resource allocation, task scheduling, and for the hibernation of virtual machines. The idea behind this model is to combine energy and performance-aware scheduling policies in order to hibernate those virtual machines that operate in idle state. The efficiency achieved by applying the proposed models has been tested using a realistic large-scale CC system simulator. Obtained results show that a balance between low energy consumption and short makespan can be achieved.

Several security constraints may be considered in this model. Each security constraint is characterized by: (a) Security Demands (SD) of tasks; and (b) Trust Levels (TL) provided by virtual machines. SD and TL are computed during the scheduling process in order to provide proper security services.

Experimental results show that the proposed solution reduces up to 45% of the energy consumption of the CC system. Such significant improvement was achieved by the combination of an energy-aware scheduler with energy-efficiency policies focused on the hibernation of VMs.

Keywords:
Cloud computing
Energy efficiency
Independent task scheduling
Genetic algorithms
VM hibernating
Cloud security

1. Introduction

Virtualization of resources and Containerization Platforms, such as Docker, have improved the resource efficiency in Cloud Computing (CC) environments. These strategies allow the execution of several heterogeneous services, such as MapReduce frameworks, web servers, databases, and multi-purpose virtual machines on the same physical resources. Although both performance and energy efficiency in CC environments depend mainly on hardware features, proper scheduling policies may significantly shorten task completion time, which can lead to the reduction of the energy consumption in CC environments [29,33].

CC systems should ensure the appropriate security level for every task deployed on the system [37], and must provide tools for CC operators to develop security frameworks that suit their use cases [38].

In this paper, we defined various energy-efficient optimization strategies for multi-purpose central, monolithic schedulers in CC systems. The energy efficiency is achieved through dedicated policies that hibernate virtual machines that run in an idle state. Moreover, we present a new model for the calculation of the energy consumption of security operations. Based on this model, CC operators are able to select one of the possible security levels.

This information enables users to set longer or shorter keys for cryptographic procedures. Such key-scaling related services are available in Amazon Cloud, RackSpace, OpenStack, and Google Clouds [1,3–5].

The presented model may be used in any High-Performance Computing system that requires the assignation of tasks to computing units. The computing units used in this work, thus, virtual machines, are characterized by their computing capacity. This model could be adapted to work with any type of computing unit characterized by its computational power, such as those used in edge computing networks, grid computing systems, systems based on micro-containers, and small data centers.

The paper is organized as follows. In Section 2, we present the state of the art in measurement of energy for virtualized environments and optimization of energy consumption in CC. In Section 3, we present various approaches and methods: (a) a methodology for estimation of power consumption of virtual machines in CC; (b) a Batch Scheduling problem in CC with security criteria; (c) computation of the total energy consumed by a given task in a schedule; (d) a multi-objective scheduling problem with energy consumption and security; and (e) energy-efficiency policies based on hibernating virtual machines are presented. In Section 4, the experimental environment and scenarios, where the two more representative energy-efficiency strategies have been implemented, are described. We evaluate our models through extensive realistic simulation. Achieved results are presented and analyzed in Section 5. Finally, the paper is summarized in Section 6.

2. Related work and progress beyond the state-of-the-art

Several strategies have been developed over last years for the estimation of energy consumption of virtual machines in Cloud Computing systems. The power requirements of physical servers in a cluster can be measured by the means of established procedures [35,46,47] focused on measuring the utilization of microprocessors. The measurement process is more complex when virtual machines are considered [16,53].

The VM energy consumption may be computed in terms of the CPU, memory, and IO utilization, as proposed by Li, et al. in [54].

In [13], the virtual machine energy consumption was computed according to hardware performance results collected from various components, mainly the CPU-related ones. The memory utilization is considered in the approach proposed by Krishnan in [53]. The energy consumption of both network interface cards and hard drives was also taken into account in the model presented by Wassmann et al. in [68].

In addition, a linear model based on nine independent parameters was proposed by Bertran et al. in [12] in order to measure virtual machines energy consumption. These parameters were, among others, the first level cache activity and the number of accesses to the first level cache per cycle.

On the other hand, a Gaussian Stochastic Mixture model was proposed in [18] by Dhiman in opposition to the aforementioned linear mathematical models with independent parameters. However, none of the proposed strategies are sufficient to deal with realistic cloud virtual resource allocation and scheduling problems [59].

Various tools aiming to compute VM power consumption have been proposed in an isolated way from cloud platforms. These algorithms, such as FitGreen [20], Julemeter [44], and the algorithm proposed by Murwantara [57], need special configurations to access the hardware layer. Hence, they can only be deployed as an external framework at the cloud provider or Infrastructure as a Service level.

As the importance of CC rises, the energy efficiency of these infrastructures becomes more and more important. These facilities, which consume as much energy as many factories, are responsible for approximately 1.5% of global energy consumption [52].

The strategies developed for optimization of energy consumption in CC may be classified into three major categories:

- **Cooling strategies**. The goal of these strategies is the reduction of the energy consumption of chillers, which represents an important part of the total energy used by a data center. A dynamic thermal management system at the data center level was proposed by Sharma et al. [65]. Rising the data center operating temperature was proposed by El-Sayed et al. [21], whereas Gao et al. extensively evaluated the risks related to this approach [28]. On the other hand, Zimmerman et al. proposed the reutilization of the wasted heat in order to propose a hot water-cooled data center prototype [70]. A multi-stage outdoor air-enabled cooling system composed of a water-side economizer, an air-side economizer, and mechanical cooling was proposed by Kim et al. [50].
- **Hardware-related strategies**. Many hardware-based models have been proposed in order to achieve high energy-conservation levels. Dynamic Voltage Frequency Scaling (DVFS) model is one of the most popular approaches. Miyoshi et al. evaluated benefits of using CPU DVFS [56], while David et al. applied this technique to memory components [17]. The replacement of mechanical components, such as HDDs, with non-mechanical devices, such as SSDs, was proposed by Andersen et al. [8]. Regarding the power supply, a dynamic and non-uniform global power-allocation model among nodes was proposed by Femal et al. [23].
- **VM consolidation and migration**. Several strategies have been developed in order to schedule VMs and redistribute them to reduce the energy consumption. Beloglazov et al. [10] propose a resource management system focused on the minimization of the energy consumption by the utilization of VM allocation and migration policies. This work is extended by the proposal of several heuristics for dynamic consolidation of VMs in [11].

While many of these strategies have been adopted by companies, such as Google, Microsoft, and Amazon, there is another area of research that has been barely implemented on CC systems: the achievement of power-proportional systems by turning off idle resources. The idea is that the energy consumption of CC systems should be proportional to workload requirements, which are hardly ever stable.

There are some reasons that prevent the shut-down of machines that run in an idle state, including: the fear of any change that could break operational requirements [25], the complexity and heterogeneity of all the subsystems involved, and the fast development of new systems and paradigms that could break the established standards and systems. However, keeping servers underutilized or in an idle state is highly inefficient from an energy-efficiency perspective.

Much effort has been made by the research community in order to hibernate underutilized resources. A power-proportional distributed file based on the partition of data centers according to redundant data replicas was proposed by Amur et al. [7]. In such systems, servers that store redundant replicas of data may be switched off. On the other hand, Kaushik and al. proposed in [48] a variant of Hadoop Distributed File System that partitions data centers into zones according to the data usage, which enables servers that store low-used data to be shut down.

Other approaches have been proposed for small mobile systems, such as Virtual Backbone Scheduling [69], and multi-flow multicast transmission [66]. These strategies are well-known in

wireless networks and sensor networks [22] environments. However, the described approaches are not easily applicable to CC systems, since the shut-down of CC servers is more complex and expensive than in the aforementioned mobile systems.

The novelty of the research presented in this paper is the combination of the following two different approaches for the improvement of energy efficiency into one model: (a) an energy-aware scheduler that assigns tasks to VMs according to security demands; and (b) a set of energy-efficiency policies that hibernate underutilized resources, based on the energy policies presented in [26,27] for Grid Computing systems. These energy-efficiency policies have been evolved in order to be applied to CC systems.

Major contributions of this paper include:

1. The proposal of a task service model combining a security-aware task scheduler with a set of energy-efficiency policies.
2. The implementation and testing of the proposed model by using a realistic CC simulator.
3. The analysis of the impact of the proposed algorithms on the task processing flow and the energy consumption of the CC system.

Moreover, we developed a theoretical model for the schedule of tasks according to the energy consumption of the security operations related to tasks.

3. Approaches for energy saving and security issues in CC environments

3.1. VM power consumption

The construction of a model for the energy consumption of virtual machines in CC systems is not straightforward. It depends on several elements and processes, including the virtualization process. However, the power consumption of various components, such as microprocessors, memory, devices, hard drives, and networks may be measured by the means of frameworks like Watts UP PRO Power, and APIs like Amazon Cloud CloudWatch metrics [2].

Moreover, models of energy consumption for virtual machines may be defined as an extension of those applied to physical servers, as long as the virtual machines features are taken into consideration.

Let P_{Static} denote the power a server required to run all the tasks that a VM needs to be ready for work. $P_{Virtual}$ denotes the dynamic power used by VMs hosted by that machine. The overall server power consumption may be described as follows [40]:

$$P_{Phys} = P_{Static} + \sum_{i=1,\ldots,m} P(VM_i) = P_{Static} + P_{Virtual}, \quad (1)$$

where $P(VM_i)$ denotes the energy consumed by the ith virtual machine and m is the number of available VMs. This value is estimated by the means of several approaches. The non-observable parameter $P(VM_i)$ is derived from the observable parameter P_{Phys}.

These methods are mathematical models that consider the power-related resources as independent parameters. Several samples of P_{Phys} are typically collected to estimate the $P(VM_i)$ parameter. This data can be collected by following a black-box approach, i.e. by using a virtual machine hypervisor. On the other hand, a proxy may be deployed in each VM if a white-box strategy is to be used to collect this data [34].

In this work we follow the approach presented in [13]. This means that the energy consumption of virtual machines is based on VM states (working, idle, or hibernated).

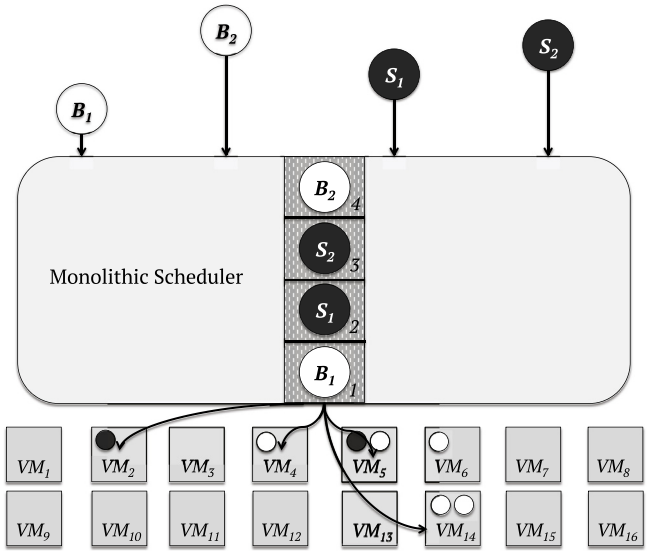


Fig. 1. Single-path scheduling workflow, *B* - Batch type task, *S* - Service type task, *M* - Virtual Machine.

3.2. Monolithic scheduling with a central scheduler

Conceptually, a monolithic scheduler is an omniscient unit responsible for all scheduling decisions, for the allocation all resources, and for maintaining the task deployment process. In this model all workloads are governed by the same scheduler and all tasks are processed by the same scheduling logic [64]. The scheduling algorithm applies a set of heuristics according to tasks requirements, then deploys the tasks on the chosen resources and updates the system state, as illustrated in Fig. 1. Monolithic schedulers usually implement complex scheduling algorithms in order to fulfill various workload types. In this work we consider two types of tasks:

- **Batch tasks**: This type of workload is composed of several independent tasks that can be processed in parallel. Tasks arrive at the system at the same time. Execution of a batch is completed when all of the tasks are finished. After that the whole batch may be processed by another service, stored, or send back to the end user. MapReduce jobs are an example of batch tasks.
- **Service tasks**: This type of workload is composed of long-running tasks. As opposed to the batch tasks, these tasks have no determined end, but are submitted by an operator (or an automated equivalent) and are killed when they are no longer required. Web server instances or service instances, such as BigTable [15], are good examples of service tasks.

In addition to the classical scheduling-related challenges, like minimizing the time a task waits in a queue, satisfying task constraints, respecting priorities, fulfilling end-user SLAs, etc., the ever-growing use of the Cloud Computing paradigm and large-scale web services add several new challenges, such as: (a) scalability; (b) flexibility; (c) scheduling algorithms complexity; and (d) environment fragmentation. These challenges have been addressed by developing new distributed approaches and the scheduling process, such as: (a) shared state scheduling frameworks (e.g. Google Omega [64]); (b) two-level scheduling frameworks (e.g. Mesos [36]); (c) distributed scheduling frameworks (e.g. Sparrow [58]); and (d) hybrid scheduling frameworks (e.g. Mercury [45]).

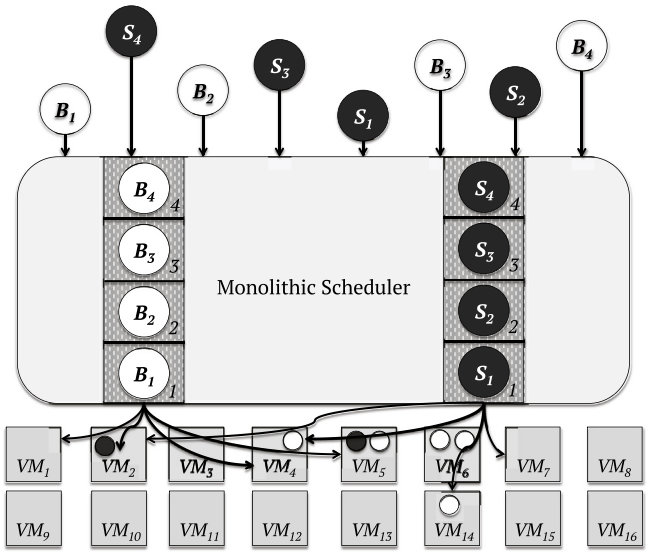


Fig. 2. Multi-path scheduling workflow.

However, for most usual scenarios, such as those present in low and mid-size CC infrastructures up to approximately 10,000 machines, monolithic scheduling frameworks, such as Google Borg [14], are still the best and simplest option.

Two monolithic scheduling approaches are taken into consideration in this paper:

- **Single-path**: This scheduling strategy uses a single scheduling path for every task in the workload, as shown in Fig. 1.
- **Multi-path**: This scheduling strategy uses several scheduling paths by taking advantage of internal parallelism and multi-threading to solve head-of-line blocking and scalability issues, among others. In this work, the multi-path scheduling process represents a system composed of two scheduler paths. The first scheduler path performs the scheduling logic related to batch tasks, whereas the second one is responsible for the scheduling logic related to service tasks. In this approach, any given service task would only need to wait in queue until all previous service tasks are scheduled, since they are independently scheduled, as shown in Fig. 2.

3.2.1. Batch task scheduling considering security demands

In this work, we consider the problem of Independent Batch Scheduling in large Cloud Computing systems. Fig. 3 shows the workflow of the simulated environment, which is composed of the following processes: (a) generation and collection of tasks; (b) task scheduling; (c) task execution; (d) results storage; (e) communication with end-users; and (f) management of the security issues related to all the aforementioned processes.

However, a single batch may contain tasks that require different security levels: e.g. the process of an open-access free stock and the process of clinical images of a hospital. The security demands of tasks were introduced in order to meet these security-related requirements [31,32,42]. The scheduler computes these security demands by implementing a security demand vector that represents the security requirements of the tasks:

$$SD = [sd_1, \ldots, sd_n], \tag{2}$$

where sd_j is specified by the jth task in the batch. On the other hand, different computing units may offer different security services and levels. Amazon Cloud offers high security standards, whereas a private Cloud with an older version of software may offer a lower security level. To reflect this situation, the following trust level vector is introduced:

$$TL = [tl_1, \ldots, tl_m]. \tag{3}$$

It represents the security capacities of all VMs in the system. All the parameters assume values in the range [0,1], where 0 means the lowest security level for a task and the least trusted VM. A particular task will be scheduled to a VM which offers a security level greater or equal than that demanded by the task.

In order to achieve an effective and efficient scheduling process, the previously developed Non-Deterministic Central Scheduler based on a Genetic Algorithm [30,41,42] has been chosen as the main scheduling policy for the monolithic scheduler. In addition to the aforementioned makespan-focused Genetic Algorithm, a new criterion that takes the energy consumption of every task into account is considered in this paper. The developed scheduling policy relies on an Expected Time to Compute (ETC) matrix [51], adapted to virtual machines (ETC_V). The ETC_V matrix can be defined as follows [39]:

$$ETC_V = [ETC_V[j][i]]_{j=1,\ldots,n;i=1,\ldots,m} \tag{4}$$

where

$$ETC_V[j][i] = wl_j/cc_i, \tag{5}$$

where cc_i denotes the computational capacity of the ith virtual machine and wl_j is the workload of the jth task; n and m represent the number of tasks and number of virtual machines, respectively.

Security demands involve additional security operations that must be performed before or after task execution. The possible security operations in the CC system are denoted by a padlock icon in Fig. 3. Security issues may require additional computing time. For this reason, we used an extended version of the ETC_V matrix — $SBETC$ (Security Biased Expected Time to Compute) matrix. This matrix takes the additional security bias (SB) parameter b into consideration in order to represent the time spent for security operations [42]:

$$b(sd_j, wl_j, tl_i, cc_i). \tag{6}$$

All the biases give the matrix representation:

$$SB[j][i](SD, TL) = [b(sd_j, wl_j, tl_i, cc_i)], \tag{7}$$

where SD and TL denote the security demand vector (see Eq. (2)), and the trust level vector (see Eq. (3)) for the VMs in the system, respectively.

The ETC_V matrix can be evolved to the Security Biased Expected Time to Compute ($SBETC$) matrix when the security biases are considered:

$$SBETC[j][i](SD, TL) = wl_j/cc_i + b(sd_j, wl_j, tl_i, cc_i) \tag{8}$$

$$SBETC(SD, TL) = SB(SD, TL) + ETC_V. \tag{9}$$

The main goal of the scheduling and allocation processes is to find an optimal solution for the specified criteria. Among all batch workload scheduling process factors, the makespan is considered the main objective. It can be described as follows:

$$C_{\max} = \min_{S \in Schedules} \left\{ \max_{j \in Tasks} C_j \right\}, \tag{10}$$

where C_j is the jth task completion time. $Tasks$ is the set of tasks in the batch of task, while $Schedules$ represents the set of all possible schedules that may be generated for the tasks of that batch of task, as illustrated in Fig. 4.

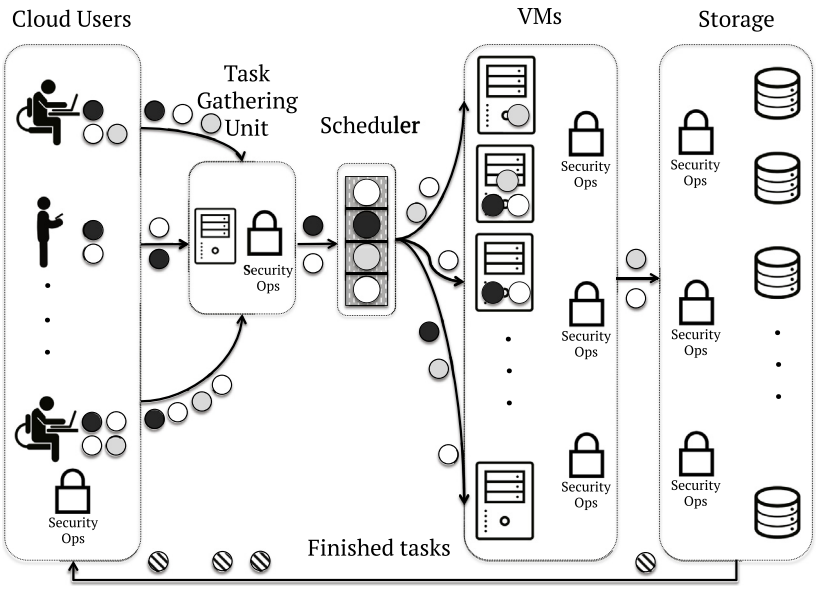


Fig. 3. Cloud computing system workflow, Security Ops — additional security operations and procedures.

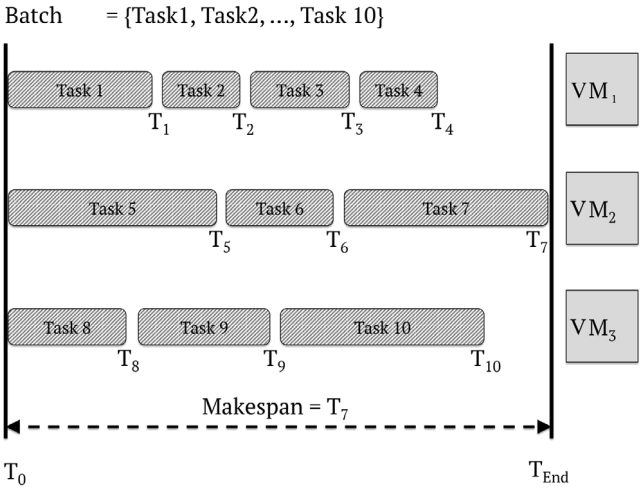


Fig. 4. Makespan measuring workflow.

3.3. Genetic algorithm

The scheduling of tasks in cloud-computing data centers constitutes an NP-complete problem [67], whose complexity depends on the features considered [51], such as: (a) the number scheduling of criteria to be optimized (one vs. multi-criteria); (b) nature of the environment (static vs. dynamic); (c) nature of tasks (*Batch* or *Service*); and (d) dependency between tasks (independent vs. dependent).

In this work, we use a heuristic algorithm that takes into account the aforementioned requirements in order to solve the NP-complete problem. This scheduling algorithm is based on a genetic algorithm with dedicated population representation [30,39], which can be characterized as follows: (a) a single gene represents one task, which is unique within the population; (b) each chromosome is composed by a set of tasks (genes); (c) each individual is composed of one chromosome and represents a scheduling assignation for a single computing node; (d) the population is composed of m individuals and represents a schedule for all n tasks; (e) the fitness function depends on the optimization objectives presented in Section 3.5. All individuals take part in the reproduction process. Individuals presenting the lowest value for the fitness function (best adapted) are crossed with worst-adapted individuals (those that show the highest values for the fitness function). Crossing involves exchanging genes between chromosomes. The population obtained in the evolution process defines the suboptimal schedule.

3.4. Energy calculation

Two different power states are considered for each virtual machine in the CC: *busy* (100% core computational power is used for task computing) and *idle* state. Let: t^i_{idle} denote the time the ith machine spends in an idle state; t^i_{busy} — the time the machine spends in computing tasks-related operations; P^i_{idle} — the required power for a machine to run in idle state; and P^i_{busy} — the power required by a machine to perform actual computing operations. The power required to perform security-related activities is assumed to be the same as in busy mode.

The aforementioned parameters may vary in each schedule and can be defined as follows [42]:

$$t^i_{busy} = \max_{j \in Tasks^i} C_j \tag{11}$$

$$t^i_{idle} = C_{\max} - t^i_{busy} \tag{12}$$

$$t^i_{sec} = \sum_{j \in Tasks^i} b^i_j \tag{13}$$

where $Tasks^i$ represents the tasks assigned to VM_i and t^i_{sec} denotes the time devoted to processing only the security-related operations.

The total energy consumption can be denoted as follows:

$$E_{total} = \sum_{i=1}^{m} \int_0^{C_{max}} Pow_{VM_i}(t)dt = \sum_{i=1}^{m} (P^i_{idle} * t^i_{idle} + P^i_{busy} * (t^i_{busy} + t^i_{sec})). \tag{14}$$

The presented energetic model is designed for the assignation of tasks to virtual machines. However, this model could be adapted

to work with other scheduling technologies, such as that based on the assignation of tasks in the form of Linux micro-containers to computing nodes. These micro-containers run only for the task execution time. The energetic model could also be extended in order to consider specific hardware configurations.

3.5. Energy-aware scheduling objectives

The determination of the solution that minimizes the makespan of a given schedule that assumes a constant computing power may be defined as follows:

$$\underset{s\in Schedules}{\operatorname{argmin}} \sum_{\substack{i=1,\dots,m\\ j=1,\dots,n}} \left(\frac{wl_j}{cc_i} + b^i\right)\delta_{i,j}(s) \tag{15}$$

where $\delta_{i,j}(s)$ equals one when the schedule s assigns the jth task to the ith VM. Otherwise, $\delta_{i,j}(s)$ equals zero.

Moreover, the determination of the solution that minimizes the total energy consumption of a given schedule can be written as:

$$\underset{s\in Schedules}{\operatorname{argmin}} \sum_{i=1,\dots,m} \left(\sum_{\substack{j=1\\ \delta_{i,j}(s)=1}}^{n} P^i_{busy}\left(\frac{wl_j}{cc_i} + b^i\right) + \sum_{\substack{j=1,\\ \delta_{i,j}(s)=0}}^{n} P^i_{idle} t^i_{idle}\right) \tag{16}$$

with the following constraints (see Eqs. (2) and (3)):

$$sd_j \leq tl_i. \tag{17}$$

Various energy-saving approaches may be tested by modifying the trust level of any given machine thanks to the SBETC matrix. Moreover, numerous complex and realistic scenarios may be simulated in order to check whether these strategies can be used in real-life Cloud Computing systems.

In this work, we proposed the following four energy-aware and time-aware scheduling policies based on Eqs. (15) and (16):

1. **Makespan-centric scheduling**. Whenever two given schedules achieve the same (or close) makespan, the less energy-consuming schedule is selected. This approach is desirable when the makespan is the main scheduling objective and the importance of the reduction of energy consumption is low.
2. **Energy-centric scheduling**. Whenever two given schedules present approximately the same energy consumption, the schedule with the shorter makespan is selected. This approach is suitable when the energy efficiency is the main objective and the execution time is not critical.
3. **Makespan-centric scheduling until a given makespan threshold**. In this policy, the minimization of the makespan is the main goal. Once a makespan threshold is achieved, then the minimization of the energy consumption becomes the main objective.
4. **Energy-centric scheduling until a given energy-consumption threshold**. In this policy, the minimization of the energy consumption is the main goal. Once a energy-consumption threshold is achieved, then the minimization of the makespan becomes the main objective.

3.6. Energy policies based on the hibernation of virtual machines

The volume of work that must be executed at any given time by a CC system may significantly change, especially with peak loads largely exceeding mean loads. The proper execution of this ever-changing workload while achieving energy-proportionality represents a major challenge. CC operators may choose between the following strategies in order to face the challenge: (a) the over-provision of the data-center to satisfy worst-case scenarios; and (b) the adjustment of the available resources according to the present and future workload demands.

The first of these two approaches represents the main trend implemented in the vast majority of large Cloud Computing systems. However, this strategy requires a high amount of energy to keep servers in an idle state during long periods of time, while they wait to serve worst-case peak loads.

Many software solutions implement the second strategy by switching off either server components or whole servers to reduce the energy consumption in low-utilization periods. However, this approach could damage end-user experience and SLAs if these workload peaks are not properly determined and served.

Various energy-efficiency policies based on the shut-down of machines have been tested in grid computing scenarios, including: (a) the shut-down of every machine whenever possible; and (b) the shut-down of machines according to the workload demands. These policies have shown good energy-savings in [26,27]. In this work, we adapted these energy-efficiency policies, which are designed for grid computing environments, in order to be applied in CC systems.

Our aim is the development of energy-efficiency policies that rely on resource schedulers that could in CC systems of different sizes, and that may serve various and heterogeneous workloads rather than focusing on a specific scenario or infrastructure.

The power-off energy-efficiency policies are responsible for deciding whether any given machine should be turned off or kept in an idle state, and for performing the actual hibernation process while keeping the environment state information up to date.

These power-off policies may be deterministic, such as the *Always power off* policy, or probabilistic, which forecast future workload demands based on historical data and then to perform required actions according to this prediction. Power-off policies may check various system, workload and machine parameters in order to make decision about shutting any given machine down.

The following deterministic policies have been considered in this paper:

1. **Never power off**: This power-off policy prevents any given virtual machine to be hibernated. This is the current operating approach in many real Cloud Computing systems nowadays. Due to this, the power-off policy should be considered and studied so the energy savings achieved by any other power-off policy can be compared to the current power consumption scenario.
2. **Always power off**: Opposite to the *Never power off* policy, this policy always tries to hibernate any virtual machine that becomes idle.

The shut-down process is performed whenever any resource in use (RAM, CPU) is released due to the execution of a task finished. At this moment, the system makes a decision whether the machine those resources belongs to should be turned off or not. The system prevents any virtual machine that is executing tasks from being hibernated.

4. Evaluation of energy-aware scheduling vs. makespan scheduling in cloud computing systems

We propose an environment that simulates a monolithic scheduling framework to serve realistic and heterogeneous workloads in order to test the proposed strategies. The CC environment has been simulated for seven days of operation time and various combinations of the energy policies developed and described in Section 3.6 have been evaluated.

In the following subsections a simulation tool, a test suite and a designed environment are presented in detail.

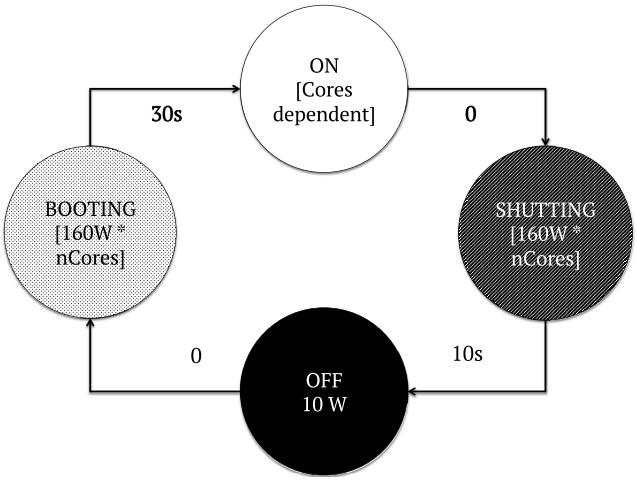


Fig. 5. Machine power states.

4.1. Simulation tool

In this work we used the SCORE simulator presented in [24]. This simulator enables us to focus on the development of energy-efficiency policies and on the performance of simulations of various scheduling frameworks and data-center environments. This simulation tool has been modified in order to perform energy-efficiency analysis by applying an energy-consumption model which considers the following states for each CPU core in a machine: (a) *On*: 150 W (b) *Idle*: 70 W. The energy consumption is linearly computed in terms of the utilization of each CPU core. In addition to these CPU core power consumption states, the following machine power states have been assumed: (a) *Hibernated*: 10W (b) *Hibernating*: 160 W * number of cores (c) *Powering On*: 160 W * number of cores.

Regarding the shut-down process time parameters, the following values have been considered: (a) $T_{On \rightarrow Hibernated}$: 10 s, and (b) $T_{Hibernated \rightarrow On}$: 30 s. The power states and transitions are shown in Fig. 5.

In order to develop and apply our energy-efficiency policies, a new set of modules has been built on top of the current simulator. Among these additions, the following can be found: (a) sorting, (b) scheduling, and (c) power-off policies.

However, in order to preserve trust in the schedulers' implementations, the behavior of the overall simulation process has not been modified. Instead of modifying the current implementation, hooks were placed in key parts of the simulation process to execute our developed policies and to register new key performance indicators, which have been added in order to measure the impact on data center energy consumption.

As a result of this approach, the developed energy-efficiency policies have achieved a high level of isolation from the base simulator implementation, thereby affecting the original simulator design to a minimum extent.

4.2. Cloud computing center

A CC data center composed of 1000 heterogeneous virtual machines has been modeled. Each machine has the following features:

- **Computing profile**: Processor's millions of instructions per second (MIPS) have been simulated by generating randomly a [1× - 4×] computing speed factor. Thus, a given VM may be, as a maximum, four times faster than the slowest one: $cc_i \in [75000, 300000]$ MIPS.
- **Energy profile**: Processor's power consumption heterogeneity has been simulated by generating randomly a [1× - 4×] energy consumption factor. Thus, a given machine M may be (as a maximum) four times more energy-wasting than the more efficient one. Hence, for a 4-core server, the maximum power consumption may be described as: $P_{total} \in [300, 1200]$ W.
- **Security profile**: Cryptographic services have been chosen according to the FIPS standard [32], and ISO/IEC 19790 standard [33] for security requirements for cryptographic modules, as described in [42]. These standards specify four operating levels of general security requirements for cryptography modules, which have been simulated by randomly generating a security factor in the range [1–4]. Therefore, $TL \in [0.25, 1]$.
- **Computational resources**: Every machine has 4 CPU cores and 16 GB of RAM.

4.3. Workload

The patterns present in the realistic Google traces [61,62] were followed to generate the synthetic workload used in the experimentation. The interpretations by [6,19,55,60] have been studied to model the synthetic workloads.

These workload tasks are composed of one or more (sometimes more than thousand) tasks. Every task is modeled to use a given number of millions of instructions (MI).

Moreover, the two types of tasks described in Section 3.2 are considered.

Each experiment executes the workload generated by replicating the behavior of the workload present in typical Google data centers. Therefore, although the workload generated in each simulation run is unique, it follows the same model design. In this workload, the vast majority of tasks are batch tasks, however, over half of the available resources are reserved to service tasks.

Moreover, batch tasks are usually composed of a greater number of tasks than service tasks. However, these tasks require fewer resources and run for a shorter time than service tasks. Hence, the simulator generates a day/night patterned synthetic dataset composed of tasks whose attributes follow an exponential distribution.

Taking into account the aforementioned environment and workload scenario, the generated workload presents 22,208 batch tasks and 2252 service tasks for each experiment that simulates 7 days of operation time, reaching 30.08% of average computational power and 25.72% of memory in use. This data center utilization rates follow industry trends presented in [9,63].

4.4. Key performance indicators

In order to measure the results of the application of energy-efficiency policies that switch machines on/off, the authors need to measure key performance indicators of the data-center operation. These indicators have been divided into two categories: (a) energy savings; and (b) performance.

The following indicators were selected in order to describe the energy savings and the behavior of the powering on/off operations:

- **Energy consumption**: The total energy consumed in each experiment, E_{total} (14).
- **Energy savings**: The total energy saved in each experiment.

The following indicators were selected as the most relevant performance indicators:

- **Queue time**: Represents the time a task waits in the queue until it is scheduled. This indicator is usually related to the real computing experience, and therefore it is critical to maintain this time as short as possible.

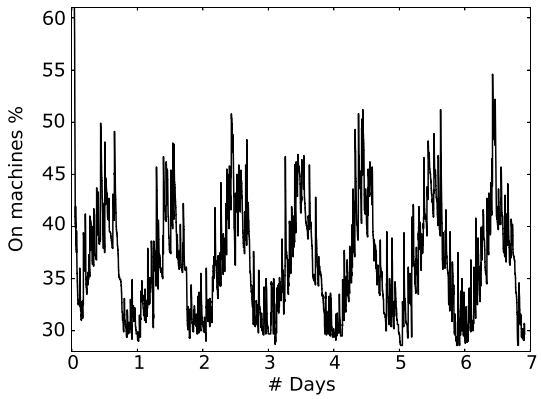


Fig. 6. Percentage of powered-on machines when the *Always power off* policy is used for the single-path scheduler.

- **Makespan**: C_{max} (see (10)).

In order to analyze and compare the energy savings and the performance impact of deploying hibernating energy-efficiency policies, the simplest and most aggressive energy policies have been applied, i.e., the *Never power off* and the *Always power off*. They will be applied to the most representative scheduling policies proposed in Section 3.5. Among them:

- The Makespan-centric scheduling (policy 1) is applied to batch tasks. The scheduling policy tries to load every machine up to 90%. The rest of the computational power is used for service tasks (cf. [49]). The evolution of the fitness function value in average of the genetic process applied to batch tasks can be observed in Fig. 7b;
- The Energy-centric scheduling (policy 2) is applied to batch tasks. The same scheduling policy described in the Makespan-centric scheduling is used for service tasks. The evolution of the fitness function value in average of the genetic process applied to batch tasks may be observed in Fig. 7a;
- The Random strategy for both batch and service tasks. This strategy selects a random machine from the subset of machines that meet tasks requirements. This scheduling policy is especially important because many of top-industry companies implement a similar strategy, such as round robin-like methods.

The scheduling algorithm workflows may be described as follows: The random scheduler assigns tasks to VMs randomly, according to the Round Robin-like schema (i.e., the *Random* strategy). The GMakespan (Genetic-based with makespan as the main objective) scheduler assigns tasks according to the solution of the optimization problem shown in (15), by the means of the genetic algorithm described in the policy 1 in Section 3.5. The GEnergy (Genetic-based with energy as a main objective) scheduler assigns tasks according to the solution of optimization problem presented in (16), by the means of the genetic algorithm described in the policy 2 in Section 3.5. The *Never power off* policy lets VMs be in an idle state when the execution of tasks is finished, while the *Always power off* policy hibernates them.

5. Results and discussion

In this section, the simulation results obtained for the *Makespan-centric*, *Energy-centric* and *Random* scheduling policies are discussed through key performance indicators concerning: (a) energy savings; and (b) performance impact. We choose to only show the results obtained for the most representative scheduling policies since the *Makespan-centric scheduling until a given makespan threshold* and the *Energy-centric scheduling until a given energy-consumption threshold* are mixed strategies that could blur the main differences between the opposite *Energy-centric* and *Makespan-centric* scheduling policies. The energy savings and performance are analyzed and compared between the current/base system energy policy (*Never power off* policy) and the *Always power off* energy-efficiency policy. Table 1 shows numeric results for the single-path scheduler and Table 2 presents those for the multi-path scheduler. In general, the more hibernations there are, the more energy is saved, or from another point of view, the less idle the resources, the less energy is wasted.

Moreover, it can be observed that the utilization of only the genetic algorithm that focuses on the minimization of the energy consumption results in a higher energy consumption than the genetic algorithm that focuses on the minimization of the makespan (56.17 MWh vs. 55.96 MWh, as shown in Table 1, scenario Never off policy, *GEnergy* Scheduler vs *GMakespan* Scheduler). On the other hand, it can be noticed that high energy savings up to approximately 45% may be achieved by applying the *Always off* policy for the genetic algorithm that focuses on the minimization of the makespan (30.45 MWh consumed with the *Always off* policy vs. 55.96 MWh consumed with the *Never off* policy for the *GMakespan* Scheduler, as presented in Table 1). This behavior, that is similar for the Monolithic multi-path scheduler presented in Table 2, means that only a 20% of energy is wasted from the theoretical optimum instead of 70% of the current approach.

Regarding the performance, the application of the *Always power off* energy-efficiency policy has a negative impact of approximately 35% in terms of scheduling queue time. This behavior can be observed in Table 1, where *Batch* tasks wait on average approximately 20 more seconds (+40%) in queue for the *GMakespan* scheduler and 15 more seconds (+30%) for the *GEnergy* scheduler in queue. Similarly, *Batch* tasks have a longer makespan (+60s. and +120s. on average for the *GMakespan* and *GEnergy* schedulers respectively, as presented in Table 2) when the combination of the *Always power off* energy-efficiency policy and the genetic algorithm is used. This makespan impact is especially negative when the genetic algorithm that focuses on the minimization of energy is used (+140s. for the *Always off* policy and *GEnergy* vs. *Random* schedulers respectively, as shown in Table 2). On the other hand, it is noticeable that *Service* tasks never suffer from this negative makespan impact as shown in Tables 1 and 2.

It can be noticed that the scheduling policy is crucial not only for the performance, but also for the whole hibernation process. As shown in Table 1, the *Random* scheduling policy almost prevents any hibernation. In this case the *Always power off* policy results in a similar energy consumption (54.66 MWh) compared to that achieved by the *Never power off* policy (56.19 MWh). Table 2 shows that the Multi-path monolithic scheduler presents the same behavior as the Single-path monolithic scheduler, except for the queue times, which are notably lower (−85% between queue time results shown in Table 1 and those presented in Table 2) due to the Multi-path approach preventing the head-of-line blocking issue. Table 3 summarizes the impact of the *Never power off* policy in terms of both queue times and energy consumption compared to leaving machines in an idle state. Fig. 6 presents the percentage of VMs in a hibernated mode for a seven-day time span. It can be observed that the *Always power off* policy fits perfectly the clear day/night pattern workload. Taking into consideration the aforementioned results, we can state that the *Makespan-centric scheduling* policy provides the best results for the goals under consideration, thus: the minimization of the energy consumption through the application of hibernation policies with a minor negative impact on the CC system performance.

Table 1
Results for Monolithic Single-path scheduler.

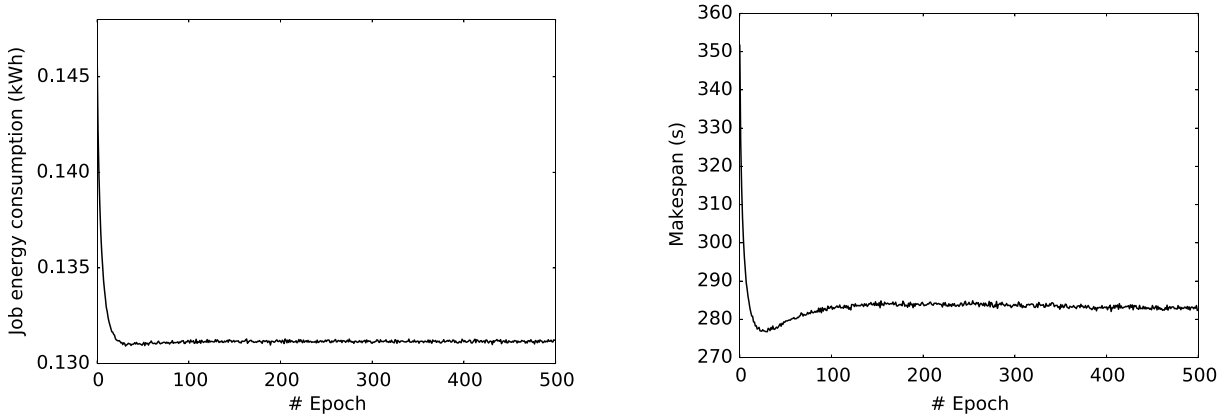
Policy	Scheduler	Energy (MWh)	E savings (MWh)	Queue time (ms)		Makespan (s)	
				Batch	Service	Batch	Service
Never off	Random	56.19	0.00	49.70	57.70	177.44	1,988.21
Always off	Random	54.66	1.57	49.70	57.70	177.21	1,988.21
Never off	GMakespan	55.96	0.00	49.70	57.70	235.71	1,988.21
Always off	GMakespan	30.43	25.92	71.40	69.90	258.95	1,988.30
Never off	GEnergy	56.17	0.00	49.70	57.70	287.48	1,988.21
Always off	GEnergy	30.68	25.83	66.90	69.10	310.19	1,988.38

Table 2
Results for Monolithic Multi-path scheduler.

Policy	Scheduler	Energy (MWh)	E savings (MWh)	Queue time (ms)		Makespan (s)	
				Batch	Service	Batch	Service
Never off	Random	56.21	0.00	06.90	06.50	178.56	1,920.60
Always off	Random	55.00	1.13	06.90	06.50	178.56	1,920.60
Never off	GMakespan	55.90	0.00	06.90	06.50	236.01	1,920.21
Always off	GMakespan	30.08	26.12	10.30	07.20	258.46	1,920.66
Never off	GEnergy	56.09	0.00	06.90	06.50	290.55	1,920.60
Always off	GEnergy	30.65	25.76	11.00	06.90	313.41	1,920.67

Table 3
Always off policy results vs. current situation, represented by the *Never power off* policy.

Scheduler	Strategy	Savings	Queue time diff		Makespan diff	
			Batch	Service	Batch	Service
Random	Single-path	02.79%	0	0	−00.13%	N/A
GMakespan	Single-path	45.99%	+43.67%	+21.14%	+45.94%	N/A
GEnergy	Single-path	45.71%	+34.60%	+19.76%	+74.81%	N/A
Random	Multi-path	02.01%	0	0	0	N/A
GMakespan	Multi-path	46.48%	+49.27%	+10.77%	+44.75%	N/A
GEnergy	Multi-path	45.67%	+59.42%	+06.15%	+75.52%	N/A



(a) Task energy consumption.

(b) Task makespan.

Fig. 7. Genetic process fitness evolution.

An important observation about the genetic process used for finding the solution of the minimization problem (Eqs. (15) and (16)) is that an early stopping strategy should be incorporated. From Fig. 7b and 7a it can be seen that the genetic process should be stopped after approximately 50 epochs in order to achieve the best results.

6. Summary

In this paper a model for reducing the energy consumption in CC environments has been described. The presented approach enables us to reduce the energy consumption of the CC system up to 45%. The proposed model is composed of two parts: (a) an energy-aware independent batch scheduler; and (b) a set of energy-efficiency policies for the hibernation of idle VMs. We proposed four scheduling policies for the control of the energy consumption and the makespan during the assignation of tasks to VMs.

The contributions of this work include:

1. The scheduler task assignation to VMs based on a makespan optimization process. As a result, each batch of tasks is computed in the shortest time, taking into account the current state and the characteristics of the CC system.
2. The hibernation of virtual machines that remain in an idle state, while the rest of VMs continue to execute the batch of tasks. This guarantees the maximum positive impact on the system performance since it does not negatively impact the virtual machines under use.

The proposed scheduler takes the security demands of each task and trust levels of VMs that are computing those tasks into account. Additionally, the proposed model enables us to compute the energy consumption of the whole system, including the energy spent on performing security operations.

The developed methods were tested using the realistic workload of Google traces for seven consecutive days on the simulated environment equipped with 1000 virtual machines. The experimental results show that the application of the proposed model, especially that parameterized with a scheduling policy focused on the minimization of the makespan, in addition to an energy-efficiency policy based on the hibernation of every virtual machine whenever possible, could successfully reduce the energy consumption of large-scale data centers which securely serve heterogeneous workloads without notably impacting on the cloud computing system overall performance.

The next stage of our research is the optimization of security operations. We intend to apply game theory solutions which have been developed previously (see: [43]) for the optimization of the Trust Levels of VMs and for the decision of the applied security biases.

Acknowledgment

This article is based upon work from COST Action IC1406 "High-Performance Modelling and Simulation for Big Data Applications" (cHiPSet), supported by COST (European Cooperation in Science and Technology).

The research is supported by the VPPI — University of Sevilla.

References

[1] Amazon Cloud Scaling Service, URL http://docs.aws.amazon.com/autoscaling/latest/userguide/scaling_plan.html.
[2] Amazon CloudWatch, URL https://aws.amazon.com/cloudwatch/.
[3] Google Cloud Scaling Service, URL https://cloud.google.com/compute/docs/autoscaler/scaling-cpu-load-balancing.
[4] OpenStack Cloud Scaling Service, URL https://wiki.openstack.org/wiki/Heat/AutoScaling.
[5] Rackspace Cloud Scaling Service, URL https://support.rackspace.com/how-to/rackspace-auto-scale-overview/.
[6] O.A. Abdul-Rahman, K. Aida, Towards understanding the usage behavior of Google cloud users: the mice and elephants phenomenon, in: IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Singapore, 2014, pp. 272–277.
[7] H. Amur, J. Cipar, V. Gupta, G.R. Ganger, M.A. Kozuch, K. Schwan, Robust and flexible power-proportional storage, in: Proceedings of the 1st ACM symposium on Cloud computing, ACM, 2010, pp. 217–228.
[8] D.G. Andersen, S. Swanson, Rethinking flash in the data center, IEEE Micro 30 (4) (2010) 52–54.
[9] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58.
[10] A. Beloglazov, R. Buyya, Energy efficient resource management in virtualized cloud data centers, in: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE Computer Society, 2010, pp. 826–831.
[11] A. Beloglazov, R. Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers, Concurr. Comput.: Pract. Exper. 24 (13) (2012) 1397–1420.
[12] R. Bertran, Y. Becerra, D. Carrera, V. Beltran, M. Gonzalez, X. Martorell, J. Torres, E. Ayguade, Accurate energy accounting for shared virtualized environments using PMC-based power modeling techniques, in: 2010 11th IEEE/ACM International Conference on Grid Computing, 2010, pp. 1–8. http://dx.doi.org/10.1109/GRID.2010.5697889.
[13] A.E.H. Bohra, V. Chaudhary, VMeter: Power modelling for virtualized clouds, in: 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010, pp. 1–8. http://dx.doi.org/10.1109/IPDPSW.2010.5470907.
[14] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, J. Wilkes, Borg, omega, and kubernetes, Commun. ACM 59 (5) (2016) 50–57.
[15] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, R.E. Gruber, Bigtable: A distributed storage system for structured data, ACM Trans. Comput. Syst. (TOCS) 26 (2) (2008) 4.
[16] M. Colmant, M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, A. Sobe, Process-level power estimation in VM-based systems, in: Proceedings of the Tenth European Conference on Computer Systems, EuroSys '15, ACM, New York, NY, USA, 2015, pp. 14:1–14:14. http://dx.doi.org/10.1145/2741948.2741971.
[17] H. David, C. Fallin, E. Gorbatov, U.R. Hanebutte, O. Mutlu, Memory power management via dynamic voltage/frequency scaling, in: Proceedings of the 8th ACM International Conference on Autonomic Computing, ACM, 2011, pp. 31–40.
[18] G. Dhiman, K. Mihic, T. Rosing, A system for online power prediction in virtualized environments using Gaussian mixture models, in: Design Automation Conference, 2010, pp. 807–812. http://dx.doi.org/10.1145/1837274.1837478.
[19] S. Di, D. Kondo, C. Franck, Characterizing cloud applications on a Google data center, in: 42nd International Conference on Parallel Processing (ICPP), Lyon, France, 2013.
[20] C. Dupont, T. Schulze, G. Giuliani, A. Somov, F. Hermenier, An energy aware framework for virtual machine placement in cloud federated data centres, in: 2012 Third International Conference on Future Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012, pp. 1–10. http://dx.doi.org/10.1145/2208828.2208832.
[21] N. El-Sayed, I.A. Stefanovici, G. Amvrosiadis, A.A. Hwang, B. Schroeder, Temperature management in data centers: why some (might) like it hot, ACM SIGMETRICS Perform. Eval. Rev. 40 (1) (2012) 163–174.
[22] B. Fateh, M. Govindarasu, Joint scheduling of tasks and messages for energy minimization in interference-aware real-time sensor networks, IEEE Trans. Mob. Comput. 14 (1) (2015) 86–98. http://dx.doi.org/10.1109/TMC.2013.81.
[23] M.E. Femal, V.W. Freeh, Boosting data center performance through non-uniform power allocation, in: Second International Conference on Autonomic Computing, ICAC'05, IEEE, 2005, pp. 250–261.
[24] D. Fernández-Cerero, A. Fernández-Montes, A. Jakóbik, J. Kołodziej, M. Toro, SCORE: Simulator for cloud optimization of resources and energy consumption, Simul. Model. Pract. Theory 82 (2018) 160–173. http://dx.doi.org/10.1016/j.simpat.2018.01.004.
[25] A. Fernández-Montes, D. Fernández-Cerero, L. González-Abril, J.A. Álvarez-García, J.A. Ortega, Energy wasting at internet data centers due to fear, Pattern Recognit. Lett. 67 (2015) 59–65.
[26] A. Fernández-Montes, L. Gonzalez-Abril, J.A. Ortega, L. Lefèvre, Smart scheduling for saving energy in grid computing, Expert Syst. Appl. 39 (10) (2012) 9443–9450.
[27] A. Fernández-Montes, F. Velasco, J. Ortega, Evaluating decision-making performance in a grid-computing environment using DEA, Expert Syst. Appl. 39 (15) (2012) 12061–12070.
[28] X. Gao, Z. Xu, H. Wang, L. Li, X. Wang, Why some like it hot too: Thermal attack on data centers, in: Proceedings of the 2017 ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems, ACM, 2017, pp. 23–24.
[29] D. Grzonka, The analysis of OpenStack cloud computing platform: Features and performance, J. Telecommun. Inf. Technol. 3 (2015) 52–57.
[30] D. Grzonka, A. Jakóbik, J. Kołodziej, S. Pllana, Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security, Future Gener. Comput. Syst. (2017). http://dx.doi.org/10.1016/j.future.2017.05.046.
[31] D. Grzonka, J. Kołodziej, J. Tao, Using Artificial Neural Network For Monitoring And Supporting The Grid Scheduler Performance, in: 28th European Conference on Modelling and Simulation, ECMS 2014, Brescia, Italy, May 27–30, 2014, Proceedings, 2014, pp. 515–522. http://dx.doi.org/10.7148/2014-0515.
[32] D. Grzonka, J. Kołodziej, J. Tao, S.U. Khan, Artificial neural network support to monitoring of the evolutionary driven security aware scheduling in computational distributed environments, Future Gener. Comput. Syst. 51 (2015) 72–86. http://dx.doi.org/10.1016/j.future.2014.10.031.
[33] D. Grzonka, M. Szczygiel, A. Bernasiewicz, A. Wilczyński, M. Liszka, Short analysis of implementation and resource utilization for the openstack cloud computing platform, in: 29th European Conference on Modelling and Simulation, ECMS 2015, Albena (Varna), Bulgaria, May 26–29, 2015. Proceedings, 2015, pp. 608–614. http://dx.doi.org/10.7148/2015-0608.
[34] C. Gu, H. Huang, X. Jia, Power metering for virtual machine in cloud computing-challenges and opportunities, IEEE Access 2 (2014) 1106–1116. http://dx.doi.org/10.1109/ACCESS.2014.2358992.
[35] T.W. Harton, C. Walker, M. O'Sullivan, Towards power consumption modeling for servers at scale, in: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), 2015, pp. 315–321. http://dx.doi.org/10.1109/UCC.2015.50.
[36] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A.D. Joseph, R.H. Katz, S. Shenker, I. Stoica, Mesos: A platform for fine-grained resource sharing in the data center, in: NSDI, vol. 11, 2011, pp. 22–22.

[37] A. Jakóbik, Big Data Security, in: F. Pop, J. Kołodziej, B. Di Martino (Eds.), Resource Management for Big Data Platforms: Algorithms, Modelling, and High-Performance Computing Techniques, Springer International Publishing, Cham, 2016, pp. 241–261. http://dx.doi.org/10.1007/978-3-319-44881-7_12.

[38] A. Jakóbik, A cloud-aided group RSA scheme in Java 8 environment and OpenStack software, J. Telecommun. Inf. Technol. JTIT (2) (2016) 53–59.

[39] A. Jakóbik, D. Grzonka, J. Kołodziej, Security supportive energy aware scheduling and scaling for cloud environments, in: European Conference on Modelling and Simulation, ECMS 2017, Budapest, Hungary, May 23–26, 2017, Proceedings, 2017, pp. 583–590. http://dx.doi.org/10.7148/2017-0583.

[40] A. Jakóbik, D. Grzonka, J. Kołodziej, A.E. Chis, H. Gonzalez-Velez, Energy efficient scheduling methods for computational grids and clouds, J. Telecommun. Inf. Technol. 1 (2017) 56–64.

[41] A. Jakóbik, D. Grzonka, J. Kołodziej, H. González-Vélez, Towards secure non-deterministic meta-scheduling for clouds, in: 30th European Conference on Modelling and Simulation, ECMS 2016, Regensburg, Germany, May 31–June 3, 2016, Proceedings, 2016, pp. 596–602. http://dx.doi.org/10.7148/2016-0596.

[42] A. Jakóbik, D. Grzonka, F. Palmieri, Non-deterministic security driven meta scheduler for distributed cloud organizations, in: High-Performance Modelling and Simulation for Big Data Applications, Simul. Model. Pract. Theory 76 (2017) 67–81. http://dx.doi.org/10.1016/j.simpat.2016.10.011.

[43] A. Jakóbik, A. Wilczyński, Using polymatrix extensive stackelberg games in security — Aware resource allocation and task scheduling in computational clouds, J. Telecommun. Inf. Technol. 1/2017 (1) (2017) 71–80.

[44] A. Kansal, F. Zhao, J. Liu, N. Kothari, A.A. Bhattacharya, Virtual machine power metering and provisioning, in: Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC '10, ACM, New York, NY, USA, 2010, pp. 39–50. http://dx.doi.org/10.1145/1807128.1807136.

[45] K. Karanasos, S. Rao, C. Curino, C. Douglas, K. Chaliparambil, G.M. Fumarola, S. Heddaya, R. Ramakrishnan, S. Sakalanaga, Mercury: hybrid centralized and distributed scheduling in large shared clusters, in: USENIX Annual Technical Conference, 2015, pp. 485–497.

[46] H. Kataoka, D. Duolikun, T. Enokido, M. Takizawa, Power consumption and computation models of a server with a multi-core cpu and experiments, in: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, 2015, pp. 217–222. http://dx.doi.org/10.1109/WAINA.2015.127.

[47] H. Kataoka, A. Sawada, D. Duolikun, T. Enokido, M. Takizawa, Energy-aware server selection algorithms in a scalable cluster, in: 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), 2016, pp. 565–572. http://dx.doi.org/10.1109/AINA.2016.154.

[48] R.T. Kaushik, M. Bhandarkar, Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster, in: Proceedings of the USENIX Annual Technical Conference, 2010, p. 109.

[49] G. Khaneja, An experimental study of monolithic scheduler architecture in cloud computing systems (Ph.D. thesis), University of Illinois at Urbana-Champaign, 2015.

[50] J.-Y. Kim, H.-J. Chang, Y.-H. Jung, K.-M. Cho, G. Augenbroe, Energy conservation effects of a multi-stage outdoor air enabled cooling system in a data center, Energy Build. 138 (2017) 257–270.

[51] J. Kołodziej, Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems, Springer Publishing Company, Incorporated, 2012.

[52] J. Koomey, Growth in data center electricity use 2005 to 2010, A report by Analytical Press, completed at the request of The New York Times 9, 2011.

[53] B. Krishnan, H. Amur, A. Gavrilovska, K. Schwan, VM power metering: Feasibility and challenges, SIGMETRICS Perform. Eval. Rev. 38 (3) (2010) 56–60. http://dx.doi.org/10.1145/1925019.1925031.

[54] Y. Li, Y. Wang, B. Yin, L. Guan, An online power metering model for cloud environment, in: 2012 IEEE 11th International Symposium on Network Computing and Applications, 2012, pp. 175–180. http://dx.doi.org/10.1109/NCA.2012.10.

[55] Z. Liu, S. Cho, Characterizing machines and workloads on a Google cluster, in: 8th International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems (SRMPDS), 2012, Pittsburgh, PA, USA.

[56] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, R. Rajkumar, Critical power slope: understanding the runtime effects of frequency scaling, in: Proceedings of the 16th International Conference on Supercomputing, ACM, 2002, pp. 35–44.

[57] I.M. Murwantara, B. Bordbar, A Simplified Method of Measurement of Energy Consumption in Cloud and Virtualized Environment, in: Proceedings of the 2014 IEEE Fourth International Conference on Big Data and Cloud Computing, BDCLOUD '14, IEEE Computer Society, Washington, DC, USA, 2014, pp. 654–661. http://dx.doi.org/10.1109/BDCloud.2014.47.

[58] K. Ousterhout, P. Wendell, M. Zaharia, I. Stoica, Sparrow: distributed, low latency scheduling, in: Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, ACM, 2013, pp. 69–84.

[59] J. Read, What is an ECU? CPU Benchmarking in the Cloud. URL http://blog.cloudharmony.com/2010/05/what-is-ecu-cpu-benchmarking-in-cloud.html.

[60] C. Reiss, A. Tumanov, G.R. Ganger, R.H. Katz, M.A. Kozuch, Heterogeneity and dynamicity of clouds at scale: Google trace analysis, ACM Symposium on Cloud Computing (SoCC), San Jose, CA, USA, 2012.

[61] C. Reiss, J. Wilkes, J.L. Hellerstein, Google cluster-usage traces: format + schema, Technical report, Google Inc., Mountain View, CA, USA, 2011.

[62] C. Reiss, J. Wilkes, J.L. Hellerstein, Obfuscatory obscanturism: making workload traces of commercially-sensitive systems safe to release, in: 3rd International Workshop on Cloud Management, CLOUDMAN, IEEE, Maui, HI, USA, 2012, pp. 1279–1286.

[63] S. Ruth, Reducing ICT-related carbon emissions: an exemplar for global energy policy? IETE Tech. Rev. 28 (3) (2011) 207–211.

[64] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, J. Wilkes, Omega: flexible, scalable schedulers for large compute clusters, in: Proceedings of the 8th ACM European Conference on Computer Systems, ACM, 2013, pp. 351–364.

[65] R.K. Sharma, C.E. Bash, C.D. Patel, R.J. Friedrich, J.S. Chase, Balance of power: Dynamic thermal management for internet data centers, IEEE Internet Comput. 9 (1) (2005) 42–49.

[66] W. Tu, Efficient resource utilization for multi-flow wireless multicasting transmissions, IEEE J. Sel. Areas Commun. 30 (7) (2012) 1246–1258. http://dx.doi.org/10.1109/JSAC.2012.120810.

[67] J. Ullman, NP-complete scheduling problems, J. Comput. System Sci. 10 (3) (1975) 384–393. http://dx.doi.org/10.1016/S0022-0000(75)80008-0.

[68] I. Wassmann, D. Versick, D. Tavangarian, Energy consumption estimation of virtual machines, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, ACM, New York, NY, USA, 2013, pp. 1151–1156. http://dx.doi.org/10.1145/2480362.2480579.

[69] Y. Zhao, J. Wu, F. Li, S. Lu, On Maximizing the lifetime of wireless sensor networks using virtual backbone scheduling, IEEE Trans. Parallel Distrib. Syst. 23 (8) (2012) 1528–1535. http://dx.doi.org/10.1109/TPDS.2011.305.

[70] S. Zimmermann, I. Meijer, M.K. Tiwari, S. Paredes, B. Michel, D. Poulikakos, Aquasar: A hot water cooled data center with direct energy reuse, Energy 43 (1) (2012) 237–245.