Bi-Objective Online Scheduling with Quality of Service for IaaS Clouds

Andrei Tchernykh[1], Luz Lozano[1], Uwe Schwiegelshohn[2], Pascal Bouvry[3], Johnatan E. Pecero[3], Sergio Nesmachnow[4]

[1] CICESE Research Center, Ensenada, BC, Mexico, *{chernykh, llozano}@cicese.mx*
[2] TU Dortmund University, Dortmund, Germany, *uwe.schwiegelshohn@udo.edu*
[3] University of Luxembourg, Luxembourg, *{Pascal.Bouvry, Johnatan.Pecero}@uni.lu*
[4] Universidad de la República, Uruguay, *sergion@fing.edu.uy*

Abstract— **This paper focuses on the bi-objective experimental analysis of online scheduling in the Infrastructure as a Service model of Cloud computing. In this model, customer have the choice between different service levels. Each service level is associated with a price per unit of job execution time and a slack factor that determines the maximal time span to deliver the requested amount of computing resources. It is responsibility of the system and its scheduling algorithm to guarantee the corresponding quality of service for all accepted jobs. We do not consider any optimistic scheduling approach, that is, a job cannot be accepted if its service guarantee will not be observed assuming that all accepted jobs receive the requested resources. We analyze several scheduling algorithms with different cloud configurations and workloads and use the maximization of the provider income and minimization of the total power consumption of a schedule as additional objectives. Therefore, we cannot expect finding a unique solution to a given problem but a set of nondominated solutions also known as Pareto optima. Then we assess the performance of different scheduling algorithms by using a set coverage metric to compare them in terms of Pareto dominance. Based on the presented case study, we claim that a simple algorithm can provide the best energy and income trade-offs. This scheduling algorithm performs well in different scenarios with a variety of workloads and cloud configurations.**

***Keywords*—Cloud computing, Service Level Agreement, Energy Efficiency, Scheduling.**

I. INTRODUCTION

The shifting emphasis of clouds towards a service-oriented paradigm has led to the adoption of Service Level Agreements (SLAs). The use of SLAs has a strong influence on job scheduling, as schedules must observe quality of service constraints. To accommodate different customers with different needs, providers typically offer different levels of service. In a typical Infrastructure as a Service (IaaS) scenario, a customer has the choice between different resources that are available on demand and a variety of service guarantees. These service levels are mainly distinguished by the amount of computing power a customer is guaranteed to receive within a requested (or negotiated) time period, and a cost per unit of execution time he has to pay. This cost may depend on the type of computing resources, if different resources are available, for instance, processors and cores with different performance. Clouds typically serve two types of workloads: interactive service requests and batch jobs. Since we assume that the customers are always using the requested resources, a provider can only accept a job if the guaranteed amount of resources can be delivered. Therefore, the scheduling problem resembles deadline scheduling since a provider guarantees to observe the quality of service of a request once the job is accepted. We allow our scheduling algorithms to upgrade a request to use resources with a better performance without increase of the cost for the customer (voluntary upgrade).

The power consumption of the system is also addressed with the help of a low-level scheduler. We present several algorithms and use bi-objective analysis to discuss different scenarios for this model.

Schwiegelshohn and Tchernykh [4] have already analyzed single (SM) and parallel machine (PM) models subject to jobs with single (SSL) and multiple service levels (MSL). They use competitive analysis to determine the worst-case ratio between a provider income, when applying a given algorithm, and the optimal income. Such worst case performance bounds are provided for four greedy acceptance algorithms SSL-SM, SSL-PM, MSL-SM, MSL-PM, and two restricted acceptance algorithms MSL-SM-R, and MSL-PM-R.

To show the practicability and competitiveness of these algorithms, Lezama et al. [3] and Tchernykh et al. [23] presented simulation studies. In both cases, the authors use workloads based on real production traces of heterogeneous HPC systems to demonstrate practical relevance of the results. Based on these studies the rate of rejected jobs, the number of job interruptions, and the provider income strongly depend on the slack factor. The determination of the slack factor is subject to market constraints since it is set by the provider and must be accepted by the customer.

Tchernykh et al. [23] transformed the multi-objective problem into a single-objective one through the method of objective aggregation, assuming equal importance of each metric. First, the authors evaluate the degradation in performance of each strategy under each metric relative to the best performing strategy for the metric. Then, they average these values, and rank the strategies. The degradation approach provides the mean percentage of the degradation, but it does not show the negative effects of allowing a small portion of the results 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, the authors present performance profiles of the strategies.

In this paper, we avoid this problem by determining Pareto optimal solutions. Then, we assess the performance of different

strategies by comparing two algorithms in terms of Pareto dominance with the help of a set coverage metric [24].

The paper is structured as follow. The next section reviews related works on SLAs and energy-aware scheduling. Section III presents the problem definition, while the proposed schedulers are described in Section IV. Section V provides details of the experimental setup. The experimental results are reported in Section VI for the proposed bi-objective schedulers when solving a benchmark set of different problem instances and scheduling scenarios. Moreover, practical approximation of Pareto fronts and their comparison using a set coverage metric are presented.

II. Related work

There has been significant research on SLAs in Cloud computing [5]; usage of SLAs for resource management and admission control techniques [3, 6]; automatic negotiation protocols [7]; economic aspects associated with the usage of SLAs for service provision and the incorporation of SLA into the Cloud architecture [3]. However, little is known about efficiency of SLA scheduling solutions. There are few theoretical results on SLA scheduling mostly addressing real time scheduling with given deadlines. Schwiegelshohn and Tchernykh [4] present a more complete study.

Three main policies are used to optimize power consumption [1,2]: Dynamic component deactivation switches off parts of the computer system that are not utilized; Dynamic Voltage (and Frequency) Scaling (DVS/DVFS) slows down the speed of CPU processing; Explicit approaches (e.g. SpeedStep by Intel, or Optimized Power Management by AMD) use hardware-embedded energy saving features. These policies are designed to reduce the power consumption of one resource individually, but not for a whole system consisting of geographically distributed resources.

Tchernykh et al. [8] explored the benefits of the first approach for distributed systems. It turns off/on (activate/disactivate) servers so that only the minimum number of servers required to support the QoS for a given workload are kept active. Raycroft et al. [9] analyzed the effects of virtual machine allocation on power consumption.

DVS/DVFS energy-aware approaches have been commonly addressed in literature, from early works like the one by Khan and Ahmad [10] using a cooperative game theory to schedule independent jobs on a DVS-enabled grid to minimize makespan and energy. Lee and Zomaya [11] 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. [12] by proposing a parallel bi-objective hybrid genetic algorithm (GA). Pecero et al. [13] 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. [14] used a double minimization approach and a local search to solve the problem. Lindberg et al. [15] proposed six greedy algorithms and two GAs for solving the makespan-energy scheduling problem subject to deadline constraint and memory requirements.

Nesmachnow et al. [16] studied an explicit approach using the Max-Min mode.

Twenty fast list scheduling off-line 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. [17] showed that a parallel multi-objective local search based on Pareto dominance outperforms deterministic heuristics based on the traditional *Min-Min* strategy.

Job scheduling is a crucial part of efficient cloud implementation. In this paper, diversified aspects of the problem are discussed to cope with the new challenges of multi-domain distributed systems: hierarchical models; dynamic scheduling policies; multi-objective optimization; adaptive policies; QoS and SLA constraints; economic models; scheduling of computational intensive batch applications; and energy efficiency among other topics. We face a fundamental research problem: how to minimize power consumption and maximize provider income without violation of SLAs.

III. Problem definition

This section describes the system model and power consumption model presented by Tchernykh et al. [23].

A. Job 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 SL_j^l, the job J_j has the performance requirement s_j^l of the request that is guaranteed by providing processing capability of VM instances (MIPS - Million Instructions per Second), and 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, w_j, d_j, SL_j^l) containing the release date r_j, the amount of work w_j that represents the computing load of the application (MI - Million Instructions) that has to be done before the required response time, the deadline d_j, and the service level $SL_j^l \in SL$. Let $p_j = w_j/s_j^l$ be needed guaranteed time, according to the service level SL_j^l, the system will spend for processing of the job before its deadline. Let d_j be 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 released, characteristics of the job become 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 not. 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.

B. Machine model

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 its relative processing speed s_i and its 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. This set is defined for each job as a subset of available machines that can execute this job without its deadline violation, and can guarantee computing power s_j^l for processing. Machines that have processing speed less than guaranteed by SLA cannot accept the job. Otherwise, the job can be accepted, and the machine spends $p_j^i = p_j/s_i = w_j/(s_j^l \cdot s_i)$ time for processing of the job before its deadline.

The value s_i is conservatively selected such that the speed-ups of all target applications exceed s_i, hence, users get the same guarantees whatever processors are used. Deadlines are calculated based on the service level and cannot be changed, and guaranteed processing time is not violated by slower processing. C_{max} denotes the makespan of the schedule.

C. 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 busy, otherwise $w_i(t) = 0$. The total power consumed by the cloud system is the sum of power consumed during cloud operation: $E^{op} = \int_{t=1}^{C_{max}} P^{op}(t)dt$ with $P^{op}(t) = \sum_{i=1}^{m} P_i(t) = \sum_{i=1}^{m} o_i(t) \cdot (P^{idle} + w_i(t)P^{work})$,

D. Optimization Criteria

Two criteria are considered in the analysis: maximization of the service provider income, and minimization of the power consumption E^{op}. The income V is defined as: $V = \sum_{j=1}^{n} (u_j^l \cdot p_j)$.

IV. Scheduling algorithms

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

A. Scheduling approach

We use a two-level scheduling approach. At 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 (see Table 1).

B. Higher-level acceptance policies

We use greedy higher-level acceptance policies. It is based on the Earliest Due Date algorithm, which gives priority to jobs according to their deadlines. When a job J_j arrives to the system, in order to determine whether to accept it or reject it, the system 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 their 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 [18] showed that there cannot exist an algorithm with competitive ratio greater than $1 - 1/f_i + \varepsilon$, with $m \geq 1$, and $\varepsilon > 0$, if preemption without migration is allowed. The authors proposed 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, the machine energy efficiency and the energy consumed by executing a job are 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 [8, 21, 22]; ii) *energy-aware*, with power consumption information; and iii) *speed-aware* with speed of machines information.

V. Experimental setup

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 [19], and the Grid Workload Archive [20].

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. The workloads include nine traces from:

Table 1. Allocation strategies

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 2: Experimental setup

Site	Procs	Power consumption W		Energy efficiency MFlopS/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

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 [19] and [20].

B. Scenarios

The problem scenarios considered in the experimental analysis have the following details: workload of seven days that includes batch jobs; greedy acceptance policy 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 speed is in the range of [18.6, 481], efficiency is in [0.89, 1.75], power consumption is in [17.8, 58.9] for P^{idle}, and in [26, 66] for P^{work}.

VI. EXPERIMENTAL RESULTS

For all scenarios, we define the number of machines and number of SLs, which we use for experimental study. We consider eight infrastructure sizes with the number of machines being powers of 2 from 1 to 128. The set of SLs is $SL = [SL_1, \dots, SL_4]$ with slack factors $f_1 = 1, \dots, f_4 = 4$.

A. *Bi-objective analysis*

Multi-objective optimization is not restricted to find a unique solution of a given problem but a set of solution known as a Pareto optimal set. One solution can represent the best solution concerning energy consumption, while another solution could be the best one concerning the income. The goal is to choose the most adequate solution and obtain a set of compromise solutions that represents a good approximation to the Pareto front. Two important characteristics of a good solution technique are convergence to the Pareto front, and diversity to sample the front as fully as possible. A solution is Pareto optimal if no other solution improves it in terms of all objective functions. Any solution not belonging to the front can be considered of inferior quality to those that are included. The selection between the solutions included in the Pareto front depends on the system preference. If one objective is considered more important than the other one, then preference is given to those solutions that are near-optimal in the preferred objective, even if values of the secondary objective are not among the best obtained.

Often, results from multi-objectives problems are compared via visual observation of the solution space. A more formal and statistical approach uses a set coverage metric SC(A,B) [24] that calculates the proportion of solutions in B, which are dominated by solutions in A:

$$SC(A,B) \triangleq |\{\mathrm{b} \in \mathrm{B};\ \mathrm{a} \in \mathrm{A}\ ;\ \mathrm{a} \leq \mathrm{b}\}|/|B|$$

A metric value $SC(A,B) = 1$ means that all solutions of B are dominated by A, whereas $SC(A,B) = 0$ means that no member of B is dominated by A. This way, the larger the $SC(A,B)$, the better the Pareto front A with respect to B. Since the dominance operator is not symmetric, $SC(A,B)$ is not

necessarily equal to $1 - SC(B,A)$, and both $SC(A,B)$ and $SC(B,A)$ have to be computed for understanding how many solutions of A are covered by B and vice versa.

B. Solution space and Pareto fronts

Our aim is to obtain a set of compromise solutions that represent a good approximation to the Pareto front. This is not formally the Pareto front as an exhaustive search of all possible solutions is not carried out, but rather serves as a practical approximation of a Pareto front.

To obtain valid statistical values, 30 experiments of 7 days, testing 8 configurations with different number of machines, and 4 SLs are conducted. We compute a set of solutions approximating the Pareto front of each of eight strategies (FFit, Max-s, Max-eff, Max-seff, MCT-eff, Min-e, Rand, MLp) and get approximations of Pareto fronts considering 30×8×4=960 solutions.

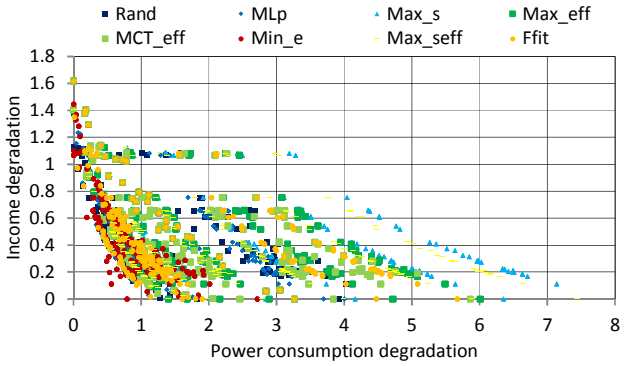

Figure 1. The solution sets

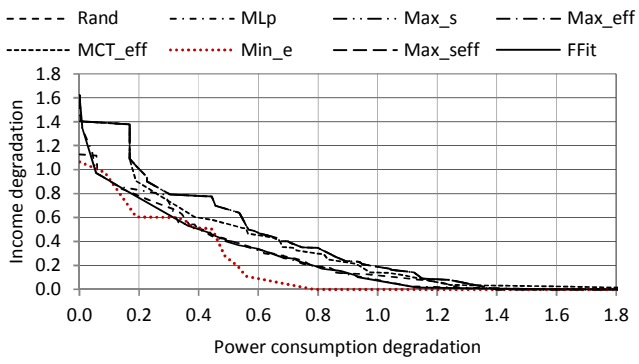

Figure 2. The Pareto fronts

Figures 1-2 show the solution sets and the Pareto fronts. Note that we address the problem of minimizing power consumption and maximization of the income providing QoS guarantees. For better representation, we convert it to the minimization of two criteria: degradation of both the income and power consumption. We evaluate the degradation in performance of each strategy under each metric. This is done relative to the best performing strategy for the metric, as follows: $\frac{strategy\ criterion\ value}{best\ found\ criterion\ value} - 1$.

Figure 1 shows the solution space (two objectives) including 960 solutions for each strategy, 7680 solution in total. Each solution is represented by income degradation and power consumption degradation.

The approximated Pareto fronts cover a wide range of values in terms of E^{op} degradation from 0 to 8, whereas values of V degradation are very similar to one another in the range from 0 to 1.6.

Figure 2 shows the eight approximations of Pareto fronts generated by the studied strategies. It can be seen that Min-e, MLp and FFit are located in the lower-left corner, being among the best solutions in terms of both objectives. They noticeably outperform MCT-eff.

However, we should not consider only Pareto fronts. When many of the solutions are outside the Pareto front, the algorithm's performance is variable. This is the case of FFit: although the Pareto front is of high quality, many of the generated solutions are quite far from it, and, hence, a single run of the algorithm may produce significantly different results. FFit solutions cover E^{op} degradations from 0 to 6, whereas Min-e solutions are in the range from 0 to 3.3 of E^{op} degradations. This indicates a more stable behavior of Min-e. As expected the energy waste is increasing with increasing income. For solutions with the best income (zero degradation), power consumption degradation is more than 1.4 for most of the strategies. Min-e shows a better result with 0.8 degradation.

C. Set coverage

In this section, to compute the performances of the multi objective strategies we use the set coverage metric (see Section VI.C). Using this metric, two sets of non-dominated solutions can be compared to each other.

Table 3: Set coverage

A \ B	FFit	Max-s	Max-eff	Max-seff	MCT-eff	Min-e	Random	MLp	Mean	Ranking
FFit	1.00	0.92	0.92	0.92	0.93	0.40	0.39	0.26	0.68	2
seMax-s	0.19	1.00	1.00	1.00	0.20	0.13	0.11	0.16	0.4	4
Max-eff	0.19	1.00	1.00	1.00	0.20	0.13	0.11	0.16	0.4	4
Max-seff	0.19	1.00	1.00	1.00	0.20	0.13	0.11	0.16	0.4	4
MCT-eff	0.11	0.70	0.68	0.71	1.00	0.13	0.07	0.13	0.36	5
Min-e	0.70	0.97	0.95	0.97	0.93	1.00	0.68	0.74	0.85	**1**
Rand	0.26	0.95	0.92	0.95	0.90	0.33	1.00	0.26	0.65	**3**
MLp	0.33	0.95	0.89	0.92	0.90	0.33	0.39	1.00	0.67	**2**
Mean	0.28	0.93	0.91	0.92	0.61	0.23	0.27	0.27		
Ranking	3	7	5	6	4	**1**	**2**	**2**		

Table 3 reports the SC results for each of the eight Pareto fronts. The rows of the table show the values $SC(A,B)$ for the dominance of strategy A over strategy B. The columns indicate $SC(B,A)$, the dominance of B over A. The last two columns show the average of $SC(A,B)$ for row A over column B, and ranking based on the average dominance. Similarly, the last two rows show average dominance B over A, and rank of the strategy in each column. We see that $SC(Min-e,B)$ dominates the fronts of the other strategies on the range 93%-68%, and 85% in average. $SC(A,B)$ is not necessary equal to $1 - SC(B,A)$. $SC(A,Min-e)$ shows that Min-e is dominated by the fronts of other strategies on 23% in average. The ranking

of strategies is based on the percentage of coverage. The higher ranking of rows implies that the front is better. The rank in columns shows that the smaller the average dominance, the better the strategy. According to the set coverage metric table, the strategy that has the best compromise between maximizing income and minimizing energy consumption is Min-e, followed by MLp and FFit on the second position.

VII. 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 a cost per unit of execution time. In return, the user receives guarantees regarding the provided resources: maximum response time (deadline) used as QoS constraints.

Our experimental case study results in several contributions. We provide an experimental analysis of eight allocation strategies that take into account heterogeneity of the system. We distinguish allocation strategies depending on the type and amount of information they require: knowledge free, energy-aware, and speed-aware. To provide effective guidance in choosing a good strategy, we performed a joint analysis of two objectives based on Pareto front and set coverage metric. Simulation results reveal that in terms of minimizing power consumption and maximizing provider income *Min-e* outperforms other allocation strategies. It dominates in almost all test cases. We showed that the strategy is stable even in significantly different conditions. *MLp* and *FFit* also provide major dominance with set coverage and cope with different demands. We find that the information about the speed of machines does not help to improve significantly the allocation strategies. When examining the overall system performance on the real data, we determined that appropriate distribution of energy requirements over the system provide more benefits in income and power consumption than other strategies. *Min-e* is a simple allocation strategy requiring minimal information and little computational complexity. Nevertheless, it achieves good improvements in both objectives and quality of service guarantees. However, further study for multiple service classes is required to assess its 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.

ACKNOWLEDGMENT

This work is partially funded by CONACYT grant #178415. The work of S. Nesmachnow is 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, 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, R. Yahyapour "Performance Evaluation of Infrastructure as a Service Clouds with SLA Constraints". *Computación y Sistemas* 17(3): 401–411, 2013.

[4] U. Schwiegelshohn, A. Tchernykh, "Online Scheduling for Cloud Computing and Different Service Levels," *26th Int. Parallel and Distributed Processing Symp.*, Los Alamitos, CA, 2012, pp. 1067–1074.

[5] P. Patel, A. Ranabahu, A. Sheth, "Service Level Agreement in Cloud Computing", *OOPSLA Cloud Computing workshop*, Orlando, 2009.

[6] L. Wu, S. Kumar Garg, R. Buyya. "SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments". *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] 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.

[9] P. Raycroft, R. Jansen, M. Jarus, P. Brenner, "Performance bounded energy efficient virtual machine allocation in the global cloud", *Sustainable Computing: Informatics and Systems*, August 2013.

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

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

[12] M. Mezmaz, N. Melab, Y. Kessaci, Y. Lee, E. G. Talbi, A. Zomaya, 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

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

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

[15] P. Lindberg, J. Leingang, D. Lysaker, S. Khan, 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.

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

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

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

[19] PWA. [Online]. http://www.cs.huji.ac.il/ labs/parallel/ workload

[20] Grid Workloads Archive [Online]. Available at http://gwa.ewi.tudelft.nl

[21] J.-M. Ramírez, A. Tchernykh, R. Yahyapour, U. Schwiegelshohn, A. Quezada, J. González, 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

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

[23] A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J. E. Pecero, S. Nesmachnow. Energy-Aware Online Scheduling: Ensuring Quality of Service for IaaS Clouds. International Conference on High Performance Computing & Simulation (HPCS 2014). 2014. Bologna, Italy

[24] E. Zitzler, Evolutionary algorithms for multiobjective optimization: Methods and applications, PhD thesis, Swiss Federal Institute of Technology. Zurich, 1999.