Simulation Modelling Practice and Theory 93 (2019) 3–20

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

GAME-SCORE: Game-based energy-aware cloud scheduler and simulator for computational clouds

Damian Fernández-Cerero[*,a], Agnieszka Jakóbik[b], Alejandro Fernández-Montes[a], Joanna Kołodziej[b]

[a] Department of Computer Languages and Systems, University of Seville, Av. Reina Mercedes S/N, Seville, Spain
[b] Department of Computer Science, Cracow University of Technology, Poland

ARTICLE INFO

Keywords:
Computational cluster
Energy saving
Cloud computing
Energy-aware scheduling
Stackelberg game

ABSTRACT

Energy-awareness remains one of the main concerns for today's cloud computing (CC) operators. The optimisation of energy consumption in both cloud computational clusters and computing servers is usually related to scheduling problems. The definition of an optimal scheduling policy which does not negatively impact to system performance and task completion time is still challenging. In this work, we present a new simulation tool for cloud computing, GAME-SCORE, which implements a scheduling model based on the Stackelberg game. This game presents two main players: a) the scheduler and b) the energy-efficiency agent. We used the GAME-SCORE simulator to analyse the efficiency of the proposed game-based scheduling model. The obtained results show that the Stackelberg cloud scheduler performs better than static energy-optimisation strategies and can achieve a fair balance between low energy consumption and short makespan in a very short time.

1. Introduction

New paradigms, such as cloud computing, and the ever-growing web applications and services, have imposed new challenges to traditional high-performance computing (HPC) systems. In the same time, computational clouds that provide the core foundation for the parallel computing solutions have grown drastically in recent years to satisfy the ever-evolving user requirements. Modern large-scale cloud computing systems are composed of thousands of computational distributed servers. The energy consumed by such cloud computing systems may be compared to the energy utilized by small towns and large factories. computational clusters account for more than 1.5% of global energy consumption [36].

Several hardware and infrastructure models have been recently developed for the successful reduction of the energy consumption in real-life large-scale computational clusters. The most popular models and technologies include:

(a) cooling and temperature management [17,45];
(b) memory and CPU power proportionality [18,39];
(c) construction of energy-efficient flash hard disks [2]; and
(d) new models in energy transportation [19].

* Corresponding author. Tel.: +655793390.
E-mail addresses: damiancerero@us.es (D. Fernández-Cerero), akrok@pk.edu.pl (A. Jakóbik), afdez@us.es (A. Fernández-Montes), jokolodziej@pk.edu.pl (J. Kołodziej).

https://doi.org/10.1016/j.simpat.2018.09.001
Received 24 June 2018; Received in revised form 28 August 2018; Accepted 1 September 2018
Available online 03 September 2018
1569-190X/

Also the resource management and scheduling models in clouds are defined with the energy optimization modules. Energy utilization policies may be based on various power related physical models, however the most popular scenario is to switch off idle servers. Although such power-off strategy is commonly used in small-area grids and clusters [42], in realistic CC systems, the existing power-off models need to be improved especially in the case of dynamical changes in the task workloads and cloud resource infrastructure [24]. It is also important to point out that decisions taken in the organizations might affect both positively and negatively to different parts of the systems [43]

In this work, the balance between two opposed needs of the data-center environment is modelled by means of a Stackelberg Game (SG) as an extension of a previous work presented in [22], where we presented the theoretical model for a game-based energy-aware scheduling algorithm and an algorithm for the dynamic selection of energy-efficiency policies. On one hand, the performance side, represented by the *Scheduling Manager*, which wants tasks to be processed as fast as possible, while the efficiency side (CC provider), represented by the *Energy-Efficiency Manager*, wants the minimization of the energy consumption of the computational cluster.

In our SG model, the *Scheduling Manager*, that is the leader of the game, processes firstly every task to make its decision (move). Once the particular *Task* is processed by the leader, then the follower, that is the *Energy-efficiency Manager*, handles it to make its move. This competition process is implemented in a trustworthy simulation tool focused on simulating realistic large-scale computational-cluster scenarios. Our contributions in this paper include:

(a) An energy-efficient model based on Stackelberg Game which balances the trade-offs of any energy-aware cluster: energy efficiency and performance, through the dynamic application of shut-down policies; and
(b) A simulation tool called GAME-SCORE which implements this model. The presented tool is able to simulate large-cluster environments that supports popular cloud computing services.

The paper is organized as follows. In Section 2 we present a simple analysis on the most relevant energy-aware cluster simulators, as well as a brief description of the main models for energy efficiency in CC systems. In Section 3, we present the developed simulation tool, GAME-SCORE, as well as the formal definition of the implemented Stackelberg Game model for the balance between energy consumption and performance in CC systems. This model theoretical core of this work. Due to this, we also present simple theoretical example of its utilization in this section. The experimental environment, analysis and results obtained are presented in Section 4. Finally, the conclusions and future work are discussed in Section 5

2. Related work

Many efforts have been made in order to increase resource and energy efficiency in computational clouds. Most of the energy is consumed in computational clusters, where the data necessary for the computation is stored. However, the optimization of the scheduling procedures may significantly reduce the time of keeping the requested data ready for use. The data records may be archived when the computation is complete or simply removed from the computational cluster based on the specific end users requests. In this section, we first survey in Section 2.1 the most popular cloud simulators for large-scale clusters, with a special focus on the energy awareness. We present then a simple comparison of the evaluated simulators and critical analysis for better motivation of our work. In in Section 2.2, we define a simple taxonomy of the energy-aware cloud schedulers and survey the classes of models considered in the experimental evaluation presented in this paper.

2.1. Simulation tools for energy-aware cloud scheduling

Cloud simulators are still the main virtual environments for evaluation of the new models of cloud services and schedulers. In this section, we evaluate the following most relevant cloud-computing simulators for the implementation and evaluation of energy-efficiency techniques.

GreenCloud [34] is the extension of the NS2 network simulator. Its purpose is to measure and compute the energy consumption at every computational cluster level, and it pays special attention to network components. However, its packet-level nature compromises performance in order to raise the level of detail, which may be not optimal for the simulation of large computational clusters. In addition, it is not designed to offer ease of development and extension in various scheduling models.

CloudSim is based on SimJava and GridSim and is mainly focused on IaaS-related operation environment features [8]. It presents a high level of detail, and therefore allows several VM allocation and migration policies to be defined, networking to be considered, features and energy consumption to be taken into account. However, it features certain disadvantages when applied for the simulation of large data-center environments: CloudSim is considered cumbersome to execute for large scale scenarios, as well as being closely bound to only monolithic scheduling models.

CloudReports is a highly extensible simulation tool for energy-aware Cloud Computing environments. The tool uses the CloudSim toolkit as its simulation engine and provides features such as a graphic user interface, reports generation, simulation data exportation, power utilization models and an API for easily extend the tool [49]. However, as CloudReports is based on CloudSim, it fails to achieve good performance levels when it comes to large-scale data-center infrastructures, as well as not providing easy and out-of-the-box energy-efficiency strategies based on the shut-down of idle nodes for several scheduling frameworks.

CloudAnalyst adds visual modelling and simulation of large-scale applications that are deployed on Cloud Infrastructures to CloudSim. Several configuration parameters may be set at a high level by using this GUI. However, CloudAnalyst keeps the performance limitations of CloudSim. In addition, this simulator does not provide power-consumption measures out of the box, as it is

focused on the visual modelling of data-center infrastructures, not in energy-awareness [53].

Omnet++ [51] is focused on modelling communication networks (mainly), multiprocessors and other distributed or parallel systems. OMNeT++ is public-source discrete-event simulation tool which has been used as the core simulation engine to test several energy-efficiency techniques in computational clusters . However, this simulator does not provide ready-to-use tools for the measurement and implementation of energy-aware algorithms, and the main focus on modelling networking may make some scheduling-related implementation complicated.

GDCSimGreen [27] focuses on the simulation of the physical behaviour of a computational cluster. This implies the evaluation of energy efficiency of computational clusters under various workload characteristics, platform power management schemes, and scheduling algorithms. However, the low-level hardware and cooling characteristics simulated by this tool make it sub-optimal for large-scale computational clusters due to performance issues. In addition, this simulator does not provide easy and out-of-the-box tools for the simulation of energy-efficiency policies based on the shut-down of the idle machines.

Grid'5000 Toolbox simulates the behaviour of Grid'5000 (France) resources for real workloads while changing the state of the resources according to several energy policies. The simulator includes:

(a) A GUI that allows the user to simulate a set energy policies for each location of Grid'5000;
(b) A graphical visualisation of the state of the resources during the simulation;
(c) A graphical view of the results.

On the other hand, the simulator fails to include various scheduling frameworks and it does not simulate the behaviour or consumption of network devices and resources.

CoolEmAll [11] is focused on evaluation the thermal side of the data-center operation in order to achieve energy-efficiency by combining the optimization of IT, cooling and workload management. On the other hand, the support for testing energy-aware scheduling algorithms as well as workload consolidation and other strategies based on the shut-down of idle nodes is not extensively covered. In addition, this simulator is not designed to achieve high performance when large-scale cluster are considered.

SCORE [20] is designed for the comparison of various scheduling framework in large clusters. To this end, it focuses on maximizing the performance of the simulations by reducing the level of detail . The simulation tool enables the modelling of heterogeneous, realistic and synthetic workloads, as well as it provides the tools to easily develop both new energy-aware scheduling policies for various scheduling frameworks and energy-efficiency policies based on the shut-down of idle nodes.

To summarize, the most widely adopted simulation tools for cloud computing clusters lack some critical features, as shown in Figure 1, that motivated us to develop ENERGY-SCORE based on the SCORE simulator, mainly:

(a) the capacity to dynamically switch between energy-efficiency and scheduling strategies; and
(b) the performance required to run large-scale, heterogeneous and realistic experimentation in a reasonable time.

2.2. Taxonomy of energy-efficiency techniques

In Fig. 2, we present a simple taxonomy of the main techniques to improve energy-efficiency in cluster scheduling.

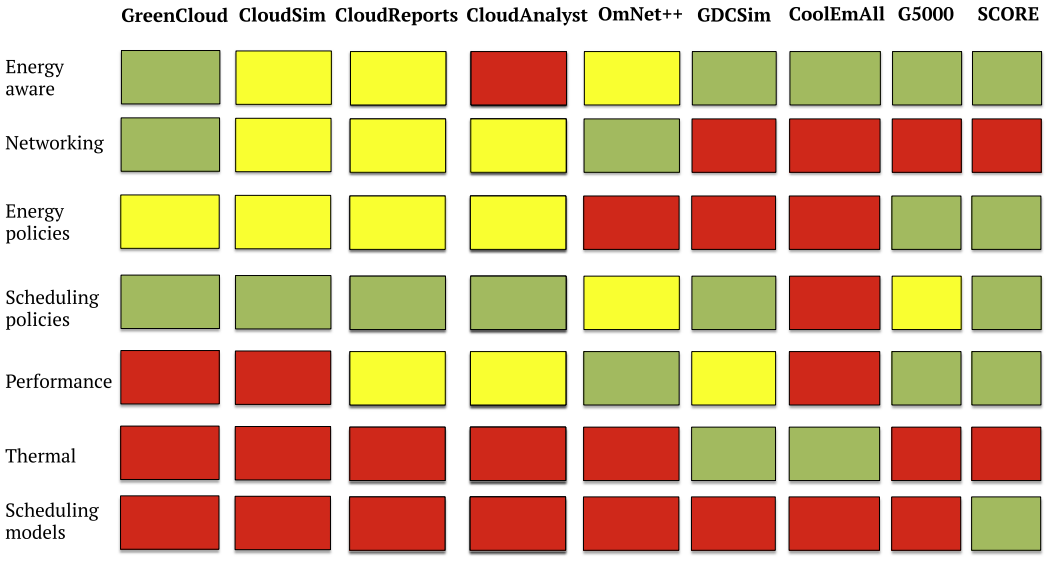

Fig. 1. Brief comparison between available simulation tools. Green parameters mean that either the characteristic is high or the easiness to implement that characteristic is high. The same applies for medium and low (yellow and red respectively).

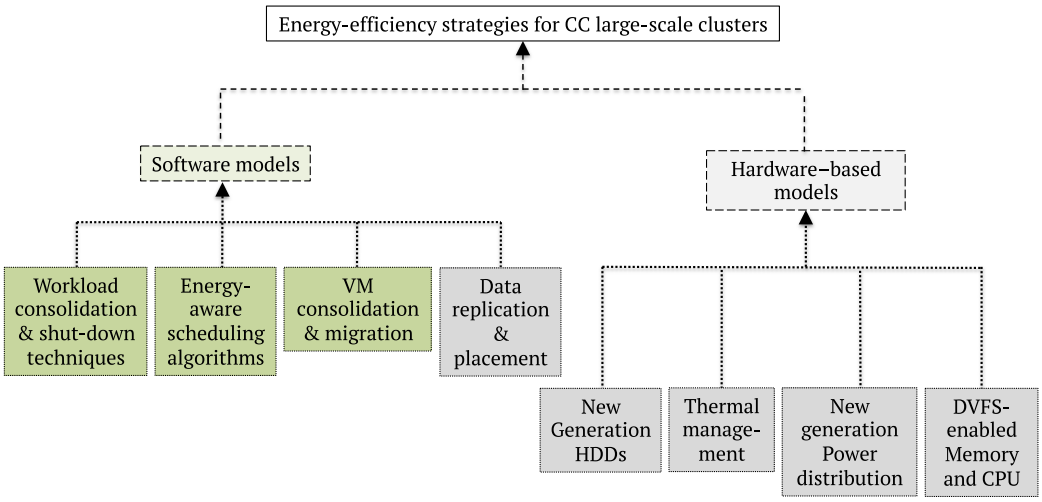

Fig. 2. Generic taxonomy of energy-efficiency techniques for clusters. The strategies in studied in this paper are marked in green.

There are two main categories of schedulers defined in that taxonomy, namely software and hardware-based models. The former focus on the improvement of several pieces of the physical infrastructure of the cloud, such as cooling equipment and thermal management, power distribution and hardware. The latter, on the other hand, focuses on the development of software strategies which make an smarter use of the physical computational nodes.

In this paper, we study the following three classes of software-based models:

(a) Algorithms focused on the consolidation of workload and shut-down of idle nodes.
(b) Energy-aware scheduling algorithms; and
(c) VM scaling and migration algorithms;

These three classes have been chosen since they are the core of resource-managing systems. Resource managers are the responsible for the allocation and execution of tasks, the main goal of computational clusters. For the very same reasons, data replication and placement models have not been covered, since they only affect to distributed file systems.

The first considered category contains the schedulers defined for the reduction of energy consumption in computational clusters, which manly focus on the consolidation of the workload. That consolidation is necessary for the proper estimation of the number of idle nodes in the physical cloud clusters and switch them into sleeping mode [7,21,23]. In [7], Berral et al. propose a consolidation model that combines

(a) the shut-down of idle servers;
(b) power-aware workload-consolidation algorithms; and
(c) machine-learning techniques

to improve energy-efficiency in computational clusters. The computational cluster with 400 nodes is simulated using the OmNet++ simulator. The results obtained show that about 10% of energy consumption can be reduced without negatively impacting on SLAs. In [21] the authors developed the static power-efficiency policies based on the idea of deactivation of the idle nodes in the realistic environments. The SCORE simulator [20] was defined and used for simulation the the large-scale computational clusters with 5000 machines and for heterogeneous workloads. 20% energy reduction results are shown without notably impacting on data-center performance. The SCORE simulator was also used for evaluation of the energy-aware scheduling strategies based on the genetic algorithms with additional scheduling criteria such as security requirements defined by the end users [23]. We defined a realistic cloud environment with 1000-nodes computational nodes and executed the realistic heterogeneous workloads. The implemented scheduling model allowed to save up to 45% of the consumed energy.

A substantial part of the efforts on improving energy-efficiency has been directed towards energy-aware scheduling strategies that could lead to powering off idle nodes, such as [29,33,37]. In [37], Lee et al. present two energy-aware task consolidation heuristics. These strategies aim to maximize resource utilization in order to minimize the wasted energy used by idle resources. To this end, these algorithms compute the total cpu time consumed by the tasks and prevent a task being executed alone. In [33], Juarez et al. propose an algorithm that minimizes a multi-objective function which takes into account the energy-consumption and execution time by combining a set of heuristic rules and a resource allocation technique. This algorithm is evaluated by simulating DAG-based workloads, and energy-savings in the range of [20-30%] are shown. In [29], Jakóbik et al. propose energy-aware scheduling policies and methods based on Dynamic Voltage and Frequency Scaling (DVFS) for scaling the virtual resources while performing security-aware scheduling decisions.

In addition, different techniques of energy conservation such as VM consolidation and migration [4–6,48] are also proposed. In [5], Beloglazov et al. describe a resource management system for virtualized cloud computational clusters that aims to lower the energy consumption by applying a set of VM allocation and migration policies in terms of current CPU usage. This work is extended by focusing on SLAs restrictions in [4] and by developing and comparing various adaptive heuristics for dynamic consolidation of VMs in terms of resource usage in [6]. These migration policies are evaluated by simulating a 100-node cluster. Energy reductions up to approximately 80% are shown with low impact on quality of service and SLAs. In [48] a Bayesian Belief Network-based algorithm that aims to allocate and migrate VMs is presented. This algorithm uses the data gathered during the execution of the tasks in addition to the information provided at submission time in order to decide which of the virtual machines are to be migrated when a node is overloaded.

In this work we present a different approach to the energy-aware scheduling problem: we model the concurrency between energy efficiency and performance as rival players in a Stackelberg-Game model. This game-based model enables us to balance optimally the trade-off between these two opposite needs in energy-aware computational clusters.

The main simulator characteristics necessary to implement the Stackelberg-Game model are the following:

(a) The simulation tool must be energy-aware and provide the tools to measure the energy consumption and reduction;
(b) The simulation tool must provide several already-implemented scheduling models;
(c) The simulation tool must provide several already-implemented scheduling algorithms;
(d) The simulation tool must provide several already-implemented energy-efficiency policies based on the shut-down of idle nodes; and
(e) The simulation tool must be performant when large-scale computational clusters (thousands or even tens of thousands of nodes) are evaluated

. As presented in this Section, the SCORE simulator fulfills the majority of requirements. In this work we extend the SCORE simulator in order to implement the Stackelberg-Game model.

Differently to most of the studied strategies and simulation tools which implement them, which rely on static scheduling strategies for the consecution of energy efficiency, we model the trade-offs of energy-efficient computational clusters, that are performance and energy efficiency, as the sides of this game. The application of the proposed model results on the balance between fast and reliable task execution and low energy consumption. Hence, the major contributions of this work include a model for the dynamic application of energy-efficiency policies based on the Stackelberg-Game model, as well as a trustworthy simulation tool, GAME-SCORE, that implements this model.

3. GAME-SCORE simulator

In this section we define the GAME-SCORE simulator, which is the extension of SCORE [20] simulator, following the same design pattern: a hybrid approach between discrete-event and multi-agent simulation. The main aim of the GAME-SCORE is the simulation of energy-efficient IaaS of the clouds. However, this simulation tool has one limitation that may have a negative impact for the reduction of the energy consumption in computational clusters: the application of energy policies is made statically. Hence, only one static energy policy is applied at the beginning of the experiment and cannot be changed in runtime. This makes the simulation tool sub-optimal for realistic heterogeneous workloads and changing operation environments such as those present in Cloud-Computing scenarios.

In the following sections, we describe in detail the developed simulation tool and its modules. This simulation tool enables us to dynamically choose between a catalog of energy-efficiency policies that shut-down idle machines in runtime. In addition we present, as a realistic use case, an algorithm based on the Stackelberg Game which makes use of this feature. However, any other strategies aiming to dynamically switch between a set of energy-efficiency policies and/or scheduling algorithms could be easily implemented with this new simulation tool.

3.1. GAME-SCORE components

The GAME-SCORE base architecture is composed of two main modules:

- **Core Simulator Module**, responsible for executing the experiments, and is composed of three submodules:
  - **Workload generation**, responsible for the generation of the synthetic workloads, either based on statistical distributions or on real-life workload traces.
  - **Core engine**, responsible for the creation of the simulation environment, cluster, and the multiple scheduling agents. This engine is the module which actually runs the simulation.
  - **Scheduling module**, responsible for the assignation of tasks to worker nodes, as well for the implementation of several scheduling framework models.
- **Energy-Efficiency Module**, responsible for the implementation of the energy-efficiency policies based on the shut-down of idle machines.

In addition, we extended this base architecture to implement the Stackelberg Game process as means to dynamically switch

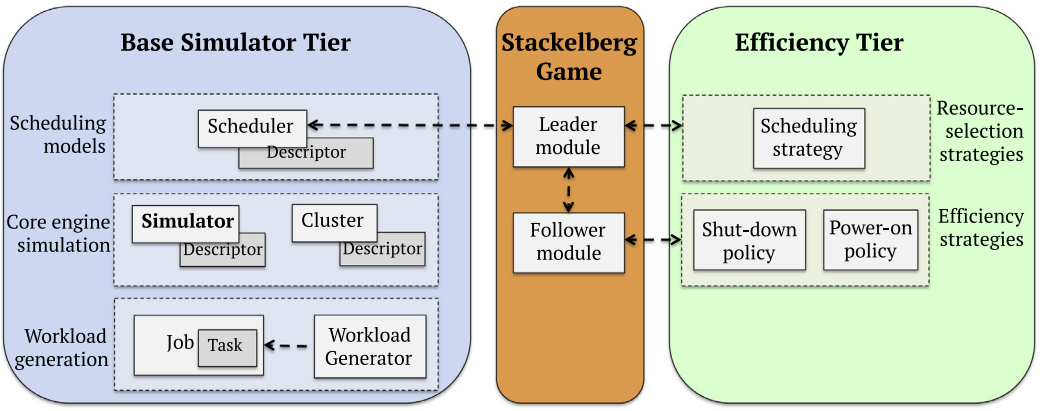

Fig. 3. GAME-SCORE architecture

between energy-efficiency policies. To this aim, we developed the following modules which negotiate between the *Scheduling* module and the *Energy-Efficiency* module:

(a) a Central Energy-efficiency Manager module which governs the catalog of energy-efficiency policies;
(b) a Stackelberg-Game manager which implements the concurrency-based model;
(c) a module for the Leader player to apply its decisions; and
(d) a module for the Follower player to apply its decisions;

. The resulting architecture of the proposed simulator is shown in Figure 3. The GAME-SCORE source code is publicly available at: https://github.com/DamianUS/game-score.

3.2. Shut-down decision policies

We assume in our model that the energy conservation policies do not have a notable negative impact on the performance of the whole computational cluster Therefore we define in our model a *Central Energy-efficiency Manager* that decides the power-off strategy to be applied, which deactivates the servers in an idle mode. It should be noted that *Always* strategy cannot be kept active when a machine computes tasks and send/receive data. In the case of huge workloads, where tasks and data may leave and arrive dynamically from and to the cloud servers, the active servers may be overloaded and the whole task execution process can be significantly delayed. Therefore, there is a need of the development of the decision model which allows us to activate the *Always* power-off strategy in the optimal periods. The following shut-down decision policies have been implemented in our model:

- **Margin** – this decision strategy activates the *Always* power-off strategy only if, at least, a specified amount of resources (servers) is ready to accept the incoming tasks.
- **Random**– in this case, the *Always* power-off strategy is activated randomly. This strategy is usually defined together with the *Never* shut-down policy, where all servers are kept in the active mode (it happens usually in realistic cloud computational clusters) and the *Always* shut-down scenario, where all idle machines are switched-off.
- **Exponential** – in this strategy, the *Always* shut-down strategy is activated depending on the probability of one (large) incoming task of oversubscribing the available resources. This probability is computed by the means of the Exponential distribution.
- **Gamma** – in this case, the *Always* shut-down strategy is activated depending on the probability of incoming tasks (in a given window time) of oversubscribing the available resources. This probability is computed by the means of the Gamma distribution.

The utilization of the *Energy-efficiency Manager* in our model does not guarantee the fair reduction of the energy consumed by the cloud system. Therefore, we define another component of the model, that is the *Scheduling Manager*. This component allows the optimal schedule of tasks onto the cloud servers based on the energy-conservation criterion. In this work, we focus on the problem of the independent tasks scheduling. We use the genetic cloud scheduler developed in [31] and ETC Matrix scheduling model described in [29]. The makespan constitutes the most representative parameter of the performance, and hence it becomes the scheduling goal.

3.3. Stackelberg Game used for scheduling decisions

In the model presented in this work we used the Stackelberg Game played by two opponents *Scheduling Manager* – Leader and Energy-efficiency Manager –Follower. Similar strategies have been used for energy-aware resource allocation [3]. The interaction between the previously described modules in this game is shown as a sequence diagram in Figure 4

As a model for balancing scheduling efficiency and energy minimization we used a non-zero symmetric game, defined by:

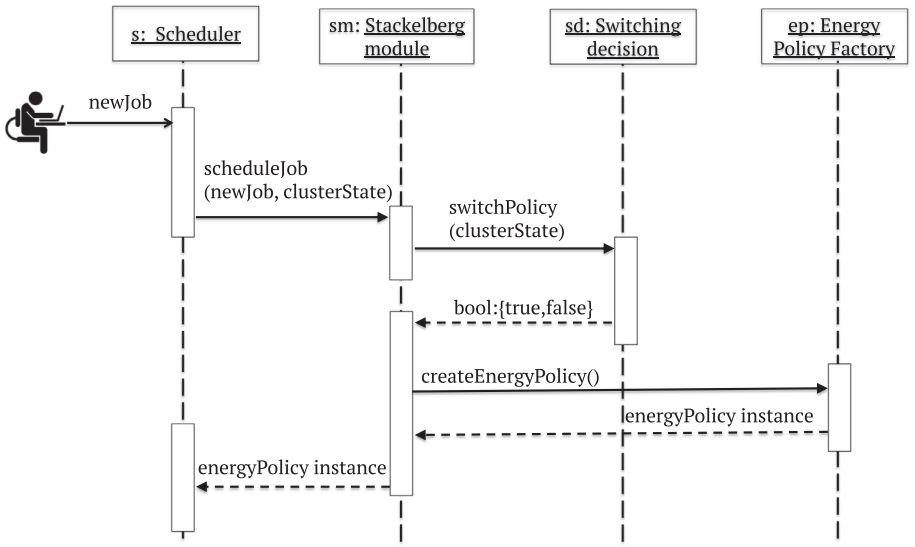

Fig. 4. Sequence diagram of the interactions between modules of the Stackelberg Game workflow

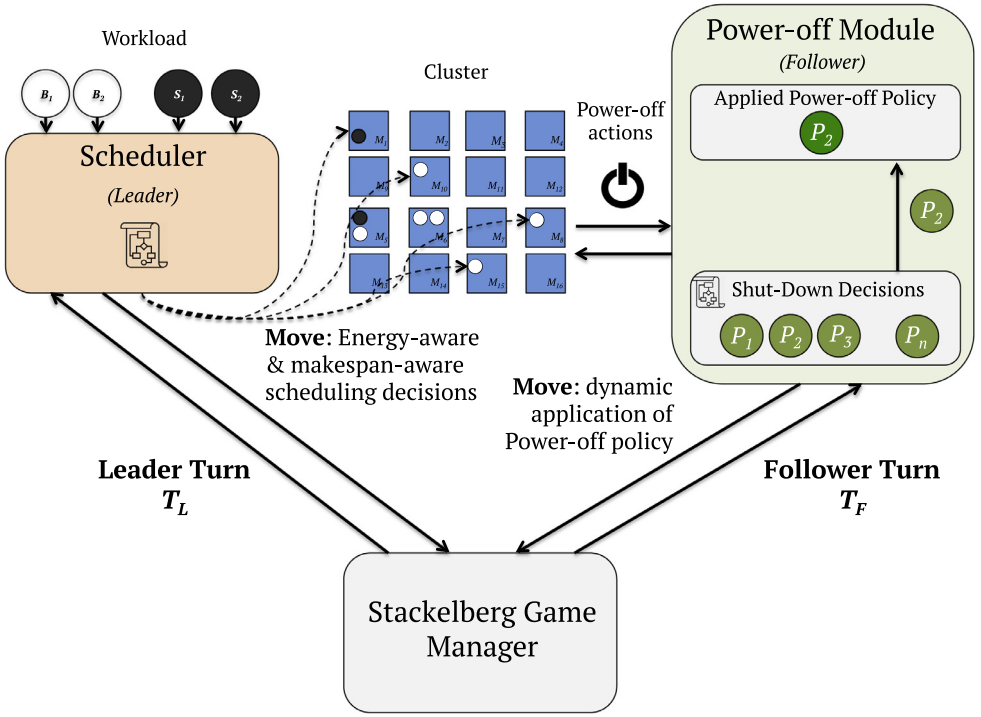

Fig. 5. Stackelberg Game workflow, scheduling workflow, *B* - Batch type task, *S* - Service type task, *M* - Virtual Machine

$$\Gamma_n = ((N, \{S_i\}_{i\in N}, \{Q_i\}_{i\in N}) \tag{1}$$

where $N = \{1, 2\}$ denotes the set of players, $\{S_1, S_2\}$ $(cardS_i \geq 2; i = 1, 2)$ denotes the set of strategies for them $\{H_1, H_2\}$; H_i: $S_1 \times \times S_2 \rightarrow \mathbb{R}$; $\forall_{i=1,2}$ denotes payoff functions for each player.

Both players are making decisions according their payoffs. A decision is a selection of one single action from the set of possible actions. Possible actions are defines as elements of strategy sets f$\{S_1, S_2\}$. The sets of actions for each player are chosen to be beneficial for this player. The payoff function is measuring the quality of actions by assigning the real value to each set of decisions. In the model pure strategies and mixed strategies are considered, see [54]. Let us denote by s_i the **Pure strategy** of the player i and the set of all pure strategies specified for player i is denoted by S_i. The **mixed strategy of the player** i is denoted by $\sigma_i \in S_i \subset \Delta S_i$ and allows to randomize over pure strategies:

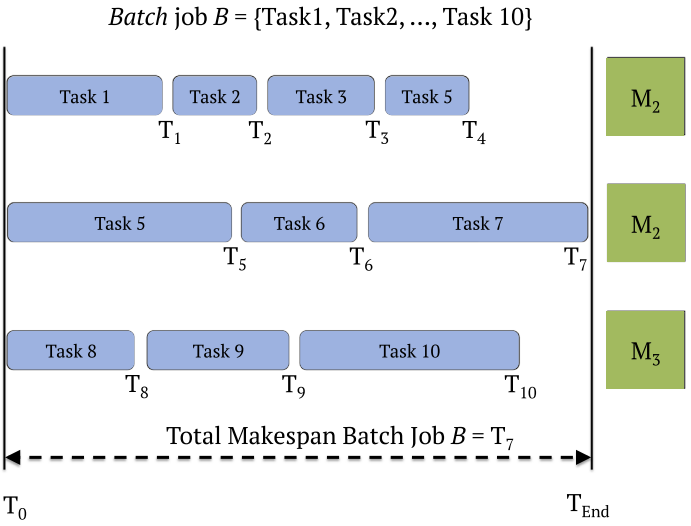

Fig. 6. Example of Leader player (Scheduler) makespan computation

$$\sigma_i = \{\sigma_i(s_{i_1}), \sigma_i(s_{i_2}), \ldots, \sigma_i(s_{i_m})\}, \tag{2}$$

where $\sigma_i(s_i)$ denotes the probability that the player i choses the pure strategy s_i.

In Stakelberg Games (SG), the leader of the game is privileged to play first, and the second players (the follower) are obliged to make their decisions after him [54]. In our model we proposed o non zero sum game, to allow the leader and the follower define their strategies separately.

The leader of the game, *Scheduler* component is making decisions how to dispatch tasks among the *Computing Nodes*. These *Computing Nodes* are grouped into *Computational Units*, denoted as CU_1, CU_2, ...CU_P. Incoming *Jobs* are composed of a set of independent *Tasks* which can be executed in parallel.

Single decision of the leader is a schedule calculated for the given batch of tasks and available set of *Computational Units*. The cardinality of the strategy set for the leader equals all possible schedules. Let us denote this number by P possible decisions. The strategy vector $\sigma_i(s_i)$ represents the probability for a *Job* to be assigned to the CU_p, for $p = 1, 2, \ldots, P$. The s_i may be taken from the set 1, 2, ..., P.

The expected payoff of the game leader is depends on the completion time of all the *Tasks* in the scheduled *Job*, thus, the makespan of that *Job*, as shown in Figure 6. The leader plays to in order to minimize the makespan. In our model we used a Monolithic Scheduler [35] which makes scheduling decisions based on the Expected Time to Compute (ETC) matrix, defined as follows:

$$ETC = [ETC[j][i]]_{j=1,\ldots,n}^{i=1,\ldots,m^p} \tag{3}$$

where

$$ETC[j][i] = wl_j / cc_i^p \tag{4}$$

In this equation, cc_i^p is the computational capacity of the i-th Computing Node (CN) in the pth Computing Unit (CU) in Giga Flops per Second (GFLOPS) and wl_j represents the workload of j-th task in Flops (FLO); n and m^p denote the number of tasks and number of Computing Nodes in the pth Computing Unit respectively, see [32]. The makespan is defined as follows

$$C_{\max}(wl_1, \ldots, wl_n, cc_1^p, \ldots, cc_{m^p}^p, m^p, n, p) = \tag{5}$$

$$= \min_{S \in Schedules} \left\{ \max_{j \in Tasks} C_j \right\}, \tag{6}$$

where C_j is the completion time of the j-th task. *Tasks* represents the set of tasks in the *Job*, and *Schedules* is the set of all possible schedules that can be generated for the *Tasks* of that *Job*. The shortest makespan is calculated by using the Expected Time to Compute (ETC) matrix. In this matrix, the cell in the ith row and the jth column represents the completion time of the jth task if is it executed on the ith CN.

The optimal schedule is is the best decision for the game leader, therefore utility function value for the game leader is defined as:

$$H_1(\sigma_1, \sigma_2) = \sum_{p=1,\ldots,P} \sum_{l=1,..,L} \sigma_1^p \sigma_2^l C_{\max}(wl_1, \ldots, wl_n, cc_1^p, \ldots, cc_{m^p}^p, m^p, n, p) \tag{7}$$

where L indicates the number of decisions that may be taken by the game follower. The value of the makespan depends on the computational power of the CNs. These parameters may be modified by the follower player:

$$H_1(\sigma_1, \sigma_2)= \sum_{p=1,..,P} \sum_{l=1,..,L} \sigma_1^p \sigma_2^l C_{\max}(wl_1, ...,wl_n, cc_1^p(l), ...,cc_{m^p}^p(l), m^p(l), n, p(l)) \quad (8)$$

The follower in the game is the *Central Energy-efficiency Manager*. It applies energy policies to all the CNs in the computational cluster. The follower may decide about the $cc_i^p(l)$ and the $m^p(l)$. The payoff for the follower is defined as the energy consumed by the CC system for the execution of the schedule computed by the *Scheduler*:

$$H_2(\sigma_1, \sigma_2)= \sum_{i=1,..,m} \sum_{j=1,..,n} \sigma_1^j \sigma_2^i E(wl_1, ...,wl_n, cc_1^p, ...,cc_{m^p}^p, m^p, n, p, schedule) \quad (9)$$

After the follower has made his decision, the leader is considering new batch of tasks. It computes next schedule and the game is repeated. Both the payoffs depends on both players moves.

In order to calculate the the follower payoff, the following equation was introduced:

E_{total} is the total energy consumed by particular *Job*, t_{idle}^i is the idle time of CU after it calculated assigned tasks ; t_{busy}^i is the time that the *i*-th CN is devoting on computing tasks; P_{idle}^i is the power a CN requires to remain in a idle state; P_{busy}^i is the power a CN consumes during computing tasks.

The time that the *i*-th CN spends on computing tasks depends on the schedule that was decided be the game leader:

$$t_{busy}^i = max_{j \in Tasks\ scheduled\ \ for\ \ CN_i} C_j \quad (10)$$

and the idle time of the *i*-th CN may be calculated as follows:

$$t_{idle}^i = C_{max} - t_{busy}^i \quad (11)$$

This model assumes that the next batch of tasks may be scheduled is the previous batch was calculated. Assuming that, the total energy consumed may be calculated in the following way:

$$E_{total} = \sum_{i=1}^m \int_0^{C_{max}} Pow_{CN_i}(t)dt= \sum_{i=1}^m (P_{idle}^i * t_{idle}^i + P_{busy}^i * t_{busy}^i + P_{sleep}^i * t_{sleep}^i + P_{off}^i * t_{off}^i + P_t^i * t_t^i) \quad (12)$$

where P_{sleep}^i, t_{sleep}^i is power consumed during sleeping mode, and time spend in this state, P_{off}^i, t_{off}^i is during power consumed during being powered of (assumed as zero) and time spend in this state. Values P_t^i and t_t^i are accumulated power and time spend during all transitions from one state to another.

Our model allows competition between two aims: to compute tasks as fast as possible and to apply the more optimal power states to the CNs in order to maximize the energy efficiency. q The core off the game is to solve two optimization problems. First is to find the the best decision for the game leader, that is finding the solution of the problem

$$argmax_{\sigma_1^1,\sigma_1^2,...,\sigma_1^P} \sum_{p=1,..,P} \sum_{l=1,..,L} \sigma_1^p \sigma_2^l C_{\max}(wl_1, ...,wl_n, cc_1^p, ...,cc_{m^p}^p, m^p, n, p) \quad (13)$$

with the following constraints

$$\sum_{p=1,..,P} \sigma_1^p = 1 \quad (14)$$

$$\forall\ \sigma_1^p: \sigma_1^p \in [0, 1] \quad (15)$$

and second, to find the best decision for the game follower:

$$argmax_{\sigma_2^1,...,\sigma_2^m} \sum_{i=1,..,m} \sum_{j=1,..,n} \sigma_1^j \sigma_2^i E(wl_1, ...,wl_n, cc_1^p, ...,cc_{m^p}^p, m^p, n, p, schedule) \quad (16)$$

with the following constraints

$$\sum_{i=1,..,m} \sigma_2^i = 1 \quad (17)$$

$$\forall\ \sigma_2^i: \sigma_2^i \in [0, 1] \quad (18)$$

Considering pure strategies problem (16)-(18) may be solved by the direct search.The best strategy is one of the m possible problem solutions. The number of possible solutions for the problem (13)-(15) is P! and the problem of finding the best schedule is consider to be NP-hard. Therefore we applied Genetic Algorithm search method for finding suboptimal solution. During such

procedure the suboptimal schedule is found and the strategy vector equals one for that schedule. Probabilities of all other schedules are assumed to be equal zero. If several sub-optimums were found each is granted the same probability that equals one divided by the number of them.

Considering mixed strategies optimization problem (16)-(18) is may be solved by one of the classical techniques, for example the simplex method. The space of the *argmax* search is $[0, 1]^m$. Mixed strategies for the problem (16)-(18) was not implemented as the part of this research.

3.4. A simple theoretical example of the proposed algorithm

For the illustration of the game implemented in the GAME-SCORE simulator, lets consider the simplest possible environment, consisting in only two computing units. Each unit is equipped with only one node. The computing capacities of the CUs are: CU_1 $cc_1^1 = 1$ GFLOPS/sec. and CU_2 $cc_1^2 = 1$ GFLOPS/sec.

The Jobs that will be considered by the scheduler are composed of three independent tasks, having the following workload: $wl_1 = 1$ FLO, $wl_1 = 2$ FLO and $wl_1 = 4$ FLO.

The Energy-Efficiency manager may choose only from two energy policies: keeping all unused CUs into idle state (strategy 1), and always switch all idle CUs to sleep mode (strategy 2). For the clearance of the presentation the transition time and energy are omitted and the rest of characteristics are assumed in the following form:

$$P^1_{idle} = 2 \text{ MWh} \qquad P^2_{idle} = 4 \text{ MWh}$$
$$P^1_{busy} = 10 \text{ MWh} \qquad P^2_{busy} = 20 \text{ MWh}$$
$$P^1_{sleep} = 0.1 \text{ MWh} \qquad P^2_{sleep} = 0.2 \text{ MWh}$$

In the previous round, the result of the strategy computed by the *Follower* player was: $\sigma_2^1 = 1/3$, $\sigma_2^2 = 2/3$. Now it is the *Leader*'s turn.

The algorithm for the application of the next Stackelberg game round is composed of the following steps:

1. Computation of the *Leader*'s move. This step is, in turn, composed of the following substeps:
   (a) Computation *ETC* matrix, 4, based on the characteristics of the incoming *Job*:

$$ETC = \begin{bmatrix} 1/1 & 2/1 & 4/1 \\ 1/2 & 2/2 & 4/2 \end{bmatrix}$$

   (b) Find possible schedules. In this simple example the utilization of a genetic algorithm to solve the NP-hard problem is not necessary. The optimal schedule may by computed by means of brute force. We will represent the schedules as follows: (tasks assigned to CU_1|tasks assigned to CU_2). All possible schedules are the following.:
      (1) $s_1 = (1, 2, 4| -)$ and $s_2 = (-|1, 2, 4)$. This schedule represents the mapping of all tasks to the selected *CU*
      (2) $s_3 = (1, 2|4)$, $s_4 = (1, 4|2)$, $s_5 = (4, 4|1)$. This schedule is the result of the assignation of one task to CU_2, while the rest of the tasks are assigned to CU_1
      (3) $s_6 = (1|2, 4)$, $s_7 = (2|1, 4)$, $s_8 = (4|1, 2)$. This schedule is the result of the assignation of one task is assigned to CU_1, while the rest of the tasks are assigned to CU_2.
   (c) Computation of the payoff function 8. To this aim, we need to calculate the makespan for all the possible schedules, based on *ETC* matrix:
      (1) $C_{max}(s_1) = 1/1 + 2/1 + 4/1 + 0 = 7$ and $C_{max}(s_2)$=0 + 1/2 + 2/2 + 4/2 = 3.5) sec.
      (2) $C_{max}(s_3) = 1/1 + 2/1 + 4/2 = 5$, $C_{max}(s_4) = 1/1 + 4/1 + 2/2 = 6$, $C_{max}(s_5) = 4/1 + 1/2 + 2/1 = 3.5$ sec.
      (3) $C_{max}(s_6)$=(1/1 + 2/2 + 4/2 = 4, $C_{max}(s_7) = 2/1 + 1/2 + 4/2 = 4.5$, $C_{max}(s_8)$=4/1 + 1/2 + 2/2 = 6.5) sec.
   (d) Find the schedule that maximize the payoff, which means the schedule that results in the shortest makespan,6 for the particular *Job*. If the run of the Genetic Algorithm results in a set of equal suboptimal schedules, then we can assign probabilities to them to avoid the "local minima" trap. According to the resulting makespans for the schedules s_2 and s_5, we may randomize the selection, and set the mixed strategy vector in the following form:

$$(sigma_1^1, sigma_1^2, ...,sigma_1^8) = (0, 1/2, 0, 0, 1/2, 0, 0, 0)$$

   Therefore the *Leader*'s payoff function is:

$$\begin{aligned} &H_1(\sigma_1, \sigma_2)= \\ &\textstyle\sum_{p=1,...,8} \sum_{l=1,2} \\ &\sigma_1^p \sigma_2^l C_{\max}(s_p)= \\ &1/2*1/3*3.5 + 1/2*1/3*3.5 + 1/2*2/3*3.5 + 1/2*3/3*3.5 \\ &= 4.08(3) sec. \end{aligned} \tag{19}$$

   (e) Selection of a single action: We chose it randomly, with equal probability among schedules s_2 and s_5. In this case, the schedule

number 2 was selected.

(f) Submission of this schedule to both the *CU* and the *Follower* player.

2. Computation of the *Follower*'s move. This step is, in turn, composed of the following substeps:

(a) Computation of the *Idle* and *Busy* times, see eq. 13 and 14 respectively for all the CUs. Given the resulting schedule: $s_2 = (-|1, 2, 4)$ results in $t^1_{busy} = 0$ sec. and $t^2_{busy} = 3.5$ sec.

(b) Computation of the total energy consumption 21 of the particular schedule for all available energy policies:

- If the policy number 1 is applied:

$$E_{total}= \sum_{i=1}^{2} (P^i_{idle}*t^i_{idle} + P^i_{busy}*t^i_{busy})= P^1_{idle}*t^1_{idle} + P^1_{busy}*t^1_{busy} + P^2_{idle}*t^2_{idle} + P^2_{busy}*t^2_{busy}= 2*3.5 + 0 + 0 + 4*3.5 = 21KWh \quad (20)$$

- If the policy number 2 is applied:

$$E_{total}= \sum_{i=1}^{2} (P^i_{sleep}*t^i_{sleep} + P^i_{busy}*t^i_{busy})= P^1_{sleep}*t^1_{sleep} + P^1_{busy}*t^1_{busy} + P^2_{sleep}*t^2_{sleep} + P^2_{busy}*t^2_{busy}= 0.1*3.5 + 0 + 0 + 4*3.5 = 14.35KWh \quad (21)$$

(c) Find the energy policy that minimizes the energy consumption, that is, the maximization of the payoff function. In this particular example, the optimal energy policy is the policy number 2: $\sigma^1_2 = 0$, $\sigma^2_2 = 1$.

(d) If necessary, we may randomize over several suboptimal solutions. In this simple example, this step is not necessary.

(e) Computation of the *Follower's* payoff 22, as follows:

$$H_1(\sigma_1, \sigma_2)= \sum_{p=1,\ldots,8} \sum_{l=1,2} \sigma_1^p \sigma_2^l E_{total} = 1/2*1 + 14.35KWh \quad (22)$$

(f) Selection of the decision made by the system: The application of policy number 2 (only).

(g) Application of this energy policy to the *CU*s, and return the decision to the game *Leader*.

These steps are repeated for every incoming *Job*. In real-life scenarios we have to employ a Genetic Algorithm (GA) to sole the NP-hard problem of the *Leader*'s move (the resulting scheduling) and a Simplex method to compute the follower strategy.

4. Experimental analysis

In this work, we aim to empirically demonstrate that the proposed simulation tool, GAME-SCORE, as well as the implemented energy-aware scheduling algorithm based on the Stackelberg-Game model may have a notable positive impact in terms of energy efficiency and performance, as well as an optimal balance between them.

In the following subsections we present the scheduling framework and algorithm considered, the simulation environment, the parameters and KPIs under evaluation and the considered realistic scenarios where we compare our proposal to static energy-efficiency policies.

4.1. Scheduling framework

In this experimental analysis, we employ a *Monolithic centralized scheduler* model. This scheduling model [28] works very well under low job-arrival rate conditions, such as long-running MapReduce jobs [12], since latencies of seconds or minutes [15] are acceptable in this context. This kind of scheduler can perform high-quality scheduling decisions [13,56] by examining the whole cluster state to determine the performance impact of hardware heterogeneity and interference in shared resources [25,38,41,46,55], and thereby it can choose the best resources for each task. This model leads to higher machine utilization [52], shorter execution times, better load balancing, more predictable performance [14,57], and increased reliability [44]. The scheduling process for the monolithic centralized scheduler is illustrated in Figure 7

Various parameters may characterize the jobs that make up the workloads of the Cloud-Computing system [40]. In this work, we focus on the following main attributes of the job j_k:

- **Job inter-arrival time** TI_{j_k}- represents the time elapsed between two consecutive jobs (j_k) submissions of the same workload type

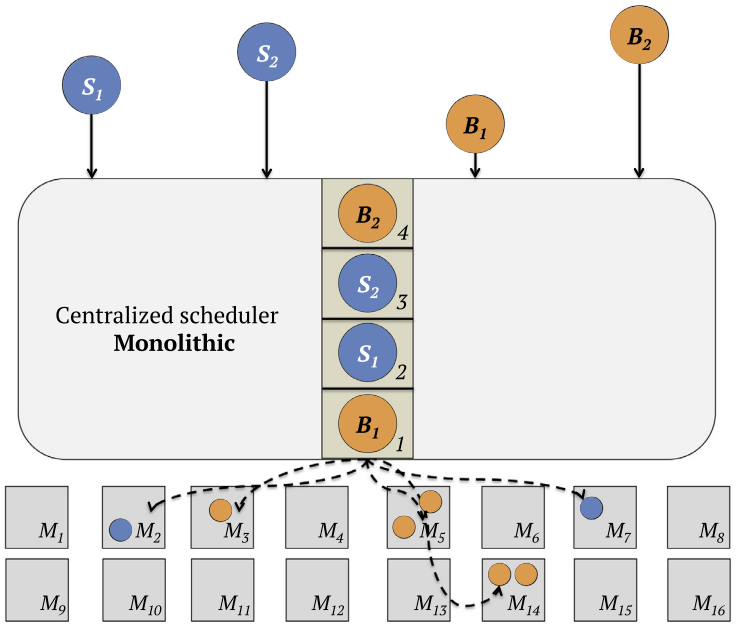

Fig. 7. Monolithic centralized scheduling workflow, B - Short-running Batch task, S - Long-running Service task, M - Machine

W. Thus, the number of jobs to be scheduled and executed by the Cloud-Computing system in a given time is defined by this parameter.

- **Job duration time** TD_{j_k} - is the completion time of the j_k job in the Cloud-Computing system
- **Number of tasks** TT_{j_k} - is the number of tasks that makes up the job j_k.

The performance efficiency of any Cloud-Computing *Scheduler* is related to the number of jobs that can be scheduled in a given time as well as the quality of the scheduling decisions. We consider the processing time (makespan) of the total set of jobs, that is, the workload W_s, as the main key performance indicator of the scheduling quality.

Usually, the term *workload* is conformed by the whole set of inputs related to Cloud-Computing systems, such as: applications, service packages and related data required by tasks. In Cloud Computing, such inputs are often submitted by the Cloud-Computing users by means of cloud services hosted in Cloud clusters. It should be also borne in mind that Cloud-Computing workloads are not usually composed of real-time applications.

4.1.1. Genetic Algorithm for searching optimal schedule

The scheduling of tasks in cloud-computing computational clusters constitutes an NP-complete problem [50], whose complexity depends on the features considered [35], such as:

(a) the number of scheduling 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 [26,30], 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;
(e) the population is composed of *m* individuals and represents a schedule for all *n* tasks;
(f) the fitness function depends on the optimization objectives

. 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, as

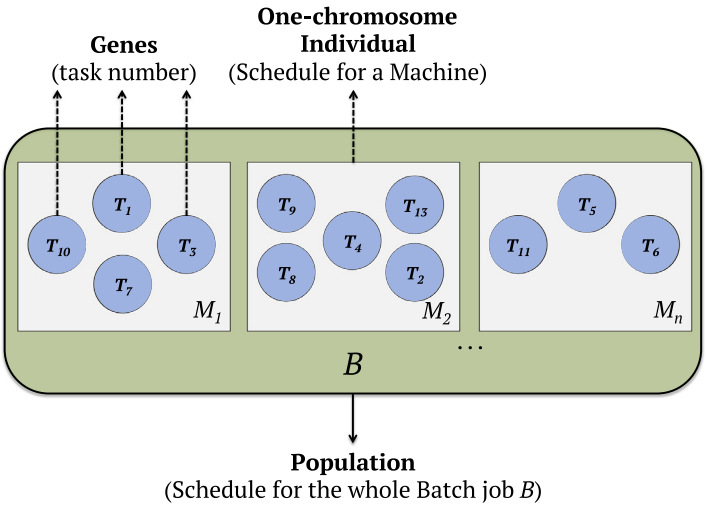

Fig. 8. Genetic algorithm model for energy-aware scheduling

shown in Figure 8.

4.1.2. Workload types

The quality of the scheduling process has a notable impact in Cloud-Computing systems, both on the overall quality of the cloud services as well as on the fulfillment of Service Level Agreements (SLAs).

We can classify the workloads to be processed according to two main characteristics:

- The internal architecture of the workload, that is, the relationship between jobs in the same workload. In this model, the kind and number of jobs that form the cloud applications, as well as the dependencies between them describe the whole workload. Hence, such jobs may be processed as a Directed Acyclic Graph [9], in parallel, and sequentially.
- The processing model of the jobs. In this model, we consider the following type of jobs:
  - **Batch workload *BW***– this workload is composed of jobs that have a strictly-specified job arrival, start and completion times since these jobs are designed to perform a given computation and then finish.
  - **Service workload *SW*** – this workload is defined as a set of long-running jobs which usually need a higher amount of resources. These jobs have a determined job arrival and start time, but the completion time is not a priori determined.
    As real-life examples of the aforementioned workloads, MapReduce jobs [12] are classified as belonging to the *Batch* workload *BW*. On the other hand, long-running services such as BigTable [10] and HDFS [47], and web servers make up the *Service* workload *SW*.

In this experimental analysis, we focus on the evaluation of the proposed simulation tool and the dynamic application of energy-efficiency policies based on the Stackelberg-Game model. To this aim, We created heterogeneous and realistic workloads composed of *BW* and *SW* jobs based on the trends of the industry and Google Data-Center traces present in [1,16].

4.2. Simulation environment

We used the GAME-SCORE simulator to perform a simple experiment that simulates seven days of operation time of a computational cluster composed of 1,000 heterogeneous machines of 4 CPU cores and 8GB RAM and one central monolithic scheduler. Each machine has the following features:

- **Computing profile**: Differences in the processor's computing power has been mocked by generating randomly a [1x - 4x] computing speed factor. Thus, a given computing node may be, as a maximum, four times faster than the slowest one.
- **Energy profile**: Processor's power consumption heterogeneity has been simulated by generating randomly a [1x - 4x] energy consumption factor. Thus, a given machine *M* may be (as a maximum) four times more energy-consuming 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.

In this experiment, we chose an heterogeneous day-night patterned mixed workload. This workload, which is composed of 22,000 *Batch* jobs and 2,200 *Service* jobs, uses 30% of the computational cluster computational resources on average, with peak loads that achieve 60% of utilization.

In order to create this realistic cloud scheduling scenario, the following workload parameters were considered:

- **Job inter-arrival**: The inter-arrival time TI_{j_k} of *BW* jobs is sampled from an exponential distribution whose mean value is 90 (seconds). For *SW* jobs, this inter-arrival time is sampled from an exponential distribution with a mean value of 900 (seconds).
- **Job structure**: The number of tasks TT_{j_k} for each job in *BW* is sampled from an exponential distribution with a mean value is 50, while the number of tasks for each job in *SW* is sampled from an exponential distribution whose mean value is 9.
- **Task duration**: The duration TD_{j_k} of *BW*-jobs tasks is sampled from an exponential distribution whose mean value is 90 (seconds). For *SW*-jobs tasks, this duration is sampled from an exponential distribution with a mean value of 2000 (seconds).
- **Resource usage**: in *SW*-jobs tasks consume 0.3 CPU cores and 0.5 GB of memory, in *SW*-job tasks consume 0.5 CPU cores and 1.2 GB of memory.

4.3. Energy-efficiency indicators

For the analysis of the impact in terms of the reduction of the energy consumption of the Cloud-Computing system, the following energy-efficiency parameters are considered:

- E_c **– Energy consumed**: This parameter represents the total energy used by the Cloud-Computing system.
- E_s **– Energy saved**: This parameter represents the total energy saved by the Cloud-Computing system compared to the current[1] operation energy consumption.
- ***SD* – Number of shut-downs**: The total number of shut-down operations performed over all the resources during the simulated operation time. This parameter can be related to the hardware stress due to booting actions.
- $E_s SD$ **– Energy saved per shut-down**: This parameter computes the energy saved against the shut-downs performed. Hence, it shows the *efficiency* of the shut-down actions performed.
- ***IR* – Idle resources**: This parameter represents the amount of resources turned on but not in use.

4.4. Scheduling efficiency indicators

For the comparison and evaluation of the performance of the Cloud-Computing system, we define the following key performance indicators (KPIs):

- JQT_{fi} **– Job queue times until first scheduled**: This parameter represents the time a job needs to wait in queue until it scheduled for the first time.
- JQT_{full} **– Job queue times until fully scheduled**: This parameter represents the time a job needs to wait in queue until it is fully scheduled.
- ***SBT* – Scheduler busy time**: This parameter represents the total time spent by the scheduler performing scheduling operations.
- MS_t **– Final makespan**: This parameter represents the total time spent by jobs in the Cloud-Computing system on average. It is worth to mention that only the *Batch* workload *BW* has makespan, since *Service* workload *SW* has no determined end, but usually these jobs are killed by operators or automated systems when they are no longer necessary.
- MS_0 **– Epoch 0 makespan**: This parameter makes reference to the makespan of jobs in the first iteration of the genetic algorithm on average.

4.5. Simple example for Always and Never power-off policies in SCORE simulator

In this experiment, we aim to empirically show a simple strategy where a dynamic change of *Power-off* policy could represent a significant improvement of energy-efficiency.

The Stackelberg process described previously is applied for every scheduling decision in the system. In this experiment, the *Shut-down decision policy* used to switch the *Power-off* policy is made based on cluster available resources. Every time that the idle resources exceed a given threshold, the *Always* power-off policy is applied. On the other hand, when the amount of available resources is lower than that threshold, the *Never* power-off policy is applied. The results of the application of the Stackelberg game against the static energy policies are presented in Tables 1 and 2.

This experimentation shows that the Stackelberg model applies a minor negative impact (+ 15%) in terms of queue times compared to the *Never* shut-down decision (17 vs 20 ms.), while the negative impact of the *Always* and *Random* strategies are +160% and 80% (17 vs. 44 and vs. 30.5 ms.) respectively. In terms of energy consumption, the *Stackelberg* model only consumes 10% more energy than the *Always* and *Random* shut-down policies (29 vs 32 MWh). On the other hand, the *Always* and *Random* strategies achieve approximately 7% lower final average makespan time (146 vs. 155 s.) due to the dynamic changes of the *Stackelberg* model.

4.6. Extended example in SCORE simulator

In this section, we extended the simple experimentation presented in Section 4.5. In order to keep results comparable, we reused all the configuration parameters taken for the large-scale CC system shown in Section the 4.5. However, in this experiment the *Central*

[1] Current operation for the same computational cluster and workload, but without applying energy-saving polices

Table 1
Energy-efficiency results for the simple Stackelberg experiment

Strategy	E_c (MWh)	E_s (MWh)	Savings (%)	*SD*	E_sSD (kWh)	*IR* (%)
Static-Never	55.65	0	0	0	N/A	70.71
Static-Always	29.04	26.83	48.03	19,722	1.36	3.49
Static-Random	29.33	26.47	47.44	10,943	2.42	4.49
Stackelberg	**32.24**	**23.65**	**42.32**	**1,665**	**14.21**	**11.11**

Table 2
Performance results for the simple Stackelberg experiment

Strategy	Workload	JQT_{full} (ms)	JQT_{fi} (ms)	*SBT* (h)	MS_t (s)	MS_0 (s)
Static-Never	Batch	17.05	17.02	3.71	142.55	177.65
Static-Never	Service	20.08	20.07	0.12	N/A	N/A
Static-Always	Batch	43.93	19.58	4.25	146.12	185.74
Static-Always	Service	35.11	21.19	0.13	N/A	N/A
Static-Random	Batch	30.50	18.44	4.02	143.98	180.48
Static-Random	Service	33.46	22.03	0.12	N/A	N/A
Stackelberg	**Batch**	**20.40**	**17.62**	**3.75**	**155.04**	**179.63**
Stackelberg	Service	29.91	21.53	0.12	N/A	N/A

Energy-efficiency Manager switches dynamically between the *Never* and the *Always* power-off policies by applying every *Shut-down decision policy* described in Section 3.2. The results obtained are shown in Table 3 and 4.

In general, the Stackelberg process may apply a negative impact in terms of makespan due to that the *Power-off* policy may suddenly change. This change can impact on two consecutive scheduling processes of a single job, which could apply a performance penalty if there are no sufficient resources to immediately execute the job tasks. This negative impact can be mitigated by the scheduler when only one static *Power-off* policy is applied. It should be borne in mind that only *Batch* jobs would suffer from this negative impact since *Service* jobs have no determined finish.

This experimentation shows that the results of the Stackelberg model depends directly on the *Shut-down decision* policy. More conservative probabilistic models, such as Exponential and Gamma, achieve and 45% faster queue times than a *Random* strategy (33 vs 18 and 19 ms.) respectively while consuming approximately 8 and 12% more energy (29.4 vs. 31.5 and 33.8 MWh) respectively. On the other hand, strategies that rely on leaving a security margin of free resources, such as *Margin*, could achieve approximately 40% faster queue times than a *Random* strategy (33 vs 20 ms.), and it would only consume 10% more energy (29.4 vs 32.2 MWh). It can be noticed that conservative strategies such as Gamma apply almost no stress to the hardware, performing approximately 1,000 shut-downs in a week of operation time, which represents 10% of those performed by the *Random* decision policy.

4.7. Results summary

The results obtained show that in general, the Stackelberg-Game-based can balance the trade-offs present in energy-efficient computational clusters: energy-consumption reduction vs. performance.

As showed, our model can notably improve the scheduling performance compared to static energy-efficiency policies while achieving almost the same levels of energy efficiency when the proper energy policies and switching decisions are used. However, the Stackelberg process applies a minor negative impact in terms of makespan due to the sudden changes of the *Power-off* policy applied.

Table 3
Energy-efficiency results for the extended Stackelberg experiment, where the *Always* and *Never* shut-down policies are switched following several decision policies

Strategy	Switch Decision	E_c (MWh)	E_s (MWh)	Savings (%)	*SD*	E_sSD (kWh)	*IR* (%)
Static-Never	N/A	55.65	0	0	0	N/A	70.71
Static-Always	N/A	29.04	26.83	48.03	19,722	1.36	3.49
Stackelberg	Random	29.37	26.43	47.36	11,434	2.31	4.63
Stackelberg	Margin	32.24	23.65	42.32	1,665	14.21	11.11
Stackelberg	Exponential	31.48	24.32	43.49	2,998	8.08	10.05
Stackelberg	Gamma	33.78	22.25	39.71	1,074	20.72	14.63

Table 4
Performance results for the extended Stackelberg experiment, where the *Always* and *Never* shut-down policies are switched following several decision policies

Strategy	Workload	Switch Decision	JQT_{full} (ms)	JQT_{fi} (ms)	SBT (h)	MS_t (s)	MS_0 (s)
Static-Never	Batch	N/A	17.05	17.02	3.71	142.55	177.65
Static-Never	Service	N/A	20.08	20.07	0.12	N/A	N/A
Static-Always	Batch	N/A	43.93	19.58	4.25	146.12	185.74
Static-Always	Service	N/A	35.11	21.19	0.13	N/A	N/A
Stackelberg	Batch	Random	33.29	18.78	9.88	143.07	181.42
Stackelberg	Service	Random	33.17	22.51	0.69	N/A	N/A
Stackelberg	Batch	Margin	20.40	17.62	3.75	155.04	179.63
Stackelberg	Service	Margin	29.91	21.53	0.12	N/A	N/A
Stackelberg	Batch	Exponential	18.92	17.16	9.20	144.08	178.28
Stackelberg	Service	Exponential	24.67	20.39	0.66	N/A	N/A
Stackelberg	Batch	Gamma	18.23	17.12	9.20	163.80	180.00
Stackelberg	Service	Gamma	22.05	20.14	0.66	N/A	N/A

5. Conclusions

In this paper, we presented a new simulation tool called GAME-SCORE which implements method that focus on the balance between two opposite needs of every energy-efficient CC system: high performance throughput and low energy consumption.

The proposed simulation tool and model are based on a non-zero sum Stackelberg Game with the leader player, the *Scheduling Manager*, which tries to minimize the makespan with its scheduling decisions while the follower player, the *Energy-efficiency Manager*, responds to the leader player move with the application of energy-efficiency policies that may shut-down the idle machines. These strategies are represented by the independent utility functions for each player. Our model enables the dynamic application of energy-efficiency strategies depending on the current and predictable workload.

The results of our simple experimental evaluation show that the proposed model perform better than the application of only one energy-efficiency policy, both in terms of energy-efficiency and performance. This means that the Stackelberg Game model can balance better between opposed needs (performance and energy efficiency) and can adapt better to heterogeneous workloads.

It could be also observed in the experimental analysis, that probabilistic decision strategies that try to predict the short-term future workload can balance better between energy consumption and performance impact.

For the presented reasons, we consider that the proposed simulator GAME-SCORE overperforms other simulators which only permit the application of static energy-aware scheduling algorithms and static energy-efficiency policies based on the shut-down of idle machines.

The presented model is just our first step towards the development of the new scheduling and resource allocation policies in order to optimize the energy utilization in the whole cloud distributing system . The model improvement plans include:

(a) exploration of more advanced energy policies;
(b) introduction of multiple players in order to play several games simultaneously without any central energy manager;
(c) examination of more scheduling models, such as two-level or shared-state models;
(d) test more complex and dynamic scheduling strategies;
(e) inclusion of VM/container migration and consolidation; and
(f) empirical comparison of the simulation results with real-life data.

.

Acknowledgements

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) and by the VPPI - University of Sevilla.

References

[1] O.A. Abdul-Rahman, K. Aida, Towards understanding the usage behavior of Google cloud users: the mice and elephants phenomenon, IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Singapore, (2014), pp. 272–277. doi:10.1109/CloudCom.2014.75 .
[2] D.G. Andersen, S. Swanson, Rethinking flash in the data center, IEEE micro 30 (4) (2010) 52–54.
[3] G.S. Aujla, M. Singh, N. Kumar, A. Zomaya, Stackelberg game for energy-aware resource allocation to sustain data centers using res, IEEE Transactions on Cloud Computing (2017).
[4] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future generation computer systems 28 (5) (2012) 755–768.
[5] A. Beloglazov, R. Buyya, Energy efficient resource management in virtualized cloud data centers, Proceedings of the 2010 10th IEEE/ACM international

conference on cluster, cloud and grid computing, IEEE Computer Society, 2010, pp. 826–831.
[6] 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, Concurrency and Computation: Practice and Experience 24 (13) (2012) 1397–1420.
[7] J.L. Berral, I.n. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, J. Torres, Towards energy-aware scheduling in data centers using machine learning, Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, e-Energy '10, ACM, New York, NY, USA, 2010, pp. 215–224, https://doi.org/10.1145/1791314.1791349.
[8] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and experience 41 (1) (2011) 23–50.
[9] M.C. Calzarossa, M.L. Della Vedova, L. Massari, D. Petcu, M.I. Tabash, D. Tessera, Workloads in the clouds, Principles of Performance and Reliability Modeling and Evaluation, Springer, 2016, pp. 525–550.
[10] 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 Transactions on Computer Systems (TOCS) 26 (2) (2008) 4.
[11] L. Cupertino, G. Da Costa, A. Oleksiak, W. Pia, J.-M. Pierson, J. Salom, L. Siso, P. Stolf, H. Sun, T. Zilio, et al., Energy-efficient, thermal-aware modeling and simulation of data centers: the coolemall approach and evaluation results, Ad Hoc Networks 25 (2015) 535–553.
[12] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.
[13] C. Delimitrou, C. Kozyrakis, Paragon: Qos-aware scheduling for heterogeneous datacenters, ACM SIGPLAN Notices, 48 ACM, 2013, pp. 77–88.
[14] C. Delimitrou, C. Kozyrakis, Quasar: resource-efficient and qos-aware cluster management, ACM SIGPLAN Notices, 49 ACM, 2014, pp. 127–144.
[15] C. Delimitrou, D. Sanchez, C. Kozyrakis, Tarcil: reconciling scheduling speed and quality in large shared clusters, Proceedings of the Sixth ACM Symposium on Cloud Computing, ACM, 2015, pp. 97–110.
[16] S. Di, D. Kondo, C. Franck, Characterizing cloud applications on a Google data center, 42nd International Conference on Parallel Processing (ICPP), Lyon, France, (2013).
[17] 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 Performance Evaluation Review 40 (1) (2012) 163–174.
[18] X. Fan, W.-D. Weber, L.A. Barroso, Power provisioning for a warehouse-sized computer, ACM SIGARCH Computer Architecture News, 35 ACM, 2007, pp. 13–23.
[19] M.E. Femal, V.W. Freeh, Boosting data center performance through non-uniform power allocation, Second International Conference on Autonomic Computing (ICAC'05), IEEE, 2005, pp. 250–261.
[20] D. Fernández-Cerero, A. Fernández-Montes, A. Jakóbik, J. Kołodziej, M. Toro, Score: Simulator for cloud optimization of resources and energy consumption, Simulation Modelling Practice and Theory 82 (2018) 160–173, https://doi.org/10.1016/j.simpat.2018.01.004.
[21] D. Fernández-Cerero, A. Fernández-Montes, J.A. Ortega, Energy policies for data-center monolithic schedulers, Expert Systems with Applications (2018), https://doi.org/10.1016/j.eswa.2018.06.007.
[22] D. Fernández-Cerero, A. Jakóbik, A. Fernández-Montes, J. Kołodziej, Stackelberg game-based models in energy-aware cloud scheduling, in: L. Nolle (Ed.), Proc. 32nd European Conference on Modelling and Simulation ECMS 2018 (ECMS, Wilhelmshaven, Germany, May 2018), ECMS '18, European Council for Modelling and Simulation, Dudweiler, Germany, 2018, pp. 460–467.
[23] D. Fernández-Cerero, A. Jakóbik, D. Grzonka, J. Koodziej, A. Fernández-Montes, Security supportive energy-aware scheduling and energy policies for cloud environments, Journal of Parallel and Distributed Computing 119 (2018) 191–202, https://doi.org/10.1016/j.jpdc.2018.04.015.
[24] 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 Recognition Letters 67 (2015) 59–65.
[25] S. Govindan, J. Liu, A. Kansal, A. Sivasubramaniam, Cuanta: quantifying effects of shared on-chip resource interference for consolidated virtual machines, Proceedings of the 2nd ACM Symposium on Cloud Computing, ACM, 2011, p. 22.
[26] 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 Generation Computer Systems (2017), https://doi.org/10.1016/j.future.2017.05.046. (in press)
[27] S.K.S. Gupta, R.R. Gilbert, A. Banerjee, Z. Abbasi, T. Mukherjee, G. Varsamopoulos, Gdcsim: A tool for analyzing green data center design and resource management techniques, 2011 International Green Computing Conference and Workshops, (2011), pp. 1–8, https://doi.org/10.1109/IGCC.2011.6008612.
[28] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, A. Goldberg, Quincy: fair scheduling for distributed computing clusters, Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, ACM, 2009, pp. 261–276.
[29] A. Jakóbik, D. Grzonka, J. Kołodziej, Security supportive energy aware scheduling and scaling for cloud environments (2017).
[30] A. Jakóbik, D. Grzonka, J. Kołodziej, Security supportive energy aware scheduling and scaling for cloud environments, European Conference on Modelling and Simulation, ECMS 2017, Budapest, Hungary, May 23-26, 2017, Proceedings. (2017), pp. 583–590, https://doi.org/10.7148/2017-0583.
[31] A. Jakóbik, D. Grzonka, J. Kolodziej, A.E. Chis, H. González-Vélez, Energy efficient scheduling methods for computational grids and clouds, Journal of Telecommunications and Information Technology (1) (2017) 56.
[32] A. Jakobik, D. Grzonka, F. Palmieri, Non-deterministic security driven meta scheduler for distributed cloud organizations, Simulation Modelling Practice and Theory. in press(available online 4 November 2016). doi:10.1016/j.simpat.2016.10.011.
[33] F. Juarez, J. Ejarque, R.M. Badia, Dynamic energy-aware scheduling for parallel task-based application in cloud computing, Future Generation Computer Systems (2016).
[34] D. Kliazovich, P. Bouvry, S.U. Khan, Greencloud: a packet-level simulator of energy-aware cloud computing data centers, The Journal of Supercomputing 62 (3) (2012) 1263–1283.
[35] J. Kołodziej, Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems, Springer Publishing Company, Incorporated, 2012.
[36] 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).
[37] Y.C. Lee, A.Y. Zomaya, Energy efficient utilization of resources in cloud computing systems, The Journal of Supercomputing 60 (2) (2012) 268–280.
[38] J. Mars, L. Tang, Whare-map: heterogeneity in homogeneous warehouse-scale computers, ACM SIGARCH Computer Architecture News, 41 ACM, 2013, pp. 619–630.
[39] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, R. Rajkumar, Critical power slope: understanding the runtime effects of frequency scaling, Proceedings of the 16th international conference on Supercomputing, ACM, 2002, pp. 35–44.
[40] R. Nallakumar, N. Sengottaiyan, K.S. Priya, A survey on scheduling and the attributes of task scheduling in the cloud, Int. J. Adv. Res. Comput. Commun. Eng 3 (10) (2014) 8167–8171.
[41] R. Nathuji, A. Kansal, A. Ghaffarkhah, Q-clouds: managing performance interference effects for qos-aware clouds, Proceedings of the 5th European conference on Computer systems, ACM, 2010, pp. 237–250.
[42] E. Niewiadomska-Szynkiewicz, A. Sikora, P. Arabas, J. Kołodziej, Control system for reducing energy consumption in backbone computer network, Concurrency and Computation: Practice and Experience 25 (12) (2013) 1738–1754.
[43] J.M. Pérez-Álvarez, M.T. Gómez-López, A.J. Varela-Vaca, F.F. de la Rosa Troyano, R.M. Gasca, Governance knowledge management and decision support using fuzzy governance maps, Business Process Management Workshops - BPM 2016 International Workshops, Rio de Janeiro, Brazil, September 19, 2016, Revised Papers, (2016), pp. 208–219, https://doi.org/10.1007/978-3-319-58457-7_16.
[44] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, J. Wilkes, Omega: flexible, scalable schedulers for large compute clusters, Proceedings of the 8th ACM European Conference on Computer Systems, ACM, 2013, pp. 351–364.
[45] 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 Computing 9 (1) (2005) 42–49.
[46] D. Shue, M.J. Freedman, A. Shaikh, Performance isolation and fairness for multi-tenant cloud storage. OSDI, 12 (2012), pp. 349–362.
[47] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed file system, 2010 IEEE 26th symposium on mass storage systems and technologies (MSST), IEEE, 2010, pp. 1–10.
[48] S. Sohrabi, A. Tang, I. Moser, A. Aleti, Adaptive virtual machine migration mechanism for energy efficiency, Proceedings of the 5th International Workshop on Green and Sustainable Software, ACM, 2016, pp. 8–14.
[49] T.T. Sá, R.N. Calheiros, D.G. Gomes, CloudReports: An Extensible Simulation Tool for Energy-Aware Cloud Computing Environments, Springer International

Publishing, Cham, 2014, pp. 127–142.
[50] J. Ullman, Np-complete scheduling problems, Journal of Computer and System Sciences 10 (3) (1975) 384–393, https://doi.org/10.1016/S0022-0000(75)80008-0.
[51] A. Varga, Discrete event simulation system, Proc. of the European Simulation Multiconference (ESM'2001), (2001).
[52] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, J. Wilkes, Large-scale cluster management at google with borg, Proceedings of the Tenth European Conference on Computer Systems, ACM, 2015, p. 18.
[53] B. Wickremasinghe, R.N. Calheiros, R. Buyya, Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications, 2010 24th IEEE International Conference on Advanced Information Networking and Applications, (2010), pp. 446–452, https://doi.org/10.1109/AINA.2010.32.
[54] A. Wilczyński, A. Jakóbik, Using Polymatrix Extensive Stackelberg Games in Security–Aware Resource Allocation and Task Scheduling in Computational Clouds, Journal of Telecommunications and Information Technology 1 (2017).
[55] H. Yang, A. Breslow, J. Mars, L. Tang, Bubble-flux: Precise online qos management for increased utilization in warehouse scale computers, ACM SIGARCH Computer Architecture News, 41 ACM, 2013, pp. 607–618.
[56] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, I. Stoica, Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling, Proceedings of the 5th European conference on Computer systems, ACM, 2010, pp. 265–278.
[57] X. Zhang, E. Tune, R. Hagmann, R. Jnagal, V. Gokhale, J. Wilkes, Cpi 2: Cpu performance isolation for shared compute clusters, Proceedings of the 8th ACM European Conference on Computer Systems, ACM, 2013, pp. 379–391.

DAMIÁN FERNÁNDEZ CERERO received the B.E. degree and the M.Tech. degrees in Software Engineering from the University of Sevilla. In 2014, he joined the Department of Computer Languages and Systems, University of Seville, as a PhD. Student. Currently he both teaches and conducts research at University of Sevilla. He has worked on several research projects supported by the Spanish government and the European Union. His research interests include energy efficiency and resource scheduling in computational clusters. His e-mail address is: damiancerero@us.es

AGNIESZKA JAKÓBIK (KROK) received her M.Sc. in the field of stochastic processes at the Jagiellonian University, Cracow, Poland and Ph.D. degree in the field of neural networks at Tadeusz Kosciuszko Cracow University of Technology, Poland, in 2003 and 2007. She is an Assistant Professor. Her e-mail address is: agneskrok@gmail.com

ALEJANDRO FERNÁNDEZ-MONTES GONZÁLEZ received the B.E. degree, M. Tech. and International Ph.D. degrees in Software Engineering from the University of Sevilla, Spain. In 2006, he joined the Department of Computer Languages and Systems, University of Sevilla, and in 2013 became Assistant Professor. His research interests include energy efficiency in distributed computing, applying prediction models to balance load and applying on-off policies to computational clusters. His e-mail address is: afdez@us.es

JOANNA KOŁODZIEJ is an associate professor in Research and Academic Computer Network (NASK) Institute and Department of Computer Science of Cracow University of Technology. She serves also as the President of the Polish Chapter of IEEE Computational Intelligence Society. Her e-mail address is: joanna.kolodziej68@gmail.com