Towards a PaaS Architecture for Resource Allocation in IaaS Providers Considering Different Charging Models

Cristiano C. A. Vieira[1], Luiz F. Bittencourt[2], and Edmundo R. M. Madeira[2]

[1] Federal University of Mato Grosso do Sul - Faculty of Computing
Cidade Universitária, Campo Grande/MS - Brasil
[2] University of Campinas - Institute of Computing
Av. Albert Einstein, 1251 Cidade Universitária, Campinas/SP - Brasil

Abstract. With the increase in computing infrastructure commercialization through the pay-as-you-go model, competition among providers puts the user as a decision agent on which is the best provider to comply with his/her demands and requirements. Currently, users rely on instances offered as on-demand, reserved, and spot to decide which is the best resource allocation model over the time. In this work, we present substantial contributions to compose a PaaS architecture that leverages different charging models, where we propose the use of a new charging model called time-slotted reservation. Moreover, we developed an integer linear program (ILP) to perform the scheduling of incoming requests according to different QoS levels, proposing a mapping of those levels into the charging models offered by IaaS providers. Simulations show the applicability of the ILP in the proposed model, being able to maximize the number of requisitions executed following the user's QoS requirements.

Keywords: Cloud Computing, Architecture, PaaS, IaaS, Charging model, Scheduling

1 Introduction

The increase in the pay-as-you-go model in Infrastructure as a Service (IaaS) cloud providers allowed corporations to reduce the initial capital needed for IT infrastructure. This popularization, not coincidently, comes as both higher bandwidth is available over the Internet and virtualization technologies maturates. In this sense, IaaS cloud providers such as Amazon, GoGrid, lixiscale, Windows Azure, and Ninefold, offer different charging models to commercialize different types of services. One of these services is the virtual machine (VM) leasing, where the user can choose to lease a VM from a variety of hardware configurations (processor speed, processor cores, RAM, storage, and so on). The main charging models currently available are *on-demand* (OD), *reserved* (RE), and *spot* (SP), which present differences in availability and charged price.

The competition among IaaS providers puts the user as a decision agent that selects the provider that best matches the application demands and requirements. When the user has many requests with different running times and quality of service (QoS) requirements, it is desirable to automate the decision-making process of choosing what type of VM, and with which charging model, should run each request.

In general, the resource allocation problem in IaaS providers can be tackled with two different objectives: reduce costs to the client [1][2][3] or increase the provider income guaranteeing the user satisfaction [4][5][6][7]. We are interested in the first objective, i.e., reducing the running costs for the client of public cloud providers. A good application scheduling can reduce the allocation costs, avoiding high budgets when running applications in the cloud.

In this work, we present substantial contributions to compose a PaaS architecture that helps customers to schedule VM requests on different public clouds. The architecture leverages different charging models, and we propose the use of a new charging model, called *time-slotted reservation*, which enables a better utilization for leased instances in the reserved (RE) charging model. The proposed platform supports two levels of SLA. The first SLA level governs the interaction between the user and the PaaS, while the second level governs the the interaction of the PaaS with a set of IaaS providers, and therefore contains charging models in use.

We consider that the platform belongs to an organization and receives requests for VMs to be allocated to public clouds. In this sense, the main contributions of this work are: a) a PaaS architecture with two SLA levels; b) a new charging model called “time-slotted reservation-TS”; c) a mapping proposal between the two SLA levels of the architecture; d) an Integer Linear Program (ILP) formulations for scheduling; and e) analysis of experimental results.

In the next section we give an overview of the related work. The background and problem formulation are provided in Section 3. A PaaS architecture is proposed in Section 4. A detailed discussion on simulation set-up, metrics, and experimental results is given in Section 5. Finally, Section 6 presents the conclusions and the future work.

2 Related Work

The utilization of public clouds to extend the locally available computing power has been widely explored recently [8][9]. Zhao et. al. [5] present CloudBay, a platform to offer resources from different public clouds utilizing auction strategies. Other auction strategies for marketing resources in clouds were proposed [10][11].

Cloud federation was proposed by Toosi et. al. [6], where a provider uses resources from other providers in the federation to meet the need for reserved instances. The provider can trade reserved instances but also sell them in the on-demand (OD) model. When the client wants to utilize the reserved instance, the provider can look for an instance in the federation to serve the user.

Chen et. al. [7] developed a new utility model for measuring customer satisfaction in the cloud. Based on the utility model, they designed a mechanism to support utility-based SLAs in order to balance the performance of applications and the cost of running them. They presented two scheduling algorithms that can effectively bid for different types of VM instances to make trade-off between profit and customer satisfaction.

An ILP is presented by Genez et al. [2] to solve the scheduling problem for dependent services in SaaS/PaaS with two SLA-levels. They utilize both RE and OD models in the simulations. Similarly, but for a single SLA level, Bittencourt et al. [3] propose an algorithm to schedule workflows which considers costs and deadlines, to select VM configurations from public IaaS providers to expand the available computational power locally available.

Assunção et al. [1] investigated the benefits that organizations can reach by using cloud computing providers to increase the computing capacity of their local infrastructure. They evaluated the cost of seven scheduling strategies used by an organization that operates a cluster managed by virtual machine technology and seeks to utilize resources from a remote IaaS provider to reduce the response time of its user requests.

Although some work uses charging models, a few consider OD, RE and SP charging models in scheduling. It is interesting to consider the utilization of different charging models for different QoS categories. Besides, some work deals with the provider perspective only, i.e., they aim to increase the provider profit. In this work, we focus on decreasing the scheduling cost for the user.

3 Background and Problem Formulation

Users request VMs from an IaaS provider to run their applications, therefore we define the user need for a single VM as a *VM request*. A user can perform a number of requests to run applications that need different QoS levels. More specifically, considering execution time as the main QoS parameter, in this work we classify the VM requests into three QoS categories:

– Fixed-time request (FTRx): In this category, the start time is immediate and the VM cannot be interrupted (preempted) during its execution.
– Floating-time request: (FTRt): The request may not start immediately, but once it is started, it cannot be interrupted (preempted).
– Variable-time request (VTR): The request may not start immediately and it can be interrupted (preempted), i. e., the execution of requests in this category can be fragmented in smaller parts.

In the face of the wide availability of services offered by a variety of IaaS providers, the user has the burden of choosing which resources from which provider he/she should utilize. On the other hand, providers have SLAs composed of different charging models utilized to lease the resources, which regulates how and how much the user will pay for leasing a VM instance. In this context, a user must choose VMs and charging models according to requests with different QoS needs. We consider that VMs requests from the user must be allocated over VMs from IaaS providers with the objective of achieving a low scheduling cost. Moreover, the set of VMs considered during the scheduling process is already leased, thus we are dealing with a pre-determined set of already rented VMs from various providers. Optimizing the use of currently leased VMs allows the user/organization to determine which VMs are needed, reducing costs.

The problem tackled in this paper is stated as follows: *Compute a schedule S to a set $\mathcal{R}$ of VM requests over a set Ψ of VM instances from IaaS providers with the objective of achieving the smallest allocation cost without violating QoS.*

The $\mathcal{R}$ set is composed of n VM instance requests $r = \{k, d, d\alpha, qos\}$, where $r(k)$ represents the instance type, $r(d)$ represents the total time for which the instance must be available to the user, $r(d\alpha)$ represents a relaxing over d, and $r(qos) = \{$ FTRx | FTRt | VTR $\}$ corresponds to the QoS category of the request. By definition, we have $d = d\alpha$, $\forall\ r \in \mathcal{R}$ such that $r(qos) = \{FTRx\}$; and $d < d\alpha$, $\forall\ r \in \mathcal{R}$ such that $r(qos) = \{FTRt, VTR\}$.

Each provider has a set of VM instances available to the user. From the user perspective, it is necessary to verify how requests can be fulfilled, and at which costs, utilizing the VMs from the providers. A request must be fulfilled with the smallest cost without violating its maximum execution time. In this work, we assume that tasks have no dependency among them. Figure 1 presents the scenario we are considering in this work.

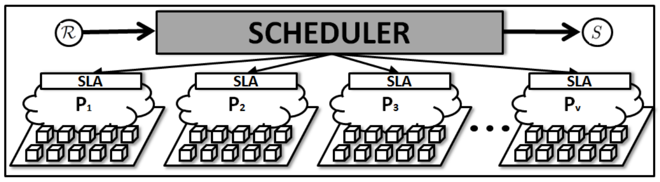


Fig. 1. Scenario for request submission.

Given a set $\mathcal{R}$ of n requests, we must obtain a schedule $S = \{t, q, c, E\}$, composed, respectively, of: t, the total execution time of $\mathcal{R}$; q, the number of requests fulfilled; c, the lowest cost to the user and the scheduling containing the instance; and E, the time at which each request is to be allocated. Moreover, we denote, as follows, some definitions utilized in this paper. Let:

- $\mathcal{R} = \{r_1, r_2, ..., r_n\}$: set containing n requests;
- $\mathcal{P} = \{p_1, p_2, ..., p_v\}$: set of v providers;
- Ψ_i: set of instances available to the user in the provider p_i;
- $\Psi = \bigcup_{i=1}^{v} \Psi_i$:set of all instances available to the user;
- $\mathcal{B}^{s1}$: set of PaaS charging models;
- $\mathcal{B}_i^{s2}$: IaaS providers i charging models set;
- m: number of instances in Ψ.

4 Towards a PaaS Architecture

Considering the problem formulated in Section 3, we present substantial contributions to compose a PaaS that receives a set of VM requests and performs the scheduling of these requests over a set of instances available from IaaS providers. The objective is to compute a schedule that minimizes the execution costs to the user [3]. Next, we present the PaaS architecture, proposing a new charging model and a mapping between the two levels of SLA, and an ILP to compute the schedule in this architecture.

4.1 PaaS Architecture

The proposed architecture is composed of three layers: application, business, and infrastructure. Figure 2 presents the platform architecture. The application layer receives a set of requests $\mathcal{R}$ and performs their scheduling over the instances available to the PaaS utilizing a scheduling algorithm. The business layer is composed of the allocation strategies and the charging models, both of which utilized by the scheduler to allocate a request r in a provider p_i. The business models in this layer can be changed with no interference in the models made available by the IaaS providers. The infrastructure layer is responsible for maintaining information to manage and monitor the IaaS providers used by the PaaS. We consider all requests are allocated to the IaaS providers.

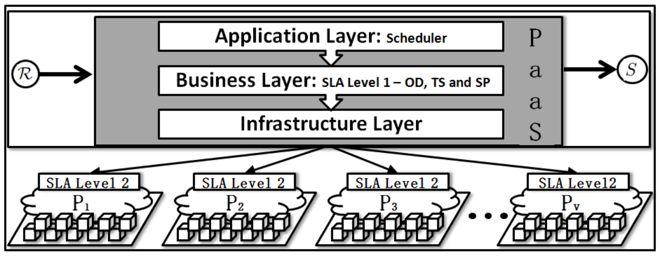


Fig. 2. Architecture utilized in the VM scheduling.

Two SLA levels are present in the platform. The first level acts in the interaction between the user and the PaaS, while the second level is between the interaction among the PaaS and the IaaS providers. Charging models belonging to level 1 are located in the business layer, and charging models belonging to the second level refer to the models adopted by the IaaS providers.

[3] The proposed PaaS can be extended to act as a Broker [12], where it could lease VMs from the IaaS providers and offer them to the users. However, it would be necessary to consider economic aspects to enable this broker to be profitable. At this point, we do not consider this implementation as a profitable broker.

4.2 Increasing Instance Utilization

An RE instance imposes the payment of an initial surcharge to offer a discounted price during the VM utilization. This strategy can be advantageous to the customer only if he promotes a high utilization of the reserved instance, otherwise the cost per time unit can end up larger than the one for the OD instance. For example, Fig. 3 presents scenarios where the customer utilizes 10%, 20%, 40%. 60%, 80%, and 100% of the time of RE instances for 1 and 3 years, considering prices and configurations from Amazon EC2[4]. A one year usage of an OD instance costs $1,051.20, while for 100% of utilization for an RE instance would cost $644.22, including the initial surcharge. On the other hand, utilizing the RE instance only 25% of the time would have a cost of $368.98. The same use (2,190 hours) in the OD charging model would cost $262.80. Therefore, low utilization makes the RE instance to be more expensive than the OD instance. Moreover, we can see from the *limit economy* line that, for current market prices, the customer should utilize more than 40% of an instance reserved for one year, and 20% for an instance reserved for three years to them to be worthless.

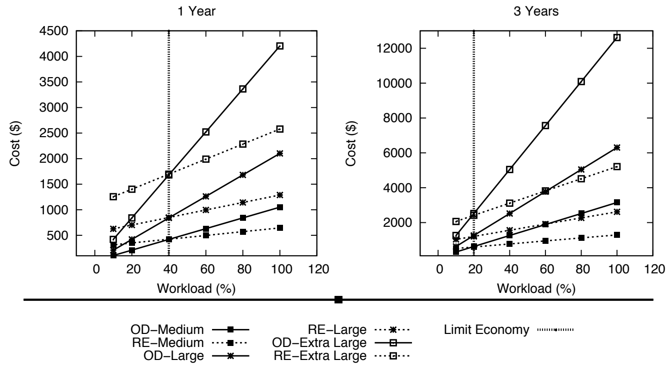


Fig. 3. Comparative costs for 1 and 3 years reservation against on demand instances.

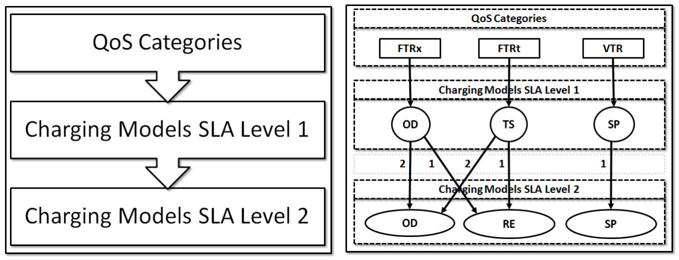
This analysis shows that the user can reduce costs by increasing utilization of RE instances. Following this, we propose a new charging model called "Time-slotted reservation - TS", where the user schedules a VM utilization in slots of time. In this model, requests cannot be interrupted. This enables better utilization of RE instances, minimizing the cost per time unit. We added the new TS charging model to $\mathcal{B}^{s1}$, as well as the OD and SP models. Thus, we have that $\mathcal{B}^{s1} = \{\text{OD} \mid \text{TS} \mid \text{SP}\}$ is placed in the business layer of the architecture presented in Fig. 2.

[4] From: Amazon - http://aws.amazon.com/ec2/pricing/ in 05/2013.

4.3 A Mapping Proposal Between the Two SLA Levels of the Architecture

As presented in Fig. 2, the platform is composed of two SLA levels. Considering the QoS categories (FTRx, FTRt, VTR), we define a conceptual model of interaction between these QoS categories, the level-1 SLA set $\mathcal{B}^{s1}$, and the level-2 SLA set $\mathcal{B}^{s2}$, as illustrated in Fig. 4(a). The conceptual model allows new charging models to be incorporated without interfering in the charging models currently available from IaaS providers.

Figure 4(b) presents a mapping proposal between the two levels in the conceptual model. Other possibilities can be explored, mainly with the appearance of new models. In this work we focused in the mapping from Fig. 4(b).


QoS Categories
Charging Models SLA Level 1
Charging Models SLA Level 2
QoS Categories
FTRx
FTRt
VTR
Charging Models SLA Level 1
OD
TS
SP
2
1
2
1
1
Charging Models SLA Level 2
OD
RE
SP

(a) Conceptual model (b) Mapping proposal

Fig. 4. a)Model of interaction between QoS categories and SLAs at levels 1 and 2. b) Proposed charging model mapping between SLA1 and SLA2.

The presented mapping proposal has the objective of mapping a charging model from $\mathcal{B}^{s1}$ to another from $\mathcal{B}^{s2}$. The SP model at level 2 has low availability guarantee, and therefore it can receive only mappings from the SP model from level 1. The OD and TS models at level 1 can be mapped to both OD and RE models at level 2. However, there is a scheduling priority in the mapping of RE over the OD, aiming at higher utilization (and thus lower prices) using the RE charging model.

4.4 Requests Scheduling

The requests scheduling is obtained utilizing the ILP and two heuristics.

An Integer Linear Program (ILP) Formulation for Scheduling

The integer linear program solves the scheduling problem through the binary variables w, x, y and z and the constants C, M and K as follows:

- w_r: binary variable that assumes the value 1 if request r is executing; otherwise it assumes the value 0;

- $x_{r,\psi}$: binary variable that assumes the value 1 if request r is executing in VM ψ on independent time t; otherwise it assumes the value 0;
- $y_{r,t,\psi}$: binary variable that assumes the value 1 if request r is executing in VM ψ on the time t; otherwise it assumes the value 0;
- $z_{r,t,\psi}$: binary variable that assumes the value 1 if request r with $r(qos) = \{FTRt\}$ starts executing on time t in VM ψ; 0 otherwise;
- $C_{t,\psi}$: constant that assumes the cost per time unit for using the VM ψ.
- $\mathcal{M}$: sufficiently large constant that assigns a weight for each request.
- $\mathcal{K}$: sufficiently large constant used to ensure FTRt requests start once.

We formulate the objective function $F = \sum\limits_{r\in\mathcal{R}} \sum\limits_{t\in T} \sum\limits_{\psi\in\Psi} (y_{r,t,\psi} \times C_{t,\psi}) - \sum\limits_{r\in\mathcal{R}} (w_r \times \mathcal{M})$ that computes the scheduling of $\mathcal{R}$ in Ψ aiming lower cost allocation. Thus, we want to minimize F subject to:

$$\sum_{t=1}^{r(d\alpha)} y_{r,t,\psi} = r(d) \times x_{r,\psi}; \forall r \in \mathcal{R}, \forall \psi \in \Psi \tag{1}$$

$$\sum_{\psi\in\Psi} \sum_{t=d\alpha+1}^{T} y_{r,t,\psi} = 0; \forall r \in \mathcal{R} \tag{2}$$

$$\sum_{\psi\in\Psi} x_{r,\psi} = w_r; \forall r \in \mathcal{R} \tag{3}$$

$$\sum_{r\in\mathcal{R}} y_{r,t,\psi} \leq 1; \forall t \in \Big[1, r(d\alpha)\Big], \forall \psi \in \Psi \tag{4}$$

$$\sum_{r\in R} y_{r,t,\psi} = 0; \forall t \in T; \psi \in \Psi; MF(r,\psi) = 0 \tag{5}$$

$$r(d) - \mathcal{K} \times (1 - z_{r,t,\psi}) \leq \sum_{s=t}^{r(d)+t-1} y_{r,s,\psi} \leq r(d) + \mathcal{K} \times (1 - z_{r,t,\psi}) \tag{6}$$

$$\forall r \in \mathcal{R}, \forall t \in \Big[1, r(d\alpha) - r(d) + 1\Big], \forall \psi \in \Psi, r(qos) = \{FTRx, FTRt\}$$

$$\sum_{t=1}^{\mathcal{L}} z_{r,t,\psi} = x_{r,\psi}; \forall r \in \mathcal{R}, \forall \psi \in (\Psi^o \bigcup \Psi^a) \tag{7}$$

$$w_r, x_{r,\psi}, y_{r,t,\psi}, z_{r,t,\psi} \in \{0,1\}; \forall r \in \mathcal{R}, \forall t \in \mathcal{T}, \forall \psi \in \Psi \tag{8}$$

The ILP utilizes the binary variables w_r, $x_{r,\psi}$, $y_{r,t,\psi}$, $z_{r,t,\psi}$ and constants $\mathcal{K}$, $\mathcal{M}$ and $C_{t,\psi}$ to compute a schedule that minimizes the cost, but also utilizing VM instances from the IaaS providers that guarantee the QoS of each request. Moreover, it is built over the mapping presented in Fig. 4(b).

The constraints (C-1) and (C-3) specify that a request, if executed, must be executed in $r(d)$ time units and in a single VM. The constraint (C-2) specifies that a request mustn't be executed in $t > r(d\alpha)$, while the constraint (C-4) specifies that a VM must perform one request per time. The constraint (C-5) specifies that a request must be executed considering the proposed charging model mapping between SLA1 and SLA2 presented in Fig. 4(b). The constraints (C-6) and (C-7) are used to ensure FTRt requests are executed so atomic. The constraint (C-1) ensures the atomicity for FTRx requests. The last constraint (C-8), specifies that the variables of this ILP, called w, x, y, and z, will only assume the binary values. MF() is a mapping function.

The schedule generated is the best schedule that can be achieved with the highest number of requests fulfilled. A request cannot be partially fulfilled. However, the number of VMs available may not suffice to run all requests. In this case, the ILP returns the schedule with the highest number of requests that can be fulfilled, and the smallest cost to run them.

The constant $\mathcal{M}$ is used to achieve the schedule with the highest number of requests, since it establishes a weight for the variable w_r. This means that the larger the number of w_r variables set to true, the smaller will be the ILP result.

Let C_t be the total cost returned by the ILP. Then, $s.q = \lfloor \frac{-C_t}{\mathcal{M}} \rfloor + 1$ and $s.c = (s.q \times \mathcal{M}) - C_t$. Let d_m and ψ_m, respectively, be the largest execution time among all requests $r \in \mathcal{R}$ and the largest execution cost per time unit among all VMs $\psi \in \Psi$. Following this, $s.q$ and $s.t$ can be computed as stated before only if $\mathcal{M} > d_m \times \psi_m$. This is how the constant M must be defined, otherwise it would compromise the result obtained by the ILP.

Heuristics We implemented two heuristic algorithms to compute the scheduling of $\mathcal{R}$ over VMs of Ψ: 1-FIFO (*First-in First-out*) and 2-DO (*Doubly Ordered*):

1. **FIFO**: Performs the request scheduling using the strategy of allocation the first request in the first VM possible, considering the time duration, the relaxation and the QoS category. Let l be the highest value of $d\alpha$ among all requests. The asymptotic complexity is $O(nml)$.
2. **DO**: Order $\mathcal{R}$ in non-increasing order of size (execution time) and Ψ in non-decreasing order of execution cost per time unit. Use the FIFO algorithm to compute the scheduling. This is a straightforward adaptation of the Max-Min algorithm [13], focusing on the execution of requests with greater duration in VMs with lower cost. Its asymptotic complexity is $O(nml + n \log n + m \log m)$.

5 Experimental Results

We implemented the platform proposed in this work using JAVA, and the scheduler using IBM ILOG CPLEX with default configuration. The mapping model proposed in Fig. 4 was utilized in the scheduling. The simulations were run in a dual-processor Xeon Quad-Core with 32GB of RAM. The metrics utilized were the number of fulfilled requests and the scheduling cost.

The number of VMs in some cases can be insufficient to fulfill all requests, when the execution cost of the R set must be analyzed considering only the fulfilled requests. To minimize the execution cost and maximize the number of fulfilled requests, we verified the scheduling result when relaxing the maximum execution time of each request in FTRt and VTR categories. The experiments were conducted with the requested time extended by 10%, 30%, 50%, and 100%. We do not extend $d\alpha$ for FTRx requests since this would violate the QoS.

5.1 Simulation Setup

We evaluated the number of fulfilled requests, execution costs, and we also assessed the behavior of the scheduler in the platform when $d\alpha$ is varied. The scenario has four providers with a total of $m = 25$ VMs, with characteristics distributed as shown in Table 1.

Table 1. VMs configuration available to the PaaS users.

Provider	**OD**	$	**RE**	$	**SP**	$	**Provider**	**OD**	$	**RE**	$	**SP**	$
1	3	10.00	2	5.00	3	1.00	2	3	11.00	1	6.00	1	2.00
3	2	12.00	2	7.00	2	3.00	4	2	13.00	3	8.00	1	4.00

Three experiments, E_1, E_2, and E_3 were run, each one with 30 sets of requests, $E_i = \{R_1, R_2, ..., R_{30}\}$. $d\alpha$ was in the set $\{10\%, 30\%, 50\%, 100\%\}$. In each experiment, the number of requests in each set was E_1: $|R_j| = 30$, E_2: $|R_j| = 45$, E_3: $|R_j| = 60$, $1 \leq j \leq 30$. The $r(d)$ of each request in each set $R_j \in E_i$ is randomly taken from the $(1, 15)$ interval. The composition $r(d\alpha)$ is done by summing a value t_1 randomly taken from the same $(1, 15)$ interval: $r(d\alpha) \leftarrow r(d) + t_1$. The number of requests in the FTRx, FTRt, and VTR categories are, respectively, 20%, 40% e 40%. We defined $\mathcal{M} = 1.000$.

5.2 Results

We have run the simulations to compare the three approaches: ILP, FIFO, and DO. In the first experiment, E_1, the number of requests is close to the number of VMs available to the PaaS, while E_2 and E_3 have more requests than the number of VMs available. Figure 5 presents the results obtained by the three algorithms in the three experiments. The schedule cost found by the ILP is lower than the costs found by FIFO and DO. Moreover, the ILP was able to fulfill more requests than the other algorithms, satisfying the QoS of almost 100% of the requests with the original $d\alpha$.

Figures 5(g), 5(h) and 5(i), present the cost per fulfilled requests for the schedules found in experiments E_1 to E_3. The ILP results presented lower costs per request when compared to the FIFO and DO algorithms. In experiment E_1, the ILP reduced costs from 25% (0% relaxation) to 35% (100% relaxation) in relation to FIFO, and from 6% to 3% in relation to DO. This difference is enlarged with more requests, reaching 33% to 37% in relation to FIFO, and from 27% to 19% in relation to DO for experiment E_3 (60 requests).

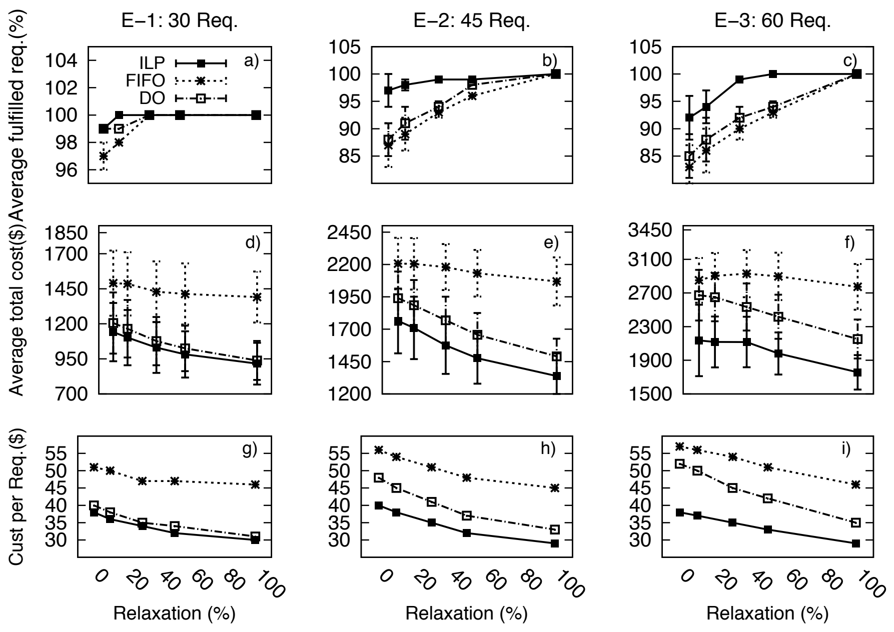


Fig. 5. Results for experiments E_1, E_2, and E_3.

Results showed in this section suggest that the presented charging model allows a better utilization of RE instances, reducing costs from the user perspective. Furthermore, the presented ILP was able to reduce costs and fulfill a higher number of requests than FIFO and DO algorithms in IaaS VMs within the PaaS proposal considering the TS charging model.

6 Conclusions and Future Work

We presented substantial contributions to compose a PaaS architecture and a platform to schedule requests on a set of leased VMs from IaaS providers. We proposed a mapping of currently existing charging models to three different quality of service requirements from the users requests, also introducing the *Time-Slotted Reservation* charging model. The objective of the proposal is to allow a better utilization of the leased VMs from a set of IaaS providers in order to reduce costs, avoiding the lease of new VMs and helping to determine if any leased VM contracts could be finished. Moreover, we propose an integer linear program (ILP) scheduler that considers the leased VMs, their costs, and the quality of service of requests to maximize the number of fulfilled requests and reduce costs. Indeed, the ILP presented better results than two heuristics used in the comparison, allowing more requests to be fulfilled without QoS violation with the use of the time-slotted reservation, but also reducing the cost per request.

As future work we consider the development of dynamic mappings between the two SLA levels according to existing charging models and their prices.

Acknowledgment

We would like to thank FAPESP, CNPq, and CAPES for the financial support.

References

1. Assunção, M.D., Costanzo, A., Buyya, R.: A cost-benefit analysis of using cloud computing to extend the capacity of clusters. Cluster Computing **13**(3) (2010) 335–347
2. Genez, T.A.L., Bittencourt, L.F., Madeira, E.R.M.: Workflow scheduling for SaaS / PaaS cloud providers considering two SLA levels. In: Network Operations and Management Symposium (NOMS), 2012 IEEE. (april 2012) 906 –912
3. Bittencourt, L.F., Senna, C.R., Madeira, E.R.M.: Scheduling service workflows for cost optimization in hybrid clouds. In: Proceedings of the International Conference on Network and Service Management (CNSM), 2010. (oct. 2010) 394 –397
4. Díaz Sánchez, F., Doumith, E., Al Zahr, S., Gagnaire, M.: An economic agent maximizing cloud provider revenues under a pay-as-you-book pricing model. In: Economics of Grids, Clouds, Systems, and Services. Volume 7714 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 29–45
5. Zhao, H., Yu, Z., Tiwari, S., Mao, X., Lee, K., Wolinsky, D., Li, X., Figueiredo, R.: Cloudbay: Enabling an online resource market place for open clouds. In: Proceedings of 5th IEEE/ACM International Conference on Utility and Cloud Computing. UCC 2012, Chicago, USA (2012)
6. Toosi, A.N., Thulasiram, R.K., Buyya, R.: Financial option market model for federated cloud environments. In: Proceedings of 5th IEEE/ACM International Conference on Utility and Cloud Computing. UCC 2012, Chicago, USA (2012)
7. Chen, J., Wang, C., Zhou, B.B., Sun, L., Lee, Y.C., Zomaya, A.Y.: Tradeoffs between profit and customer satisfaction for service provisioning in the cloud. In: Proceedings of the 20th international symposium on High performance distributed computing. HPDC '11, New York, NY, USA, ACM (2011) 229–238
8. Konstantinou, I., Floros, E., Koziris, N.: Public vs private cloud usage costs: the stratuslab case. In: Proceedings of the 2nd International Workshop on Cloud Computing Platforms. CloudCP '12, New York, NY, USA, ACM (2012) 3:1–3:6
9. Khajeh-Hosseini, A., Greenwood, D., Sommerville, I.: Cloud migration: A case study of migrating an enterprise it system to iaas. In: Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing. CLOUD '10, Washington, DC, USA, IEEE Computer Society (2010) 450–457
10. Garg, S., Venugopal, S., Buyya, R.: A meta-scheduler with auction based resource allocation for global grids. In: 14th IEEE International Conference on Parallel and Distributed Systems, 2008. ICPADS '08. (dec. 2008) 187 –194
11. Mihailescu, M., Teo, Y.M.: On economic and computational-efficient resource pricing in large distributed systems. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Washington, DC, USA (2010) 838–843

12. Raj, G.: An efficient broker cloud management system. In: Proceedings of the International Conference on Advances in Computing and Artificial Intelligence. ACAI '11, New York, NY, USA, ACM (2011) 72–76
13. Ibarra, O.H., Kim, C.E.: Heuristic algorithms for scheduling independent tasks on nonidentical processors. J. ACM **24**(2) (April 1977) 280–289