Energy-Aware Online Scheduling: Ensuring Quality of Service for IaaS Clouds

Andrei Tchernykh, Luz Lozano
CICESE Research Center, Ensenada, Mexico
{chernykh, llozano}@cicese.mx

Pascal Bouvry, Johnatan E. Pecero
University of Luxembourg, Luxembourg
{Pascal.Bouvry, Johnatan.Pecero}@uni.lu

Uwe Schwiegelshohn
TU Dortmund University, Dortmund, Germany
uwe.schwiegelshohn@udo.edu

Sergio Nesmachnow
Universidad de la República, Uruguay
sergion@fing.edu.uy

Abstract— **This work focuses on the analysis of energy-efficient algorithms for online scheduling in the Infrastructure as a Service (IaaS) type of Cloud computing. The scheduling model is based on service levels that guarantee the quality of service. Each service level is associated to a price per unit of job execution time and a slack factor, which determines the deadline for delivering the results. Once a batch job is submitted to the system, the provider has to decide whether the arriving job can be accepted or must be rejected and processed on external resources. The system maintains the quality of service guarantees for all already accepted jobs. After proposing an original formulation of the problem, we study various schedulers that provide good compromise between income maximization and power consumption minimization. Our experiments and analysis have shown that we can generate schedules with power consumption reduction without degrading the service quality.**

Keywords- Cloud computing, Service Level Agreement, Energy Efficiency, Scheduling.

I. INTRODUCTION

Scheduling is based on the mapping of a set of tasks to a set of resources. Nowadays, the scheduling algorithms used in the clouds are not quite efficient. Most algorithms focus on deterministic environments assuming knowledge of the user tasks and the system. Therefore, providers are always searching how to improve resource management to ensure Quality of Service (QoS). In addition, power consumption has become a major concern in distributed computing infrastructures [1, 2].

The shifting emphasis of grids and clouds towards a service-oriented paradigm has led to the adoption of SLAs as a very important concept. Therefore, the problem of finding the most stringent SLAs has emerged. The use of SLAs is a fundamentally new approach for job scheduling. With this approach, schedulers are based on satisfying QoS constraints. To accommodate a wide range of customers, different levels of service are used, each one addresses a different set of customers and establishes bilateral agreements between a service provider and consumers to guarantee job delivery time depending on the service level selected [3,4].

In a typical Infrastructure as a Service (IaaS) scenario, an infrastructure provider offers his resources on demand and with different service levels to his customers. Based on the models in hard real-time scheduling, we introduce in this work a simple model for service level based job allocation and scheduling, taking into account power consumption on the distributed system. These service levels are mainly distinguished by the amount of computing power a customer is guaranteed to receive within a time frame. In our model, each service level is described by a slack factor and a price for a processing time unit. If the provider accepts a job it is guaranteed to complete by its deadline, that is its submission time plus its processing time times the slack factor of assigned service level. After a job has been submitted, the provider must decide immediately and irrevocably whether he accepts or rejects the job. After acceptance, a low-level scheduler is applied to optimize the power consumption on the system. We suggest various algorithms and use competitive analysis to discuss different scenarios for this model.

In [4], Schwiegelshohn and Tchernykh analyze the single (SM) and the parallel machine (PM) models subject to jobs with single (SSL) and multiple service levels (MSL). The analysis uses theoretical means and is based on the competitive factor, which is the ratio between the income of the infrastructure provider obtained by the scheduling algorithm and the optimal income in the worst case. The paper provides worst case performance bounds for four greedy acceptance algorithms named SSL-SM, SSL-PM, MSL-SM, MSL-PM, and two restricted acceptance algorithms MSL-SM-R, and MSL-PM-R. All of them are based on the adaptation of the preemptive Earliest Due Date (EDD) algorithm for scheduling the jobs with deadlines.

To show the practicability and competitiveness of the algorithms, Lezama et al. [3] conduct a study of their performance and derivatives using simulation. The authors use workloads based on real production traces of heterogeneous HPC systems to increase the practical relevance of the results. Particularly, they demonstrate the benefit of parallelism by showing that they can achieve better competitive factor in a PM scenario than in the corresponding SM scenario.

The paper is structured as follows. The next section reviews the main related works about SLA and energy-aware scheduling on distributed infrastructures. Section III presents the problem definition and Section IV introduces the metrics used to evaluate solutions. The proposed schedulers are described in Section V. The experimental analysis is reported in Section VI, including an exhaustive analysis of the

978-1-4799-5313-4/14/$31.00 ©2014 IEEE

numerical results for the proposed schedulers when solving a benchmark set of different problem instances and scheduling scenarios. Finally, Section 6 summarizes the main conclusions of the research and proposes some lines for future work.

II. RELATED WORK

There has been significant research on the topic related to SLA into the Cloud architecture [5]; usage of SLAs for resource management and admission control techniques [3, 6]; SLA profits; automatic negotiation protocols [7]; economic aspects associated with the usage of SLAs for service provision and the incorporation of the SLA into the Cloud architecture [3], but little is known about efficiency of SLA scheduling solutions [8]. There are few theoretical results on SLA scheduling, and most of them address real time scheduling with given deadlines. A more complete study is presented by Schwiegelshohn and Tchernykh [4]. We take their algorithm MSL-PM as the base algorithm for our experimental study.

Lezama *et al.* [3] performed an experimental study of the algorithms proposed in [4]. Their case study demonstrates that the rate of rejected jobs, the number of job interruptions, and the competitive factor strongly depends on the slack factor, which can be dynamically adjusted in response to changes in the system configuration and/or workload, to provide increased revenues and performance.

There are several research efforts on resource allocation to optimize power consumption. Three main strategies are used in the literature: i) dynamic component deactivation with switching off parts of the computer system that are not utilized; ii) Dynamic Voltage (and Frequency) Scaling (DVS/DVFS) to slow down the speed of CPU processing, and iii) explicit approaches using the Max-Min mode, which accounts for hardware-embedded energy saving features (e.g. SpeedStep by Intel, or Optimized Power Management by AMD). All these policies are designed to reduce the power consumption of one resource, but not across resources distributed geographically. Moreover, they work under the assumption that exact resource information is available.

The benefits of the first approach for distributed systems (grids/clouds) were explored by Tchernykh *et al.* in [9]. The considered policy turns off/on servers so that only the minimum number of servers required to support the QoS for a given workload are kept active. Raycroft *et al.* [10] analyzed the effects of virtual machine allocation on power consumption. Tests in a realistic scenario show that using an allocation policy designed to minimize energy the total power consumption could be reduced by up to 14%. Furthermore, they assessed the performance qualification of the energy/cost driven allocation policies through network capability tests.

DVS/DVFS energy-aware approaches have been common in literature, from early works like the one by Khan and Ahmad [11] using a cooperative game theory to schedule independent jobs on a DVS-enabled grid to minimize makespan and energy. Lee and Zomaya [12] studied a set of DVS-based heuristics to minimize the weighted sum of makespan and energy. Later these results were improved by Mezmaz *et al.* [13] by proposing a parallel bi-objective hybrid GA. Pecero *et al.* [14] studied two-phase bi-objective algorithms using a Greedy Randomized Adaptive Search Procedure (GRASP) that applies a DVS-aware bi-objective local search to generate a set of Pareto solutions. Pinel *et al.* [15] used a double minimization approach and a local search to solve the problem. Lindberg *et al.* [16] proposed six greedy algorithms and two GAs for solving the makespan-energy scheduling problem subject to deadline constraint and memory requirements.

The explicit approach using the Max-Min mode was studied by Nesmachnow *et al*. in [17], where twenty fast list scheduling algorithms were applied to solve the bi-objective problem of optimizing makespan and power consumption. The experimental results demonstrated that accurate schedules with different makespan/energy trade-offs can be computed with the two-phase optimization model. Using the same approach, Iturriaga *et al*. [18] show that a parallel multi-objective local search based on Pareto dominance outperforms deterministic heuristics based on the traditional *MinMin* strategy.

Job scheduling is a crucial part of efficient cloud implementation. Diversified aspects of the problem are discussed to cope with the new challenges of multi-domain distributed systems: centralized, hierarchical and distributed models; static and dynamic scheduling policies; multi-objective optimization; adaptive policies; QoS and SLA constraints; economic models; scheduling of data and computational intensive applications; and energy efficiency among other topics. In this article, we address some of these topics (QoS, SLA, energy efficiency), and face a fundamental research problem: how to minimize power consumption without violation of QoS.

III. PROBLEM DEFINITION

This section presents the system model and the power consumption model.

A. System model

In this work, we are interested in providing QoS guarantees and optimizing both the provider income and power consumption.

Let $SL = [SL_1, SL_2, \dots, SL_l, \dots, SL_k]$ be a set of service levels offered by the provider. For a given service level SL_j^l of the job J_j, user is charged with cost u_j^l per execution time unit depending on the urgency of the submitted job. This urgency is denoted by a slack factor of the job $f_j^l \geq 1$. $u_{max} = \max_l\{u_j^l\}$ denotes the maximum cost for all $l = 1..k$ and $j = 1..n$. The total number of jobs submitted to the system is n_r.

Each job J_j is described by a tuple (r_j, p_j, d_j, SL_j^l) containing the release date r_j, the execution time p_j, the deadline d_j, and the service level $SL_j^l \in SL$ for the job. d_j is the latest time that the system would have to complete the job J_j in case it is accepted. This value is calculated at the release of the job as $d_j = r_j + f_j^l \cdot p_j$ (the maximum deadline is $d_{max} = max_j\{d_j\}$) When the job is submitted at time r_j, p_j becomes known. The profit that the system will obtain for the

execution of job J_j is calculated as $u_j^l \cdot p_j$. Once the job is released, the provider has to decide, before any other job arrives, whether the job is accepted or no. In order to accept the job J_j, the provider should ensure that some machine in the system is capable to complete it before its deadline. In the case of acceptance, further jobs cannot cause job J_j to miss its deadline. Once a job is accepted, the scheduler uses some rule to schedule the job. Finally, the set of accepted jobs $J = [J_1, J_2, \dots, J_n]$ is a subset of J_r and n is the number of jobs that are successfully accepted and executed.

We consider a set of m heterogeneous machines $M = [M_1, M_2, \dots, M_m]$. Each machine M_i is described by a tuple (s_i, eff_i) indicating the processing speed factor s_i and the energy efficiency eff_i. At time t, only a subset of all machines can accept a job. Let $M^a(t) = [M_1, M_2, \dots, M_{m^a}]$ be such a set of admissible machines, so that the job processing time on machine i is $p_j^i = p_j / s_i$. Each machine M_i has a queue storing all jobs allocated for execution. As in many schedules, C_{max} denotes the makespan of the schedule.

B. Energy model

In the energy model, we assume that the power consumption $P_i(t)$ of machine M_i at time t consists of a constant part P^{idle} that denotes a power consumed by machine M_i in idle state and a variable part P^{work} that depends on the workload: $P_i(t) = o_i(t) \cdot (P^{idle} + w_i(t) P^{work})$, where $o_i(t) = 1$, if the machine is *ON* at time t, otherwise $o_i(t) = 0$, and $w_i(t) = 1$, if the machine is full loaded, otherwise $w_i(t) = 0$. The total power consumed by the cloud system is the sum of power consumed while cloud operational state (E^{op}) defined as:

$$E^{op} = \int_{t=1}^{C_{max}} P^{op}(t), \text{ where } P^{op}(t) = \int_{i=1}^{m} P_i(t)$$

$$E^{op} = \int_{t=1}^{C_{max}} \int_{i=1}^{m} o_i(t) \cdot (P^{idle} + w_i(t) P^{work}),$$

$o_i(t) = 1$, if *on*; 0 if *off*. $w_i(t) = 1$ if *work*, 0 if *idle*.

IV. METRICS

In order to evaluate the system performance, we use a series of metrics that are useful for systems with SL, where traditional measures such as makespan become irrelevant.

For this kind of systems, the metrics must allow the provider to measure the performance of the system in terms of parameters that helps him to establish utility margins as well as user satisfaction for the service. We consider two main metrics: competitive factor and power consumption. Due to the definition of this system we have to assure the benefit for the service provider, so that the first main metric is the competitive factor ρ that measures the ratio in which the income generated by our algorithm gets closer to the value obtained by an optimal income $V(A)^*$. The competitive factor ρ is defined as: $\rho = \frac{\sum_{j=1}^{n} (u_j^l \cdot p_j)}{V(A)^*} \leq 1$, where the optimal income $V(A)^*$ is approximated by an upper bound $\hat{V}(A)^*$ obtained as follow: $V(A)^* \geq \hat{V}(A)^* = u_{max} \cdot min\left(\sum_{j=1}^{n_r} p_j, d_{max} \cdot m\right)$

A lower bound for the competitive factor $\check{\rho} \leq \rho$ is obtained by using $\hat{V}(A)^*$. The first term of $\hat{V}(A)^*$ is the sum of the processing times of all released jobs multiplied by the maximum price per unit execution of all available SLAs. The second bound is the maximum deadline of all released jobs multiplied by the maximum price per unit execution value and the number of machines in the system. Due to our admission control policy, the system does not execute jobs if their deadlines cannot be reached therefore this second bound is also an upper bound of the total processing time of the system.

V. SCHEDULING ALGORITHMS

This section describes the scheduling approach and the proposed energy-aware SLA scheduling methods.

A. Scheduling approach

A two-level scheduling approach is used, as shown in Fig. 1. On the upper level, the system verifies whether a job can be accepted or not using a *Greedy acceptance policy*. If the job is accepted then the system selects a machine from the set of admissible machines for executing the job on the lower level. The selected machine can be determined by taking into account different criteria. In this work, we study eight different allocation strategies, which are described in Section V.C and Table 1.

B. Higher level acceptance policies

We use greedy higher level acceptance policies MSL_G. It is based on the Earliest Due Date algorithm, which gives priority to jobs according to its deadline.

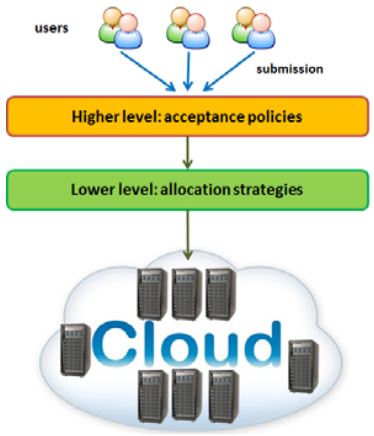

Figure 1. Two-level scheduling approach using acceptance policies (upper level) and allocation strategies (lower level)

When a job J_j arrives to the system, in order to determine whether to accept it or reject it, MSL_G searches for the set of machines capable of executing job J_j before its deadline, assuring that all the jobs in the machine queue will not miss its deadlines. If the set of available machines is not empty ($|M^a(r_j)| \geq 1$) job J_j is accepted, otherwise it is rejected. This completes the first stage of scheduling.

DasGupta and Palis [19] show that there cannot exist an algorithm with competitive ratio greater than $1 - 1/f_i + \varepsilon$, with $m \geq 1$ machines and $\varepsilon > 0$, if preemption is allowed. The authors propose an algorithm that achieves a competitive

ratio of at least $1 - 1/f_i$ and demonstrated that this is an optimal scheduler for hard real-time scheduling with m machines. They propose a simple admittance test based on EDD to verify that all already accepted jobs with a deadline greater than the deadline of the incoming job will be completed before their deadline is reached.

C. Lower level allocation strategies

Allocation strategies are characterized by the type and amount of information used for allocation decision. We distinguish two levels of available information. In *Level* 1, the job execution time, the speed of machines, and the acceptance policy are assumed to be known. In *Level* 2, in addition of the information of *Level* 1, the machine energy efficiency and the energy consumed by executing a job are assumed to be known.

Table 1 summarizes the main details of the allocation strategies used in this work. We categorize the proposed methods in three groups: i) *knowledge-free*, with no information about applications and resources; ii) *energy-aware*, with power consumption information; and iii) *speed-aware* with speed of machines information.

VI. Experimental setup

A theoretical competitive analysis could be applied to compare the considered strategies [4]. However, competitive factors are worst case factors that frequently do not occur in real systems. Thus, simulation is the only feasible way to evaluate large-scale distributed systems with heterogeneous resources. Simulation is effective when working with very large problems that involve a large number of resources and users. Simulation also makes it possible to explore different types of systems operating under varying workloads. This section presents the configuration for the experiments, including workload and scenarios, and describes the methodology used for the analysis.

A. Workloads

We evaluate the performance of our strategies with a series of experiments using traces of real HPC jobs obtained from the Parallel Workloads Archive [20], and the Grid Workload Archive [21]. These workloads are suitable for assessing the system because our IaaS model with multiple heterogeneous parallel machines is intended to execute jobs traditionally executed on Grids and parallel machines. Therefore, the performance evaluation under realistic workload conditions is essential [26]. The workloads include nine traces from: DAS2-University of Amsterdam, DAS2–Delft University of Technology, DAS2–Utrecht University, DAS2–Leiden University, KHT, DAS2–Vrije University Amsterdam, HPC2N, CTC, and LANL. The main details of the considered sites are reported on Table 2. Further details about the logs and workloads can be found in [20] and [21].

B. Scenarios

The problem scenarios considered in the experimental analysis have the following details: workload of seven days that includes batch jobs; the greedy acceptance policy MLS_G on the higher level; uniform machines; eight infrastructure sizes with the number of machines being powers of 2 from 1 to 128, four SL; and the eight lower level allocation heuristics described in Table 1.

The data on Table 2 show speed, energy efficiency and power consumption of the machines and their workloads. We see that the range of the speed is [18.6, 481] efficiency [0.89, 1.75], power consumption [17.8, 58.9] for P^{idle}, and [26, 66] for P^{work}.

C. Methodology used for the analysis

Two criteria are considered in the analysis: the income V and the power consumption E^{op}. The problem can be simplified to a single objective problem through different methods of objective weighted aggregation. There are various ways to model preferences, for instance, they can be given explicitly to specify the importance of every criterion or a relative importance between criteria. This can be done by a definition of criteria weights or criteria ranking by their importance. In order to provide effective guidance in choosing the best strategy, in this paper, we perform a joint analysis of two metrics according to the mean degradation methodology proposed in [22], and applied for scheduling in [23, 24].

*Degradation in performance***.** In this approach, the analysis assumes equal importance of each metric; hence, they can be averaged. The goal is to find a well performing strategy under all test caseswith the expectation that it will also perform well under other conditions, e.g., with different configurations and workloads.

The analysis is conducted as follows. First, we evaluate the degradation in performance (relative error) of each strategy under each metric. This is done relative to the best performing strategy for the metric, as follows:

$$(\gamma - 1) \cdot 100, \text{ with } \gamma = \frac{strategy\ metric\ value}{best\ found\ metric\ value}.$$

Then, we average these values, and rank the strategies. The best strategy with the lowest average performance degradation has rank 1. Note that we try to identify strategies, which perform reliably well in different scenarios; that is, we try to find a compromise that considers all of our test cases. For example, the rank of the strategy could not be the same for any of the metrics individually or for any of the scenarios individually. We present the metric degradation averages to evaluate performance of the strategies, and show if some strategies tend to dominate results. The degradation approach provides the mean percentage of degradation, but it does not show the negative effects of allowing a small portion of the problems with large deviation to dominate the conclusions based on averages.

To analyze those possible negative effects and to help with the interpretation of the data generated by the benchmarking process, we present performance profiles of our strategies.

Performance profile. The performance profile $\delta(\tau)$ is a non-decreasing, piecewise constant function that presents the probability that a ratio γ is within a factor τ of the best ratio [25]. The function $\delta(\tau)$ is the cumulative distribution function. Strategies with large probability $\delta(\tau)$ for smaller τ will be preferred.

TABLE I. MAIN DETAILS OF THE ALLOCATION STRATEGIES STUDIED IN THIS WORK

Type	Strategy	Level	Description
Knowledge-free	Rand	1	Allocates job j to a machine with the number random generated from a uniform distribution in range $[1..m]$.
	FFit	1	Allocates job j to the first machine available and capable to execute it.
	MLp	1	Allocates job j to the machine with the least load at time r_j: $\min_{i=1...m^a}\{n_i\}$,
Energy-aware	Max_eff	2	Allocates job j to the machine with higher energy efficiency $\max_{i=1...m^a}\{eff_i\}$..
	Min_e	2	Allocates job j to the machine with minimum total power consumption at time r_j : $\min_{i=1...m^a}\{\sum_{t=1}^{r_j} P_i^{op}(t)\}$
	MCT_eff	2	Allocates job j to the machine with the earliest completion time on an energy efficient machine $min\left\{\frac{c_{max}^i}{eff_i}\right\}$, where $c_{max}^i = \max_{g_k=i}\{c_k^i\}$ and c_k^i being the makespan and completion time of job k in the machine i, respectively
Speed-aware	Max_seff	2	Allocates job j to the maximum energy efficient faster machine: $\max_{i=1...m^a}\{s_i * eff_i\}$,
	Max_s	2	Allocates job j to the fastest machine: $\max_{i=1...m^a}\{s_i\}$

TABLE II. EXPERIMENTAL SETUP

Site	Procs	Power consumption W		Energy efficiency MFS/W	Speed GFLOPS	Log	#Jobs	#User
		P^{Idle}	$P^{Working}$					
DAS2—University of Amsterdam	64	17.8	35.35	1.36	126	Gwa-t-1-anon_jobs-reduced.swf	1124772	333
DAS2—Delft University of Technology	64	17.8	35.35	1.36	126			
DAS2—Utrecht University	64	17.8	35.35	1.36	126			
DAS2—Leiden University	64	17.8	35.35	1.36	126			
KTH—Swedish Royal Institute of Technology	100	17.8	26	1.75	18.6			
DAS2—Vrije University Amsterdam	144	17.8	35.35	1.32	230	KTH-SP2-1996-2.swf	28489	204
HPC2N—High Perf. Comp. Center North, Sweden	240	58.9	66	0.89	481	HPC2N-2002-1.1-cln.swf	527371	256
CTC—Cornell Theory Center	430	17.8	26	1.64	88.4	CTC-SP2-1996-2.1-cln.swf	79302	679
LANL—Los Alamos National Lab	1024	24.7	31	1.45	65.4	LANL-CM5-1994-3.1-cln.swf	201387	211

VII. EXPERIMENTAL RESULTS

For all scenarios, we define two parameters: the number of machines and number of SLs, which we use for experimental study. The number of SLs and slack factor values are set to 4, when up to four VM could share resources. Therefore, the set of SLs is $SL = [SL_1, \dots, SL_4]$ with slack factors $f_1 = 1, \dots\ f_2 = 2,\ f_3 = 3, f_4 = 4$.

A. Income, MSL-MM

In this section, we analyze the income obtained by the eight allocation strategies studied over the eight considered infrastructures, and jobs with 4 SLs. Fig. 2-4 show the average income per machine, total income and income degradation, respectively using four levels of service.

Fig. 2 shows that for the given workloads we get the best return on infrastructure per machine when 4 machines are used. After that the income generated by each machine is decreased. The average income generated by each machine for 4 machines scenario is about 3 times higher than on scenarios with 1, 64 and 128 machines.

Fig. 3 shows that the total income generated by each strategy is different when using 8, 16 and 32 machines. When 64 and 128 machines are used, strategies show similar results no matter what machine is used for allocation as, in these cases, all jobs are accepted providing optimal income.

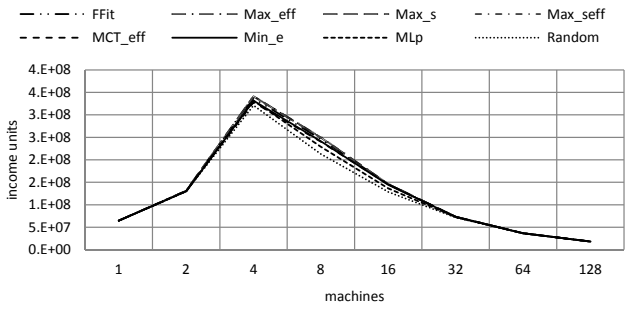

Figure 2. Average income per machine using SLA with 4 SLs

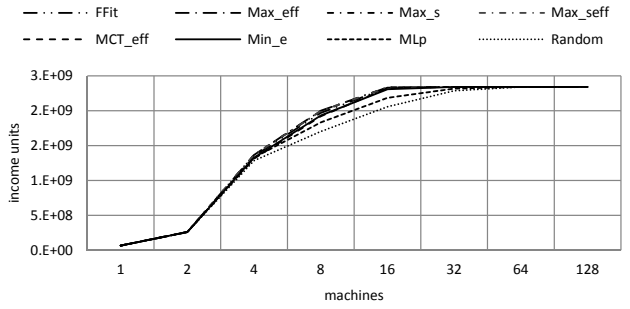

Figure 3. Total income using SLA with 4 SLs

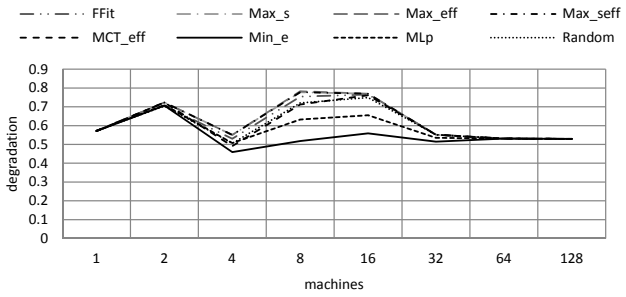

Figure 4. Income degradation using SLA with 4 SLs

For instance, in scenario with m = 16, *Max_eff* , *Max_s* and *Max_seff* generate more income, but the difference with *Min_e* is less that 1%.

Fig. 4 shows that *Min_e* strategy that allocates jobs to the machine with minimum total power consumption computes schedules with lower degradation for 8, 16 and 32 machines. We see that in scenarios with m = 1, 2, 64, and 128, the studied allocation strategies have negligible difference. The explanation for this is that in the case of m = 1, 2 there is only a small diversity of machines for the allocation. In the case of m = 64 and m = 128, almost all jobs are accepted regardless of the strategy we use because there are always available resources.

For instance, in the 16 machines scenario, we see a clear difference among the studied methods. The allocation to the machine with minimum total power consumption *Min_e* has the best behavior in all scenarios. The second best strategy is *MLp*, having 17% higher degradation. *Max_eff*, *Max_s*, and *Max_seff* have about 75% degradation comparing with *Min_e*. The difference between them is less than 1%.

B. Power consumption

In this section, we present the analysis of the power consumption.

Fig. 5 and 6 show the results obtained on the same scenarios presented in the previous section. Fig. 5 shows the total power consumption and Fig. 6 shows the degradation of the power consumption.

In Fig. 5, we see that the power consumption is an increasing function when increasing the number of machines. However, in the range from 32 to 128 machines, the consumption is almost the same for all studied heuristics. This means that all jobs are accepted and power consumption is not increased. The machines without workload are off and therefore they do not consume any power. The strategy that allocates jobs to the machine with lower total power consumption is about four times better than the strategy that assigns jobs to the fastest machine.

From our experiments, we see that as we increase the number of levels of service in the SLA, the power consumption is increased slightly and the degradation is a bit larger than in the scenarios with lower SLAs.

The results on Fig. 6 demonstrate that the degradation in the scenario with 128 machines is decreased with respect to 64 machines. With more machines, the strategies have more options for resource allocation, and, in overall, all strategies take advantage of this diversity. The *Min_e* strategy has the best behavior to minimize energy consumption.

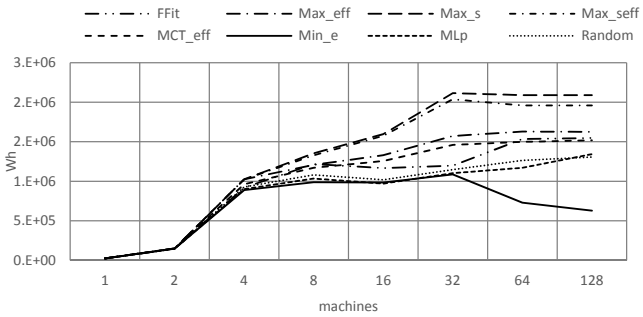

Figure 5. Total power consumption using SLA with 4 SLs

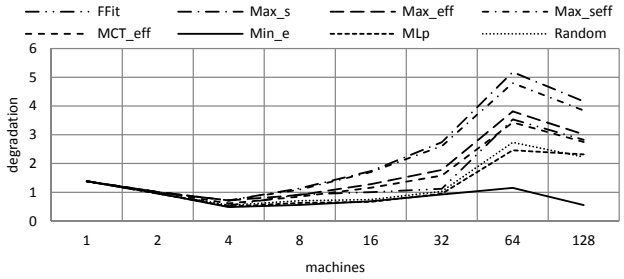

Figure 6. Degradation of the power consumption using SLA with 4 SLs

C. Bi-objective analysis

In previous sections, we presented the analysis of the income and power consumption separately. Now we are interested in finding the strategy that generates the best compromise between income and energy consumption. To perform this analysis, we use the technique of degradations and ranking, and performance profile described in Section VI.C.

Degradation in performance analysis. To obtain valid statistical values, 30 experiments of 7 days are simulated. Once we have the results of the testing of the 8 configurations with different number of machines, we calculate the degradation for each scenario and calculate their average. The results are reported in Table 3.

In the Table 3, for each set of experiments, we present the average degradation for SLA with four SF. The last three columns of the table contain the ranking of each strategy regarding to income, power and their mean. *Ranking-P* is based on the income degradation. *Ranking-E* refers to the position in relation to the degradation of power consumption. *Rank* is the position based on the averaging two degradations.

In Table 3, we see that the best strategy for resource allocation is to assigned jobs to the machine that consumed less energy up to the moment of allocation (*Min_e*). This leads us to better average income and lower power consumption.

TABLE 3. DEGRADATION AND RANKING

SLA	Strategy	Income	Energy	Mean	Ranking-P	Ranking-E	Ranking
SLA 4	FFit	0.59	1.52	1.06	5	4	4
	Max_s	0.60	2.32	1.46	7	6	6
	Max_eff	0.60	1.75	1.18	8	8	8
	Max_seff	0.60	2.20	1.40	6	7	7
	MCT_eff	0.59	1.61	1.10	3	5	5
	Random	0.59	1.30	0.94	4	3	3
	Min_e	0.54	0.84	0.69	1	1	1
	MLp	0.57	1.24	0.91	2	2	2

The good performance of this strategy is because it makes a load balancing between machines considering total power consumption. A machine can have lower power consumption due to various reasons. The first one is that machine could receive fewer loads than other ones; it has better energy efficiency, or both. All situations cause the load balancing and generate more income and less power consumption.

The second best strategy, MLp, assigns jobs to the machine having less allocated jobs. It also intends to balance load, but this balance is in relation to the assigned work.

The analysis shows that if we have no information about the speed of the machines or their energy efficiency, it is better to allocate jobs to the machine that has fewer assignments. If we have the information about speed and energy efficiency, the best option is assigning a job to the machine that has consumed less power at the time of the decision.

Performance profiles. As mentioned in Section VI.C, conclusions based on the averages may have some negative aspects. To analyze possible negative effects of allowing a small portion of the problem instances with large deviation to dominate the conclusions that based on averages, we present performance profiles of our strategies. Fig. 7 shows the performance profiles according to income in the interval $\tau = [1, ..., 2.7]$ to provide objective information for analysis of a test set. This figure displays the small discrepancies in the income on a substantial percentage of the problems. *Min_e* has the highest ranking and the highest probability of being the better strategy. The probability that it is the winner on a given problem within factors of 1.6 of the best solution is close to 0.7. If we choose being within a factor of 2 as the scope of our interest, then all strategies would suffice with probability 0.87.

Fig. 8 shows the performance profiles according to power consumption in the interval $\tau = [1, ..., 9]$. It displays the large discrepancies in the power consumption degradation on a substantial percentage of the problems. *Min_e* has the highest ranking and the highest probability of being the better strategy. If we choose being within a factor of 3 as the scope of our interest, the probability that it is the winner on a given problem is close to 1. Within a factor of 2 of the scope of our interest, it wins with probability 0.7.

Fig. 9 shows the performance profiles of the 8 strategies by averaging two metrics: energy and income. The most significant aspect of Fig. 9 is that on this test set *Min_e* dominates other strategies: its performance profile is never below any other for all values of performance ratios. *MLp* is the second best strategy.

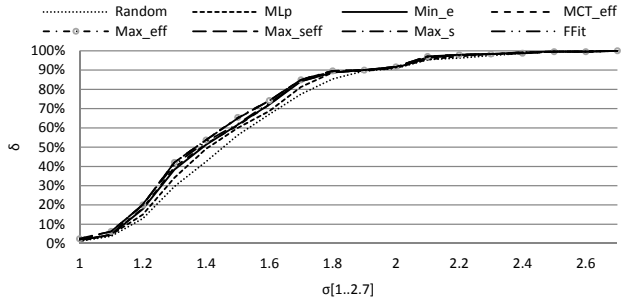

Figure 7. Performance profile of the income, 8 strategies

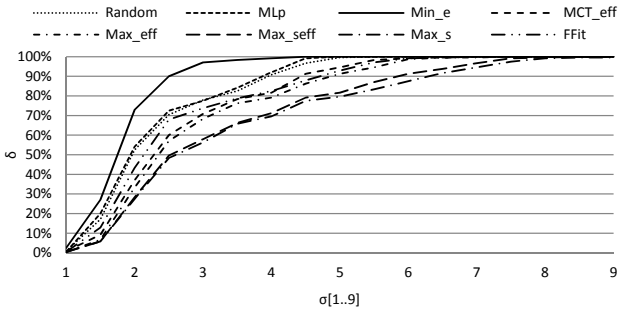

Figure 8. Performance profile of the energy consumption, 8 strategies.

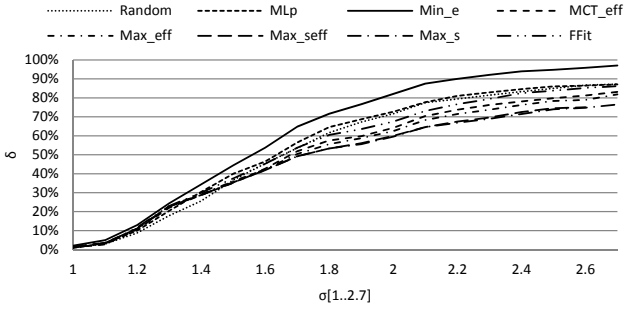

Figure 9. Performance profile of the power consumption and income average, 8 allocation strategies

VIII. CONCLUSIONS

In this paper, we analyze a variety of scheduling algorithms with different cloud configurations and workloads considering two objectives: provider income and power consumption.

A user submits jobs to the service provider, which offers four levels of service. For a given service level the user is charged by a cost per unit of execution time. In return, the customer receives guarantees regarding the provided resources. These guarantees are maximum response time or deadlines used as QoS constraints. Our experimental case study results in several contributions: (a) First, we identify the problem of the resource allocation with several service levels and quality of service to make scheduling decisions with respect to job acceptance and two criteria optimization; (b) We analyze scenarios with heterogeneous machines of different configurations and workloads; (c) We provide an experimental

study of greedy acceptance algorithms *MSL-G* with known worst case performance bound and 8 allocation strategies that take into account heterogeneity of the environment; (d) We distinguish allocation strategies depending on the type and amount of information they require: knowledge free, energy-aware, and speed-aware; (e) To provide effective guidance in choosing a good strategy, we performed a joint analysis of two conflicting goals based on the degradation in performance of each strategy under each metric; (f) Simulation results presented in the paper reveal that in terms of minimizing power consumption and maximization of the provider income *Min_e* allocation strategy outperform other algorithms. It dominates in almost all test cases. We conclude that the strategy is robust and stable even in significantly different conditions; (g) *MLp* also provides minor performance degradation and copes with different demands; (h) We find that the information about the speed of machines does not help to improve significantly allocation strategies. When examining the overall system performance on the real data, we determined that appropriate distribution of energy requirements over the system has a higher provider income and lower power consumption than other allocation strategies; (i) The final result suggests a simple allocation strategy, which requires minimal information and little computational complexity; nevertheless, it achieves good improvements in our objectives and provide quality of service guarantees.

However, further study of resource allocation algorithms for multiple service classes is required to assess their actual efficiency and effectiveness. This will be subject of future work for better understanding of service levels, QoS and multi-objective optimization in IaaS clouds. Also we will not restrict ourselves to finding a unique solution of a given problem, but a set of solutions known as a Pareto optimal set and compare them in terms of Pareto dominance for assessing the performance of strategies.

ACKNOWLEDGMENT

This work is partially supported by CONACYT (Consejo Nacional de Ciencia y Tecnología, México), grant no. 178415. The work of S. Nesmachnow was partly funded by ANII and PEDECIBA, Uruguay. The work of P. Bouvry and J. Pecero is partly funded by INTER/CNRS/11/03 Green@Cloud.

REFERENCES

[1] I. Ahmad and S. Ranka, Handbook of energy-aware and green computing, Chapman & Hall/CRC, 2012.

[2] Y. Zomaya, Y. Lee, Energy efficient distributed computing systems, Wiley-IEEE Computer Society Press, 2012.

[3] A. Lezama, A. Tchernykh, and R. Yahyapour "Performance evaluation of infrastructure as a service clouds with SLA constraints". Computación y Sistemas 17(3): 401–411, 2013.

[4] U. Schwiegelshohn and A. Tchernykh, "Online scheduling for cloud computing and different service levels," in 26th Int. Parallel and Distributed Processing Symp., Los Alamitos, CA, 2012, pp. 1067–1074.

[5] P. Patel, A. Ranabahu, and A. Sheth, "Service Level Agreement in cloud computing", In Proc. of OOPSLA Cloud Computing workshop, Orlando, Florida, 2009.

[6] L. Wu, S. Kumar Garg, and R. Buyya. "SLA-based admission control for a Software-as-a-Service provider in cloud computing environments". In Proc. Cluster, Cloud and Grid Computing (CCGrid), 2011, 195–204.

[7] C. Silaghi, G., Dan Şerban, L., & Marius Litan, C. "A framework for building intelligent SLA negotiation strategies under time constraints", In J. Altmann and O. Rana (eds.). Economics of Grids, Clouds, Systems, and Services, Berlin, Springer, 2010, pp. 48–61.

[8] L. Zhou, B. Zheng, J. Cui, and S. Tang, "Toward green service in cloud: From the perspective of scheduling" In Proc. Int. Conf. on Computing, Networking and Communications, 2012, pp. 939–943.

[9] A. Tchernykh, J. Pecero, A. Barrondo, E. Schaeffer. "Adaptive energy efficient scheduling in Peer-to-Peer desktop grids", Future Generation Computer Systems, July 2013. DOI: 10.1016/j.future.2013.07.011.

[10] P. Raycroft, R. Jansen, M. Jarus, and P. Brenner, "Performance bounded energy efficient virtual machine allocation in the global cloud", Sustainable Computing: Informatics and Systems, August 2013. DOI: 10.1016/j.suscom.2013.07.001.

[11] S. Khan, and I. Ahmad, "A cooperative game theoretical technique for joint optimization of power consumption and response time in computational grids". IEEE Transactions on Parallel and Distributed. Systems 20: 346–360, 2009.

[12] Y. Lee and A. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions", IEEE Transactions on Parallel and Distributed Systems 22:1374–1381, 2011.

[13] M. Mezmaz, N. Melab, Y. Kessaci, Y. Lee, E. G. Talbi, A. Zomaya, and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems". Journal of Parallel and Distributed Computing 71:1497–1508, 2011.

[14] J. Pecero, P. Bouvry, H. Fraire, and S. Khan, "A multi-objective GRASP algorithm for joint optimization of power consumption and schedule length of precedence-constrained applications". In Proc. Int. Conf. on Cloud and Green Computing, pp. 1–8, 2011.

[15] F. Pinel, B. Dorronsoro, J. Pecero, P. Bouvry, and S. Khan. "A two-phase heuristic for the energy-efficient scheduling of independent tasks on computational grids". Cluster Computing 16(3):421-433, 2013.

[16] P. Lindberg, J. Leingang, D. Lysaker, S. Khan, and J. Li, "Comparison and analysis of eight scheduling heuristics for the optimization of power consumption and makespan in large-scale distributed systems". Journal of Supercomputing 59(1):323–360, 2012.

[17] S. Nesmachnow, B. Dorronsoro, J. Pecero, and P. Bouvry. "Energy-Aware Scheduling on Multicore Heterogeneous Grid Computing Systems". Journal of Grid Computing 11(4):653–680, 2013.

[18] S. Iturriaga, S. Nesmachnow, B. Dorronsoro, and P. Bouvry. "Energy efficient scheduling in heterogeneous systems with a parallel multiobjective local search". Computing and Informatics Journal 32(2):273–294, 2013.

[19] B. DasGupta and M. Palis, "Online Real-time Preemptive Scheduling of Jobs with Deadlines on Multiple Machines", *Journal of Scheduling* 4(6), 297-312, 2001.

[20] PWA. (2014, March) [Online]. Available at http://www.cs.huji.ac.il/labs/parallel/ workload

[21] Grid Workloads Archive (2014, March) [Online]. Available at http://gwa.ewi.tudelft.nl

[22] D. Tsafrir, Y Etsion, and D. Feitelson. "Backfilling using system-generated predictions rather than user runtime estimates". IEEE Transactions on Parallel and Distributed Systems 18(6): 789-803, 2007.

[23] J.-M. Ramírez, A. Tchernykh, R. Yahyapour, U. Schwiegelshohn, A. Quezada, J. González, and A. Hirales. "Job Allocation Strategies with User Run Time Estimates for Online Scheduling in Hierarchical Grids". Journal of Grid Computing 9:95–116, 2011.

[24] A. Quezada, A. Tchernykh, J. González, A. Hirales, J.-M. Ramírez U. Schwiegelshohn, R. Yahyapour, and V. Miranda. "Adaptive parallel job scheduling with resource admissible allocation on two-level hierarchical grids". Future Generation Computer Systems 28(7): 965-976, 2012.

[25] E. Dolan, J. Moré, and T. Munson. "Optimality measures for performance profiles". SIAM Journal on Optimization 16:891-909, 2006

[26] A. Hirales-Carbajal, A. Tchernykh, T. Roblitz,, R. Yahyapour. "A Grid simulation framework to study advance scheduling strategies for complex workflow applications" Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on. pp.1-8, 19-23 April 2010, doi: 10.1109/IPDPSW.2010.5470918. Atlanta, Georgia. USA