Stackelberg Game-based Models in Energy-aware Cloud Scheduling

Damián Fernández-Cerero,
Alejandro Fernández-Montes,
Department of Computer Languages and Systems
University of Seville
Av. Reina Mercedes s/n, 41012 Seville, Spain

Agnieszka Jakóbik,
Cracow University of Technology
Warszawska st 24, 31-155 Cracow, Poland
Joanna Kołodziej
Research and Academic Computer Network
Kolska st 12, 01-045 Warsaw, Poland

KEYWORDS

Computational clouds; Cloud computing; Tasks scheduling; Energy saving; Cloud services modelling; Stackelberg Game

ABSTRACT

Energy-awareness remians the important problem in today's cloud computing (CC). Optimization of the energy consumed in cloud data centers and computing servers is usually related to the scheduling problems. It is very difficult to define an optimal scheduling policy without negoative influence into the system performance and task completion time. In this work, we define a general cloud scheduling model based on a Stackelberg game with the workload scheduler and energy-efficiency agent as the main players. In this game, the aim of the scheduler is the minimization of the makespan of the workload, which is achieved by the employ of a genetic scheduling algorithm that maps the workload tasks into the computational nodes. The energy-efficiency agent selects the energy-optimization techniques based on the idea of switchin-off of the idle machines, in response to the scheduler decisions. The efficiency of the proposed model has been tested using a SCORE cloud simmulator. Obtained results show that the proposed model performs better than static energy-optimization strategies, achieving a fair balance between low energy consumption and short queue times and makespan.

I. 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, HPC infrastructures 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 HPC systems are composed of thousands of computational distributed servers. The energy consumed by such HPC systems may be compared to the energy utilized by the small towns and large factories. Data centers account for more than 1.5% of global energy consumption [16].

Several hardware and infrastructure models have been recently developed for the successful reduction of the energy consumption in real-life large-scale data centers. The most popular models and technologies include: (i) cooling and temperature management [22] [4], (ii) memory and CPU power proportionality [19] [5], (ii) construction of the efficient new-generation green hard disks [1]; and new techniques in energy transportation [6]. 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 [20], 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 [8].

In this work, the balance between two opposed needs of the data-center environment is modelled by means of a Stackelberg Game (SG). On one hand, the performance side, represented by the *Scheduling Manager* (end users experience), 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 data center.

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 data-center scenarios.

The paper is organized as follows. In Section II we briefly describe the most usual and relevant strategies for achieving energy-efficiency in CC systems. In Section III, the shut-down decisions under consideration in this paper are presented. In Section IV, we formally define the proposed Stackelberg Game model for the balance between energy consumption and performance in CC systems, which is the theoretical core of this work. The used simulation tool and the experimentation performed, including the experimental environment and

Proceedings 32nd European Conference on Modelling and Simulation ©ECMS Lars Nolle, Alexandra Burger, Christoph Tholen, Jens Werner, Jens Wellhausen (Editors)
ISBN: 978-0-9932440-6-3/ ISBN: 978-0-9932440-7-0 (CD)

workload, as well as the results obtained are presented in Section V. Finally, the conclusions and future work are discussed in Section VI

II. Related work

One of the most popular method of saving the energy in the distributed computational environments is deactivation of the idle servers, since data centers in cloud environments usually operate at 20-30% of their capacity [2], [21] and the workload pressure and requirements suffer from fluctuations, such as those derived of day/night or weekdays/weekend patterns. Such servers are switched into the 'sleeping' mode in the idle periods or simply switched off. This strategy was used as the main energy conservation method in data center clouds and was fundamental for the other methodologies and models used in cloud scheduling and data and tasks processing. We present below a very simple state-of-the-art analysis of the main trends and achievements in the domain of energy-awareness in the data center clouds:

Several energy-aware scheduling policies have been developed to raise server utilization [13], [17]. These policies are useful to minimize the number of machines that process workload rather than spreading the tasks among the maximum number of available servers. This enables the application of several energy-efficiency approaches, such as DVFS and hibernating nodes in an idle state. However, these strategies are static and can not easily address drastic workload pattern changes without having a negative impact on the performance of the CC system.

Consolidation and migration of virtual machines (VMs) in the cloud clusters are other well explored models of energy conservation in the cloud computational environments [23], [3]. In this work we adopt a different strategy in order to achieve efficiency: the modeling of the data-center environment as a Stackelberg Game. However, these models may be incorporated as a part of the Stackelberg Game in a future work.

Other authors [14], [18] propose the application of various energy-efficient techniques in cloud computing subsystems, such as the distributed file system, and other paradigms such as Grid Computing [9], in order to improve cluster power proportionality. However, these approaches focus usually on only one side of the whole CC system, which makes them sub-optimal when a complex cloud-computing operation process is under consideration.

The major contribution of this work is a model for the dynamic application of energy-efficiency policies based on the Stackelberg Game model. We model the opposed requirements of any energy-efficient data center, 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.

III. "Switch-off" 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 cloud network. 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 compute 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 data centers) and the *Always* shut-down scenario, where all idle machines are switched-off.
- **Gamma** – in this case, the *Always* shut-down strategy is activated depending on the probability of incoming tasks 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 [11] and ETC Matrix scheduling model described in [10]. The makespan constitutes the most representative parameter of the performance, and hence it becomes the scheduling goal.

IV. Stackelberg Game model

In the model presented in this work we used the Stackelberg Game framework for the optimization of the balance between two main and opposed components of the model: *Scheduling Manager* – Leader and Energy-efficiency Manager –Follower.

Let us define first a 2-players non-zero symmetric game Γ_n as follows:

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

where:

- $N = \{1, \ldots, 2\}$ is the set of players,

- $\{S_1, \ldots, S_2\}$ $(cardS_i \geq 2; i = 1, \ldots, 2)$ is the set of strategies for them
- $\{H_1, \ldots, H_n\}; H_i : S_1 \times \cdots \times S_2 \to \mathbb{R}; \forall_{\beth=\nVdash,\ldots,\ltimes}$ is the set of payoff functions for each player players.

Each player in this game may make its own decisions. A single decision is one from the set of possible actions. In this game, the strategy is defined by the set of actions that the player considers beneficial for him. Both pure strategies and mixed strategies are considered in our model, see [24]. A pure strategy specifies the most beneficial actions for a given situation, thus, pure strategies are deterministic. Mixed strategies extend pure strategies by the assignation of a probability to each pure strategy. The usage of a mixed strategy enables a player to randomly select a single pure strategy from the set of available strategies. 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 may be defined as follows:

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

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

Randomization in the game is provided by the probability distribution $\sigma_i(s_i)$.

The result of following a given strategy is the **expected payoff** of the player i in the 2-players game. Let this pay-off function be defined as:

$$H_i(s_i, \sigma_{-i}) := \sum_{s_{-i} \in S-i} \sigma_{-i}(s_{-i}) H_i(s_i, s_{-i}) \tag{3}$$

It is assumed in that game, that player i plays the pure strategy $s_i \in S_i$ and his opponents plays the mixed strategy $\sigma_{-i} \in \Delta S_{-i}$.

The **expected payoff** of the player i when playing the mixed strategy $\sigma_i \in \Delta S_i$ and when his enemy plays the mixed strategy $\sigma_{-i} \in \Delta S_{-i}$ is defined as:

$$H_i(\sigma_i, \sigma_{-i}) = \sum_{s_i \in S_i} \sigma_i(s_i) H_i(s_i, \sigma_{-i}) \tag{4}$$

$$= \sum_{s_i \in S_i} \Big(\sum_{s_{-i} \in S_{-i}} \sigma_i(s_i) \sigma_{-i}(s_{-i}) H_i(s_i, s_{-i}) \Big) \tag{5}$$

In Stakelberg Games (SG), one player (the leader) may play first, and the rest of the players (the followers) are obliged to follow the leader and make their decisions after him [24]. The main objective of the game for each player is to maximize his expected payoff by finding and playing the optimal strategy.

During the game proposed in this paper the leader and the follower evolve their strategies alternately. Thus, each player reacts to the decisions made by the opponent. We assumed only two players in the game, therefore $i = 1$ or 2 and $-i = 1$ or 2.

In our model, we define independently the utility functions of both players. It is modeled as a non-zero sum game and allow us to generate a separate game model for each player.

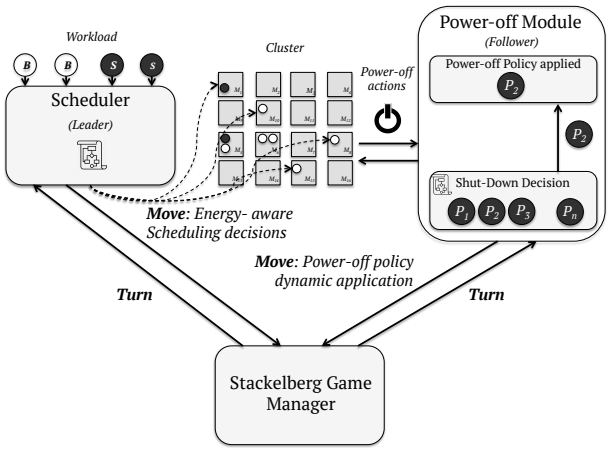

Fig. 1: Stackelberg Game workflow, scheduling workflow, B - Batch type task, S - Service type task, M - Virtual Machine

TABLE I: Players in the proposed game, their roles in the cloud and in the game model

Player 1	**Player 2**
Scheduling Unit	Energy-Efficiency Manager
Choses one of several schedules	Choses one of several energy policies
Leader	**Follower**

A. Leader Payoff and decisions

The leader in the proposed model is the *Scheduler* component, which perform the scheduling logic and dispatches tasks among the *Computing Nodes*. These *Computing Nodes* are grouped into *Computational Units*, denoted as $CU_1, CU_2, ...CU_P$. The *Scheduler* is responsible for processing incoming *Jobs*, which are composed of a set of *Tasks*, and for deploying these *Tasks* on the available *Computing Nodes* of a given *Computational Unit*.

Therefore there are P possible decisions. The strategy vector $\sigma_i(s_i)$ represents the probability for a *Job* to be assigned to the CU_p, for $p = 1, 2, ..., P$. The s_i may be taken from the set $1, 2, ..., P$.

The expected payoff of the *Scheduler* in the Stackelberg Game depends on the completion time of all the *Tasks* in the scheduled *Job*, thus, the makespan of that *Job*. In order to minimize the makespan, we employed a Monolithic Scheduler [15] which makes scheduling decisions based on the Expected Time to Compute (ETC) matrix, defined as follows:

$$ETC = [ETC[j][i]]_{j=1,\ldots,n}^{i=1,\ldots,m^p} \tag{6}$$

where

$$ETC[j][i] = wl_j / cc_i^p \tag{7}$$

In this equation, cc_i^p denotes 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 [12]. The main goal of the scheduling strategy is the minimization of the *Job* makespan:

$$C_{\max}(wl_1, ..., wl_n, cc_1^p, ..., cc_{m^p}^p, m^p, n, p) = \quad (8)$$

$$= \min_{S \in Schedules} \Big\{ \max_{j \in Tasks} C_j \Big\}, \quad (9)$$

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 achieved by the means of the Expected Time to Compute (ETC) matrix. In this matrix, the cell corresponding to the ith row and the jth column shows the completion time of the jth task if deployed on the ith CN. The lower values are to be considered, since the higher the value, the longer the makespan of the *Job*. Once the optimal schedule is computed, the obtained makespan value for that *Job* is taken as the utility function value for the game leader:

$$H_1(\sigma_1, \sigma_2) = \sum_{p=1,...,P} \sum_{l=1,..,L} \sigma_1^p \sigma_2^l C_{\max}(wl_1, ..., wl_n, cc_1^p, ..., cc_{m^p}^p, m^p, n, p) \quad (10)$$

$$= C_{\max} = \min_{S \in Schedules} \Big\{ \max_{j \in Tasks} C_j \Big\}, \quad (11)$$

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 second player. Therefore, we may rewrite the eq. (5) in the form of:

$$H_1(\sigma_1, \sigma_2) = \sum_{p=1,...,P} \sum_{l=1,..,L} \sigma_1^p \sigma_2^l C_{\max}(wl_1, ..., wl_n, cc_1^p(l), ..., cc_{m^p}^p(l), m^p(l), n, p(l)) \quad (12)$$

Thus, each player decisions infers the decision of the other player.

B. Follower Payoff and decisions

The follower in the game is the *Central Energy-efficiency Manager*, which applies energy policies to all the CNs in the data center. Those policies may change the availability of the CNs, which directly impact on their computational capacity. Therefore, the follower may decide about the $cc_i^p(l)$ and the $m^p(l)$. Those decisions will influence not only the follower's utility value, but also the behavior of the leader. The payoff for the follower player is the energy consumed by the CC system for the execution of the scheduled computed by the *Scheduler* and can be defined by the following equation:

$$H_2(\sigma_1, \sigma_2) = \sum_{i=1,...,m} \sum_{j=1,..,n} \sigma_1^j \sigma_2^i E(wl_1, ..., wl_n, cc_1^p, ..., cc_{m^p}^p, m^p, n, p, schedule) \quad (13)$$

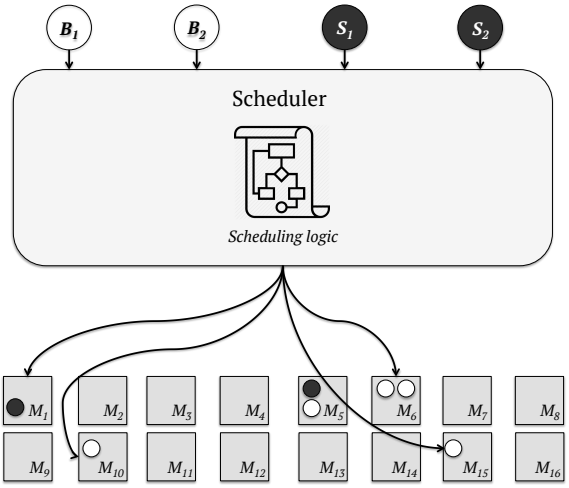

Fig. 2: Leader payoff computing, B - Batch type task, S - Service type task, M - Computing Node

The above equation shows that the payoff of the follower player depends on both the workload of the *Job* under consideration, and the schedule resulting of the leader player move. This means that every decision made by the leader player influences the next decision of the follower player. This scenario is similar to a chess play. Thus, each move of a given player changes the game environment and enforces the other player to make a decision.

In order to calculate the energy consumed by the CC during computing the given tasks according the the chosen schedule, the following values were introduced:

- E_{total} - the total energy spent by particular *Job*
- t_{idle}^i - c;
- t_{busy}^i - the time that the i-th CN spends on computing tasks;
- P_{idle}^i - the power a CN requires to operate in a idle state;
- P_{busy}^i - the power a CN requires to compute tasks

Therefore, we may express the time that the i-th CN spends on computing tasks by calculating:

$$t_{busy}^i = max_{j \in Tasks\ scheduled\ \ for\ \ CN_i} C_j \quad (14)$$

and the time that the i-th CN spends on computing tasks as follows:

$$t_{idle}^i = C_{max} - t_{busy}^i \quad (15)$$

The total energy consumption may be expressed in

the following way:

$$E_{total} = \sum_{i=1}^{m} \int_{0}^{C_{max}} Pow_{CN_i}(t)dt = \sum_{i=1}^{m} (P^i_{idle} * t^i_{idle} + P^i_{busy} * t^i_{busy} \quad (16)$$

These utility functions model the competition between two strategies that are usually contradictory: that of the *Scheduler*, which tries to compute tasks as fast as possible and that of the *Central Energy-efficiency Manager*, which tries to apply the more optimal power states to the CNs in order to maximize the energy efficiency.

V. Experimental analysis of the Stackelberg Game model

A. Simulation tool

The analysis of the described energy-efficiency strategies in real-life large-scale data centers is not feasible in such an immature stage. To overcome this limitation, in this work we chose a simulation tool designed to trustfully simulate energy-aware large-scale data centers called SCORE [7], which provides us with the means to reproduce realistic heterogeneous workloads and to easily implement various energy-efficiency policies.

In this paper, we extended the SCORE simulation tool in order to implement the Stackelberg Game process, which dynamically switches between energy-efficiency policies. The resulting architecture is shown in Figure 3.

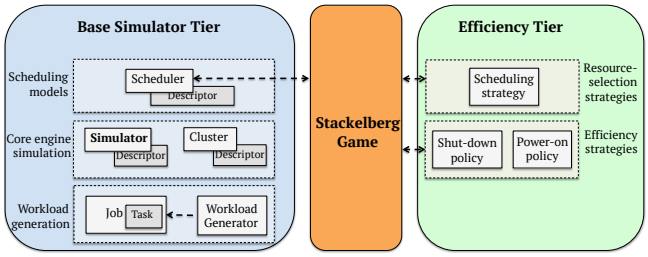

Fig. 3: Simulation tool architecture

B. 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.

We used the SCORE simulator [7] to perform a simple experiment that simulates seven days of operation time of a data center composed of 1,000 heterogeneous machines of 4 CPU cores and 8GB RAM and one central monolithic scheduler. In this experiment, we chose an heterogeneous day-night patterned mixed workload. This workload uses 30% of the data center computational resources on average, with peak loads that achieve 60% of utilization. This heterogeneous workload is composed of the following kind of jobs:

- **Batch jobs** perform a given amount of computational work and then are completed. Thus, this kind of job has a given start and end. In this experiment, *Batch* jobs are composed of 50 homogeneous tasks which consume 0.3 CPU cores and 0.5 GB of memory, and last for 90 seconds on average.
- **Service jobs** represent long-running jobs which serve end users. Due to this, this kind of job has an undetermined finish time. In this experiment, *Service* jobs are composed of 9 homogeneous tasks which consume 0.5 CPU cores and 1.2 GB of memory, and last for 2,000 seconds on average.

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 II and III.

This experimentation shows that the Stackelberg approach applies almost no negative impact in terms of queue times compared to the *Always* strategy at the cost of approximately 9% of more energy consumption. On the other hand, the *Always* strategy achieves an approximately 10% lower final average makespan time while achieving approximately 20% higher queue times.

C. Extended example in SCORE simulator

In this section, we extended the simple experimentation presented in Section V-B. In order to keep results comparable, we reused all the configuration parameters taken for the large-scale CC system shown in Section the V-B. However, in this experiment the *Central Energy-efficiency Manager* switches dynamically between the *Never* and the *Always* power-off policies by applying every *Shut-down decision policy* described in Section III. The results obtained are shown in Table IV and V.

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 approach depends directly on the *Shut-*

TABLE II: Energy-efficiency results for the simple Stackelberg experiment

Strategy	MWh consumed	MWh saved	Savings (%)	# shut downs	kWh saved per shut-down	Idle resources (%)
Never	58.53	0	0	0	N/A	69.94
Always	30.95	27.82	47.34	16,071	1.7314	3.52
Stackelberg	**36.07**	**23.00**	**38.94**	**742**	**31.00**	**13.96**

TABLE III: Performance results for the simple Stackelberg experiment

Strategy	Workload	Queue time until all tasks scheduled (ms)	Queue time until first task scheduled (ms)	Scheduler busy time (h)	Final makespan avg. (s)	Epoch 0 makespan avg. (s)
Never	Batch	74.11	74.11	9.28	136.16	175.46
Never	Service	73.72	73.72	0.66	N/A	N/A
Always	Batch	125.27	88.26	10.52	143.65	184.33
Always	Service	126.12	88.99	0.70	N/A	N/A
Stackelberg	**Batch**	**75.14**	**74.32**	**9.30**	**162.08**	**178.09**
Stackelberg	Service	78.31	74.40	0.66	N/A	N/A

down decision policy. More conservative probabilistic approaches, such as Gamma, achieve 26% faster queue times than a *Random* approach by consuming approximately 7% more energy. On the other hand, strategies that rely on leaving a security margin of free resources, such as *Margin*, could achieve approximately 22% faster queue times than a *Random* approach, and it would only consume less than 5% more energy. It can be noticed that conservative strategies such as Gamma apply almost no stress to the hardware, performing less than 1,000 shut-downs in a week of operation time, which represents a 10% of those performed by the *Random* decision policy.

VI. SUMMARY

In this paper, we presented a 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 model is 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.

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; and d) testing more complex and dynamic scheduling strategies.

ACKNOWLEDGEMENT

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] D. G. Andersen and S. Swanson. Rethinking flash in the data center. *IEEE micro*, 30(4):52–54, 2010.

[2] 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. *Communications of the ACM*, 53(4):50–58, 2010.

[3] A. Beloglazov and 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):1397–1420, 2012.

[4] N. El-Sayed, I. A. Stefanovici, G. Amvrosiadis, A. A. Hwang, and B. Schroeder. Temperature management in data centers: why some (might) like it hot. *ACM SIGMETRICS Performance Evaluation Review*, 40(1):163–174, 2012.

[5] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ACM SIGARCH Computer Architecture News*, volume 35, pages 13–23. ACM, 2007.

[6] M. E. Femal and V. W. Freeh. Boosting data center performance through non-uniform power allocation. In *Second International Conference on Autonomic Computing (ICAC'05)*, pages 250–261. IEEE, 2005.

TABLE IV: Energy-efficiency results for the extended Stackelberg experiment

Strategy	Switch Policy	MWh consumed	MWh saved	Savings (%)	# shut downs	kWh saved per shut-down	Idle resources (%)
Never	N/A	57.18	0.00	0	0	N/A	69.90
Always	N/A	30.54	26.74	46.68	13,680	1.95	3.87
Stackelberg	Margin	33.59	23.80	41.47	1,331	17.89	10.90
Stackelberg	Random	30.80	26.47	46.21	9,763	2.71	4.57
Stackelberg	Gamma	35.18	22.36	38.85	959	23.31	14.31

TABLE V: Performance results for the extended Stackelberg experiment

Strategy	Workload	Switch Decision Policy	Queue time until all tasks scheduled (ms)	Queue time until first task scheduled (ms)	Scheduler busy time (h)	Final makespan avg. (s)	Epoch 0 makespan avg. (s)
Never	Batch	N/A	71.07	71.07	9.16	139.66	177.25
Never	Service	N/A	73.89	73.89	0.66	N/A	N/A
Always	Batch	N/A	121.02	84.61	10.24	141.98	184.97
Always	Service	N/A	111.06	85.44	0.70	N/A	N/A
Stackelberg	Batch	Margin	78.33	72.61	9.24	159.98	179.74
Stackelberg	Service	Margin	77.92	75.88	0.66	N/A	N/A
Stackelberg	Batch	Random	100.79	79.58	9.88	141.95	183.85
Stackelberg	Service	Random	95.40	82.05	0.69	N/A	N/A
Stackelberg	Batch	Gamma	74.20	71.76	9.20	163.81	179.36
Stackelberg	Service	Gamma	78.47	75.69	0.66	N/A	N/A

[7] D. Fernández-Cerero, A. Fernández-Montes, A. Jakóbik, J. Kołodziej, and M. Toro. Score: Simulator for cloud optimization of resources and energy consumption. *Simulation Modelling Practice and Theory*, 82:160–173, 2018.

[8] A. Fernández-Montes, D. Fernández-Cerero, L. González-Abril, J. A. Álvarez-García, and J. A. Ortega. Energy wasting at internet data centers due to fear. *Pattern Recognition Letters*, 67:59–65, 2015.

[9] A. Fernández-Montes, L. Gonzalez-Abril, J. A. Ortega, and L. Lefèvre. Smart scheduling for saving energy in grid computing. *Expert Systems with Applications*, 39(10):9443–9450, 2012.

[10] A. Jakóbik, D. Grzonka, and J. Kołodziej. Security supportive energy aware scheduling and scaling for cloud environments. 2017.

[11] A. Jakóbik, D. Grzonka, J. Kolodziej, A. E. Chis, and H. González-Vélez. Energy efficient scheduling methods for computational grids and clouds. *Journal of Telecommunications and Information Technology*, (1):56, 2017.

[12] A. Jakobik, D. Grzonka, and F. Palmieri. Non-deterministic security driven meta scheduler for distributed cloud organizations. *Simulation Modelling Practice and Theory*, in press. (available online 4 November 2016).

[13] F. Juarez, J. Ejarque, and R. M. Badia. Dynamic energy-aware scheduling for parallel task-based application in cloud computing. *Future Generation Computer Systems*, 2016.

[14] R. T. Kaushik and M. Bhandarkar. Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster. In *Proceedings of the USENIX annual technical conference*, page 109, 2010.

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

[16] 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.

[17] Y. C. Lee and A. Y. Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280, 2012.

[18] X. Luo, Y. Wang, Z. Zhang, and H. Wang. Superset: a non-uniform replica placement strategy towards high-performance and cost-effective distributed storage service. In *Advanced Cloud and Big Data (CBD), 2013 International Conference on*, pages 139–146. IEEE, 2013.

[19] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar. Critical power slope: understanding the runtime effects of frequency scaling. In *Proceedings of the 16th international conference on Supercomputing*, pages 35–44. ACM, 2002.

[20] E. Niewiadomska-Szynkiewicz, A. Sikora, P. Arabas, and J. Kołodziej. Control system for reducing energy consumption in backbone computer network. *Concurrency and Computation: Practice and Experience*, 25(12):1738–1754, 2013.

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

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

[23] S. Sohrabi, A. Tang, I. Moser, and A. Aleti. Adaptive virtual machine migration mechanism for energy efficiency. In *Proceedings of the 5th International Workshop on Green and Sustainable Software*, pages 8–14. ACM, 2016.

[24] A. Wilczyński and 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.

AUTHOR BIOGRAPHIES

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

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

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 data centers. His e-mail address is: damiancerero@us.es

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 Data Centers. His e-mail address is: afdez@us.es