



SLA negotiation and enforcement policies for revenue maximization and client classification in cloud providers



Mario Macías*, Jordi Guitart

Barcelona Supercomputing Center/Technical University of Catalonia, Campus Nord, C6 Building, c/Jordi Girona 29, 08034 Barcelona, Spain

HIGHLIGHTS

- Providing a set of policies for maximizing BLOs achievement of a Cloud provider.
- Modeling of client classification as a complementary BLO to Revenue Maximization.
- Our rules use state-of-the-art cloud technologies: VM migration, elasticity, etc.
- Development of a fine-grain simulator for exhaustive validation of the model.

ARTICLE INFO

Article history:

Received 27 June 2012
Received in revised form
25 February 2014
Accepted 16 March 2014
Available online 24 March 2014

Keywords:

Cloud Computing
Business policies
Revenue Maximization
Client Classification
Business-level Objectives
Service-level Agreements
SLA enforcement
SLA negotiation

ABSTRACT

In Cloud Computing markets, owners of computing resources negotiate with their potential clients to sell computing power. The terms of the Quality of Service (QoS) to be provided as well as the economic conditions are established in a Service-Level Agreement (SLA). The common objective of a Cloud provider is to maximize its economic profit. However, there are situations in which providers must differentiate the SLAs with respect to the type of client that is willing to access the resources or the agreed QoS, e.g. when the hardware resources are shared between users of the company that own the resources and external users.

This article proposes two sets of policies to manage SLAs with respect to the business objectives of a Cloud provider: Revenue Maximization or classification of clients. The criterion to classify clients is established according to the relationship between client and provider (external user, internal or another privileged relationship) and the QoS that the client purchases (cheap contracts or extra QoS by paying an extra fee). The validity of the policies is demonstrated through exhaustive experiments.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

In recent years, the Utility Computing business model [1] is increasing its acceptance in the Information Technology sector thanks to the burst of Cloud Computing paradigm [2]. In Utility Computing, the users of the resources are not necessarily their owners: users run their applications or services in remote data centers and pay as a function of the usage. Utility Computing allows the users to economically benefit from economies of scale, because it minimizes the space and maintenance costs. Cloud Computing has emerged as a successful implementation of Utility Computing because virtualization allows to share physical resources transparently to the clients, by isolating different clients as if each client were using a dedicated physical host.

This article considers a Cloud Computing market in which providers and clients participate in a market to sell or buy services or resources. Although this market model is not common in current commercial Cloud providers, it is based on research proposals for creating a market place of computing resources [3–5] and could be potentially applicable in real scenarios. In such a market, a client starts a negotiation process with the providers that fulfill their requirements to establish the terms of the contract [6,7]: the client creates a proposal of agreement for each provider that meets its requirements and fills a Service-Level Agreement (SLA) template with its requirements. When the providers receive the SLA proposal, they evaluate it and return their SLA proposals, with economic information such as pricing, and terms of violations. Then, the client chooses the provider whose SLA suits best its interests. From this moment, both parties have reached an agreement, and the terms of the Quality of Service (QoS) to be provided and the economic conditions are kept in the SLA, which is shared between client and provider. If the SLA terms are not fulfilled when the client

* Corresponding author.

E-mail addresses: mario.macias@bsc.es, mario@ac.upc.edu (M. Macías), jordi.guitart@bsc.es, jguitart@ac.upc.edu (J. Guitart).

<http://dx.doi.org/10.1016/j.future.2014.03.004>
0167-739X/© 2014 Elsevier B.V. All rights reserved.

uses the provider's resources, the provider must pay a penalty to the client. The penalty is calculated according to the severity of the SLA violation, and could be influenced by other parameters, such as the provider–consumer relationship, framework contracts, or risk assessments.

The behavior of the provider is determined by its Business-Level Objectives (BLO), which are considered when the SLA is negotiated and enforced. For a Cloud provider, the maximization of economic profit is the most common BLO. This objective can be achieved by maximizing the price of the Cloud service, also by minimizing the economic impact caused by the penalties derived from possible violations of the agreed SLAs. However, Revenue Maximization is not the only BLO that a provider could be interested in.

Despite the economic benefits of using computing as a utility, there are still open security reasons to not submit the critical or confidential data to resources that are located in third parties [8]. For example, a company that stores sensible data about its clients may not be legally allowed to send this information to data centers belonging to a third company. These companies require an in-house infrastructure. Aiming for profitability, they may decide to hire out the spare resources of their data centers to external users that do not have such security or confidentiality restrictions, because the price that external clients pay to use the resources contributes to amortize the cost of the data centers. Thanks to virtualization, a company could provide VMs to external clients without putting its critical data on risk.

In such a scenario, a differentiation between internal and external users is advisable. However, a binary classification of the users as internal/external is not accurate enough in many situations. For example, headquarters of a big company can classify the users of its data centers according to different levels: users from the headquarters that own the resources are completely internal, users from other companies are completely external, and users from other headquarters of the same company are in a middle ground. Even multinationals could define more degrees of proximity for headquarters in the same country and headquarters in other countries. Whilst completely external users pay a fee and completely internal users use the resources for free, the mid-range users would pay a reduced fee to encourage each location to only use resources from external locations when strictly necessary.

Another example of intermediate users are those from trusted entities that decide to share their computing resources for sharing risks and dealing with peaks of workload without the need to overprovision resources. Examples of trusted entities are different companies from the same business cluster [9].

Clients can be classified according to other criteria. Many service providers classify their clients according to the QoS that they have purchased. For example, Spotify [10] is an online music provider that classifies its clients in three categories (*free*, *unlimited* and *premium*) according to their monthly fee. The higher the fee, the more the services and QoS: unlimited streaming hours, highest quality of sound, available downloads, etc. The provider must consider the purchased QoS when allocating the resources.

The usage of the resources by external users can affect the QoS of internal users if the SLAs do not reflect priorities between clients in terms of pricing or allocation of resources. Client Classification is applied to keep high QoS to internal users or users with high QoS requirements. Client Classification considers the information about the users when giving them access to the resources and prioritizes some SLAs according to two criteria:

QoS that the users are willing to acquire: the higher the QoS the higher the price. This is the traditional classification of services in Utility Computing.

Affinity between the client and the provider: clients from the same company as the provider or from entities that have a privileged relationship with the provider can hire the services at better

prices, better QoS, or any other privilege. This novel approach was devised with the success of Cluster and Grid Computing, in which organizations share part of their resources with users from other organizations. By prioritizing users to which there is high affinity, organizations can ensure that their internal users will have enough resources or QoS when there is a peak of external demand.

Achieving both Revenue Maximization and Client Classification BLOs can be difficult because of a limitation of current Cloud market models: the lack of communication between market and resource layers. Since the Resource Manager at the infrastructure layer does not have knowledge about the business details, it is difficult to define accurate policies or objectives to be fulfilled by the resources. Similarly, if a business broker negotiates the sale of resources without detailed information about them, this would potentially lead to waste or overload resources. To deal with this problem, we introduce the Economically Enhanced Resource Manager (EERM). The EERM enables bidirectional communication between Business and Resource layers to (1) manage resources and enforce SLAs for maximizing the BLOs and (2) support the negotiation of SLAs for adjusting them to the market status and the real capacity of the providers.

The main goal of this article is facing the problem of Cloud resource management from both business and performance sides, which are often conflicting. We provide an integral solution of several policies that work together for maximizing the achievement of the BLOs of a Cloud provider, dealing with performance issues. We describe and evaluate a set of policies for maximizing the BLOs: Price Maximization and Discrimination, Resources Overselling, Selective SLA Violation, Dynamic Scaling of Resources, and Runtime Migration of VMs. We demonstrate that they can be applied for maximizing two different BLOs: Revenue Maximization and Client Classification. Since both markets and the internal status of the providers change over time, we evaluate the introduced policies in terms of relative results and trends (e.g. using a certain policy you can generally increase the profit).

The experiments have been performed using the EERM simulator [11]; a fine-grained, customizable Cloud market simulator that triggers several business policies that are defined by users. Both the policies and the simulations are targeting the infrastructure layer within Cloud Computing (Infrastructure as a Service, IaaS). The validity of the policies is evaluated through simulations instead of real executions mainly because of two reasons: the first is the difficulty to acquire a testbed large enough to get representative data from the experiments; the second is that this article does not evaluate the performance, but the business benefits of using an EERM in Cloud Computing markets. These markets are experimental at this moment, and there are no traces currently available of real executions for their reproduction in experiments.

The rest of the article is structured as follows: after a summary of the related work, we introduce the SLA negotiation and enforcement model in Section 3. Section 4 describes the group of policies that have been used for enforcing both Revenue Maximization and Client Classification objectives: their motivation and their application. We demonstrate the validity of the policies in Section 5, which shows the results of the performed simulations. Finally, we expose the conclusions of our work and depict some future research.

2. Related work

There is a large body of work that considers Business-Oriented Resource Management. Yeo et al. [12] propose a model for market-based cluster computer with some elements common to EERM. The cluster nodes are connected to a central manager which incorporates other sub-components for performing pricing, job scheduling, monitoring, and dispatching. The main difference from our EERM is that EERM is focused on resource management, so it does

not implement some functions such as billing, or identity provisioning. Freedman et al. [13] focus on Peer-to-Peer (P2P) content distribution by identifying explicitly highly demanded files and rewarding most those peers that share highly demanded content. They use a market-based mechanism for allocating network resources across multiple files and play with the Law of Offer and Demand to incentivize providers to sell most scarce high-value resources (only files, but not CPUs or memory). Their system is also designed so that the client chooses files consistent with its best interests, because it seeks to download at the current minimum price. This article extends these policies for general purpose computing.

Amazon Spot Instances [14] prices as a function of the market status. Clients specify the maximum price they are willing to pay for an instance. The price of the spot instances fluctuates in function of the offer and demand. When the spot price is lower than the bid price, the instance is executed until the spot price is higher again. This paradigm would motivate clients to move their batch workloads to off-peak hours where the demand is low. However it is not suitable for Web Services that have real-time requirements during peak hours.

Püschel and Neumann [15] use the concept of an EERM for optimizing the revenue of a Cloud manager. They apply policies as a heuristic and demonstrate their achievements on Revenue Maximization or client classification when applying economic enhancements. This article tries to be a step forward in the usage of policies. It enables procedural policies that guide the actions of EERM under certain conditions to get the best solution to given problems. The authors in [16] present an upgraded EERM that adds more policies and performs a more intensive evaluation of their validity.

Aiber et al. [17] introduce an autonomic computing approach to business-oriented self-optimization of service providers. First, a particular scenario is modeled from both business and IT points of view, and the impact of IT on business and vice versa is studied. Next, business rules for continuous optimization of IT resources and business are defined for maximizing the business utility of the IT resources and maximize the return of investment of the infrastructures. We use a similar approach that differentiates business and IT layers and uses an EERM between them. The EERM contains rules for dealing with IT and business relationships and maximizing the business utility of the infrastructure.

There is some relevant work in rule-based resource management for distributed environments. Collaborative Awareness Management [18] promotes cooperation between resources for their optimization by means of a set of rules. Schiefer et al. [19] introduced a business rules management system that is able to sense and evaluate events to respond to changes in a business environment or customer needs. We extend these approaches by combining high-level service and business data with the low-level resource information, enforcing the flow of information between the two layers for their mutual optimization. Weng et al. [20] propose an autonomic management system of VMs that rely on policies. Their system is mainly oriented to guarantee the QoS of a pool of VMs by dynamically scaling the assignments of CPUs to each VM. Our article extends this approach by adding other reactive actions, such as migration of resources or cancelation of tasks, and is not only limited to guarantee QoS, but also business metrics.

Other related work introduced some policies, which are similar to the policies introduced in this article. Sulistio et al. [21] proposed overbooking strategies for mitigating the effects of cancellations and no-shows for increasing the revenue. The overbooking policies implemented in this article, in addition, consider the possibility of under-usage of the reserved resources from the client. Dube et al. [22] establish different ranges of prices for the same resource and analyze an optimization model for a small number of price classes. Their proposal is similar to the proposal of this article of establishing Gold, Silver and Bronze ranges and optimizing

their QoS performance giving priority to the contracts that report the highest economic profit. This article extends this work by combining the QoS ranges with several other policies, such as pricing or selective violation.

Menascé et al. [23] demonstrated the importance of managing the resources taking into account the BLOs. They maximize the revenue of e-commerce servers by dynamically calculating Customer Behavior Model Graphs and prioritizing the workloads of the users that are willing to spend more money. Poggi et al. [24] introduce a similar approach, in which QoS for user sessions are shaped based on the commercial intention of the users. However, these models are not applicable to the scenario of this article, because the Cloud provider supports more generic types of workloads, not restricted to online shops, and does not interact with human customers, but with other client machines that automatically buy resources in a market.

Previous work from the authors of this paper demonstrated that business-oriented resource management could lead to an increase of the profit and that management could be in execution time [25], or in negotiation time [6]. This article presents work which is a step forward from previous work by implementing and testing the validity of these business-oriented policies within an EERM with a rule engine [26] customizable by the system administrator, and by introducing new rules and policies, such as Selective SLA Violation or Resources Overselling.

Client Classification is a usual practice in many businesses, such as banking services [27]. These businesses categorize clients in function of their size, budget, etc. and establish policies that define clearly the priorities of the clients, their protection level, their assigned resources, QoS, etc. In Cloud Computing, Amazon Elastic Computing Cloud (EC2) provides a set of predefined VM instances [28], each one with different performance profiles (CPU load, memory, etc.), but a fixed QoS: they claim that their machines have an annual availability of 99.5%. With this paradigm, providers tend to overprovision resources for minimizing risks and provide high availability, which is not economically suitable for small providers. We try to channel the risk to the SLAs with the lowest priority according to the defined BLOs. In case of SLA violation, the clients will receive an economic compensation proportional to the seriousness of the violation.

Püschel et al. [29] propose a scheme for Client Classification by means of price discrimination, different priorities in job acceptance and differentiation in QoS. They adopt the architecture of an EERM. The EERM supports the optimization of SLA Negotiation and Management by dealing with both economic and technical information of Cloud Computing markets. In addition, this article extends the research of Püschel et al. in Client Classification with the extension and details of the policies, and deeper validation of them by means of a tailored simulation of clients, Cloud market, EERM, and resource fabrics.

3. Model for SLA negotiation and enforcement

A Cloud market has two main actors: clients and providers. When clients try to buy resources for hosting their services or applications, they send offers to providers to start a negotiation. Each provider owns a set of N physical machines. Each physical machine can host several VMs that execute tasks, such as Web Services or Batch Jobs. The SLA of a task is described as $SLA = \{Rev(vt), C, \vec{S}, \Delta t\}$:

- $Rev(vt)$ is a revenue function that describes how much money the provider earns after finishing correctly or incorrectly a task. The Violation Time (vt) is the amount of time in which the provider has not provided the agreed QoS to the client. Let MP be the Maximum Penalty (can be seen as negative revenue:

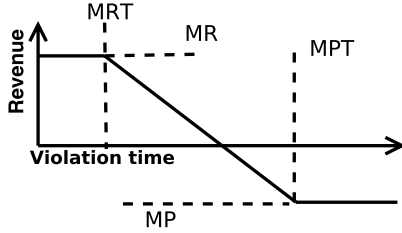


Fig. 1. Revenue of an SLA as a function of the violation time (Eq. (1)).

lower MP implies higher penalties), MR the Maximum Revenue, MPT the Maximum Penalty Threshold, and MRT the Maximum Revenue Threshold, Eq. (1) describes the revenue function. If $vt < MRT$ the SLA is not violated (0 violations); if $vt > MPT$, the SLA is completely violated (1 violations). $MPT > vt > MRT$ implies a partial violation ($\frac{vt-MRT}{MPT-MRT}$ violations).

$$Rev(vt) = \frac{MP - MR}{MPT - MRT} (vt - MRT) + MR. \quad (1)$$

This equation allows a grace period where the provider can violate the SLA without being penalized. When vt surpasses the MRT threshold, the revenue linearly decreases (see Fig. 1) in function of vt . The Maximum Penalty MP is defined for avoiding infinite penalties. Client and provider can negotiate the values of MRT , MR , MPT , MP for establishing different QoS ranges for the clients, which report different revenues and penalties for the providers [6].

- C_i is the client information. Let id be the client identifier and $\vec{CD}$ a vector that handles the description of the client, then $C = \{id, \vec{CD}\}$. The information contained in $\vec{CD}$ must be decided by the business administrator of the provider and applied consequently in the policies. An example of a template is $\vec{CD} = \{companyname, department, location, accessrights\}$. The provider could calculate the affinity to the client by comparing the client information with its own information (company, department, etc.).
- $\vec{S}$ describes the QoS of the purchased service: throughput, response time, and so on.
- Δt is the time period that is requested to allocate the task.

The revenue function $Rev(vt)$ (as well as all the revenue figures in the evaluation) subtracts the penalties from the incomes, so it indicates how profitable the allocation and execution of an SLA with a given set of policies is. However, it does not indicate the provider's net benefit because it does not consider other costs, such as infrastructure maintenance.

4. Business-driven SLA negotiation and enforcement

This article presents policies to autonomously maximize the achievement of the BLOs of a Cloud provider. This section presents two sets of policies classified by their BLO: Revenue Maximization and Client Classification.

To facilitate the reading of this article, the names of the policies are abbreviated according to the following notation $PolicyName^{BLO\ Name}$. $PolicyName$ is an abbreviation of the policy name. The abbreviations of all the policies are shown below, enclosed in parentheses next to their names. $BLO\ Name$ is an abbreviation of the Business-Level Objective that is pursued by the $PolicyName$. The actual abbreviations for the BLOs are: RM for Revenue Maximization, Aff for SLA classification as a function of the client affinity, and QoS for SLA classification as a function of the QoS. As example, $PrDsc^{Aff}$ is the Price Discrimination policy that applies discount to clients according to their affinity. We will refer $PolicyName^*$ to a given policy regardless of the $BLO\ Name$.

Table 1 provides a quick reference to each symbol referred in Sections 3 and 4.

Table 1
Definition of symbols used in Sections 3 and 4.

Symbol	Description
vt	Violation time
$Rev(vt)$	Revenue function
MP	Maximum Penalty
MPT	Maximum Penalty Threshold
MR	Maximum Revenue
MRT	Maximum Revenue Threshold
C_i	Client i description
$\vec{S}$	QoS terms of the purchased service
Δt	Time slot of a task
$u_p(SLA)$	Price utility for the provider
RP_s	Reservation Price of the seller
RP_b	Reservation Price of the buyer
$a(\Delta t)$	Aggressiveness factor
$R_{used}(t)$	Predicted usage of reserved resources
$R_{req}(t)$	Amount of resources requested by a client
$R_j(t)$	Total of resources in resource j
$R_j^{used}(t)$	Predicted usage of resources by other clients in resource j
δ	Time during which a VM would be selectively violated
aff	Client affinity
$P(C_i)$	Priority of client i (affinity-based or QoS-based)
Φ	Maximum % to (un)penalize clients according to $P(C_i)$

4.1. Policies for Revenue Maximization

Revenue Maximization is the main motivation of most Cloud Computing providers. This goal can be achieved by a combination of maximizing the incomes from service executions, minimizing the penalties due to SLA violations, and minimizing the costs of the infrastructure (e.g. energy consumption, amortization of hardware, etc.). In this article, we focus on maximizing the incomes and minimizing the penalties.

A provider may try to maximize the revenue from two points of view: *SLA Negotiation*, where the EERM must help the market agents to achieve the most profitable contracts for their BLOs; and *SLA Enforcement*, where the EERM must manage the resources for minimizing the economic penalties derived from the possible SLA Violations.

4.1.1. Price Maximization ($PrMax^{RM}$)

In a market, providers must establish an adaptable pricing policy that varies as a function of the offer/demand ratio. The objective is being attractive to the maximum number of clients [30]. Our Price Maximization policy is calculated as a function of two parameters: Price Utility and Aggressiveness factor.

Price utility: $u_p(SLA)$. When the provider establishes a price, it must know the range of prices where the agreement is possible. The *reservation price of the seller* (RP_s) is the minimum price that the seller can accept without losing money. The *reservation price of the buyer* (RP_b) is the maximum price that the buyer can pay and still get benefit from the acquired good. An agreement between buyer and seller is only possible when $RP_s \leq Price \leq RP_b$.

$$u_p(SLA) = \frac{Price - RP_s}{RP_b - RP_s}. \quad (2)$$

Eq. (2) defines the price utility. That means that the utility for the provider is high ($u_p(SLA) \rightarrow 1$) when the price tends to be RP_b . However, maximizing this utility function may not determine the price of a task: high prices will enforce clients to look for cheaper providers in the market.

The main issue of implementing this formula is to know the reservation price of the buyer, which only can be speculated in function of the market history.

Aggressiveness factor: $a(\Delta t)$. It calculates the percentage of resources that the provider has already reserved for a given time period Δt . A high value for $a(\Delta t)$ enforces the provider to establish

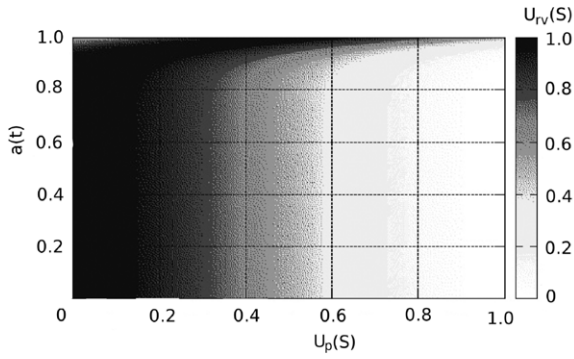


Fig. 2. Utility histogram as a function of the price and the aggressiveness factor for an SLA S .

higher prices because it is an indicator of a high demand ratio in the market. $a(\Delta t)$ may be calculated according to predictions based on historical data or other information, such as non-linear equation systems or machine learning algorithms.

Let $R_{used}(t)$ be a function that predicts the usage of the currently reserved bottleneck resources over time (for example, CPU or disk); let $R_{req}(t)$ be a constant function which represents the bottleneck resources requested by the client in the negotiation process (as a result of the decomposition of the set of QoS terms S); let $R_j(t)$ be a constant function that represents the amount of bottleneck resources of the physical resource j ; Eq. (3) shows how the aggressiveness factor $a(\Delta t)$ is calculated over a set of N physical machines.

$$a(\Delta t) = \frac{\sum_{j=1}^N \int_{t_i}^{t_f} R_{used}(t) + R_{req}(t) dt}{\sum_{j=1}^N \int_{t_i}^{t_f} R_j(t) dt}. \quad (3)$$

Maximizing the price entails maximizing a utility function $U_v(SLA)$ whose histogram pattern is shown in Fig. 2. In previous work [6], we checked that prices must tend to RP s in most scenarios, because clients are able to switch to providers that offer minimum prices. When the demand is very high (this is, $a(\Delta t) \rightarrow 1$), prices tend to RPb because the alternatives to find another provider are limited.

4.1.2. Resource Overselling ($Ovrs^{RM}$)

Clients tend to slightly overprovision the computing resources that they eventually use [31]. Although thanks to virtualization the resources are elastic and the overprovisioned proportion is low [32].

We propose to sell the computing capacity that has been already sold but the clients are not using: when a client negotiates an SLA and there are not enough resources to allocate it, the scoring function in Eq. (4) is calculated over the set $j = \{1 \dots N\}$ of physical machines. The physical resource j with the highest positive score is selected as candidate for executing the task and the $PrDsc$ policy is triggered for establishing a price. If there are no physical resources whose score is positive, the job is rejected.

$$score_j = 1 - \frac{\int_{t_i}^{t_f} R_j^{used}(t) + R_{req}(t) dt}{\int_{t_i}^{t_f} R_j(t) dt}. \quad (4)$$

The terms of Eq. (4) are described below:

- $R_{req}(t)$ is a constant function that represents the amount of bottleneck resources requested in the SLA that is being negotiated.
- $R_j(t)$ is a constant function that represents the total amount of bottleneck resources that are available in the physical resource j .

- Let $R_j^{used}(t)$ be a prediction of the total bottleneck resources used by the other clients in the physical resource j during the requested period.

The prediction of resource usage for a given resource ($R_j^{used}(t)$) can be calculated statistically or by Machine Learning algorithms [33] from the monitoring information. In this article, we use the CPU usage from resource monitoring because it is the bottleneck resource for the majority of services to be executed in the Cloud provider. Eq. (4) is abstract enough to allow using any other type of resource, such as memory or network bandwidth.

4.1.3. Selective violation of SLAs ($SLAViol^{RM}$)

When the previous policies are applied, a Cloud provider could oversell too many resources due to errors in workload estimation. Consequently, some SLAs would be violated indiscriminately. For minimizing the economic impact of such SLA violations, we propose to violate first the SLAs with lower penalties, or those that report less revenue [25].

We propose to violate first the SLAs that will report lowest revenue if fulfilled, or lowest penalties if violated. Their allocated VMs are paused temporarily (using virtualization facilities). For each SLA_i in a set of M SLAs that are being executed in the overloaded physical resource, the next formula is calculated:

$$NR(SLA_i) = \sum_{j=0}^{i-1} Rev(vt_j) + \sum_{j=i+1}^M Rev(vt_j) + Rev(vt_i + \delta). \quad (5)$$

In Eq. (5), δ is the time during which the SLA_i will be violated. The SLA whose $NR(SLA_i)$ value is the maximum will be violated during δ time.

As defined in Eq. (1), each SLA has a grace period (MRT). If $vt \leq MRT$, the violation will not involve any penalty. As a consequence, the system will tend to violate first the SLAs whose $vt \leq MRT$, or the SLAs whose economic penalty is low. This will lead to the distribution of the selective violations across the SLAs in grace period and keep a compromise between Revenue Maximization and QoS.

4.1.4. Dynamic scaling of resources ($DynScal^{RM}$)

$DynScal$ uses the potential of elasticity in virtualization: hardware resources are transparently reallocated within VMs at runtime [34]: when an SLA cannot be fulfilled, the EERM looks for VMs that are in the same physical machine and are not using all their allocated resources; then the EERM transfers the needed resources from the VMs with idle resources to the VM of the SLA that is not being fulfilled.

4.1.5. Runtime migration of tasks ($RtMigr^{RM}$)

Due to the heterogeneous nature of the Cloud, it is possible that the system becomes unbalanced and some physical resources in the pool are underutilized while others are overloaded. We propose Runtime Migration of Tasks to deal with this issue: when the load of a physical resource is over a given threshold (e.g. 90% of its total capacity), the system triggers the next rule:

Some VMs from the overloaded machine are randomly migrated to another machine with enough free space, if any. The process is stopped when the resource load is under the threshold or when there are no physical machines whose load is under the threshold.

Recent studies [35] reveal that the cost of migrating Web Services in Cloud Computing is near zero thanks to virtualization, because creating, booting, and populating a virtual machine with data takes few seconds (negligible in tasks that take from one to several hours). This policy could also be used for energy efficiency policies, by means of workload consolidation [36].

```

m ← overloaded physical resource that triggers the rule;
while  $systemLoad(m) \geq threshold$  do
  n ← physical resource with the lowest load in the
  resources pool;
  if  $m \neq n$  then
    Choose a random VM from  $m$  that would fit on  $n$ ;
    Migrate the chosen VM from  $m$  to  $n$ ;
  else
    Finish Rule;
  end
end

```

Algorithm 1: $RtMigr^{RM}$ policy

4.2. Policies for Client Classification

In addition to Revenue Maximization, another BLO is the classification of clients according to the *priority* that the provider assigns to them. This priority can be defined by considering two different criteria:

Client affinity: the affinity ($aff \subseteq [0, 1]$) measures how the client is related to the provider. For example, $aff = 1$ for a completely internal user; $aff = 0.25-0.75$ for a client from a company with privileged relationship with the provider (e.g. in the same business cluster); $aff = 0$ for a completely external client. The calculation of the affinity can be different among different providers, depending on their business goals. How affinity is calculated is not important in this article: the main topic is how to discriminate clients in function of their affinity.

Quality of service: the same Cloud provider can host critical tasks and tasks that can tolerate lower QoS. For example, e-commerce applications can need extra QoS guarantees to avoid losing money on service unavailability. It is reasonable to allow critical clients to buy extra QoS guarantees at higher prices, and keep cheap prices (but fewer QoS guarantees) for non-critical tasks. The different ranges of QoS are defined by establishing different values for MRT , MR , MPT and MP in $Rev(vt)$ (Eq. (1)). We define three ranges of QoS, in descending order: Gold, Silver, and Bronze. The higher the QoS range, the higher the MR and the lower the MP , MRT and MPT (lower values of these three values imply higher penalties).

As in Revenue Maximization scenarios, the application of the policies for Client Classification can be performed from both SLA negotiation and SLA enforcement points of view. Actually, the policies for Client Classification are similar to the policies for Revenue Maximization, but considering the client priority as a first classification criteria and the maximization of the revenue as a secondary objective.

4.2.1. Price discrimination ($PrDsc^{Aff}$)

The $PrDsc^{Aff}$ policy is built on top of the $PrMax^{RM}$ policy: after calculating the best resource allocation for maximizing the economic profit (see Section 4.1.1), the calculated revenue is multiplied by $(1 - affinity)$. This allows users with some affinity to receive a discount that is proportional to their affinity. This policy combines Client Classification with Revenue Maximization as a secondary BLO and always considers affinity as the main priority. Multiplying price by $(1 - affinity)$ will linearly prioritize users (a user whose affinity is 1 will have the double of priority than a user whose affinity is 0.5). However, other distributions such as $(1 - affinity^2)$ could be considered depending on the provider policies.

The $PrDsc^{QoS}$ policy is not considered because it would not make sense: Gold tasks must not be cheaper than Silver tasks, and Silver tasks must not be cheaper than Bronze tasks.

4.2.2. Overselling of resources ($Ovrs^{Aff}$ and $Ovrs^{QoS}$)

$Ovrs^{Aff}$ and $Ovrs^{QoS}$ are built on top of $Ovrs^{RM}$: the provider sells computing capacity that has been already sold, assuming that not

all the clients will use always 100% of their allocated resources. In this way, the provider can allocate more resources than its actual capacity.

The difference with $Ovrs^{RM}$ is the scoring function that determines if a physical resource is suitable for allocating an oversold resource (see Eq. (6)). Recall that the physical resource j with the highest positive score will be selected as candidate for executing the task.

The calculation of the scoring function is updated so that the prediction of the used resources in the scored physical resource is artificially increased when the priority of the client that negotiates the SLA is low and artificially decreased when the priority is high. Let $\phi \in [0, 1]$ be the maximum multiplier to penalize or unpenalize the predicted resource usage as a function of the client priority P . The priority-corrected prediction is defined as $(1 + \phi - 2\phi P)R_j^{used}(t)$.

$$score_j = 1 - \frac{\int_{t_i}^{t_f} (1 + \phi - 2\phi P)R_j^{used}(t) + R_{req}(t) dt}{\int_{t_i}^{t_f} R_j(t) dt}. \quad (6)$$

The adjustment made to $R_j^{used}(t)$ will motivate a higher acceptance of clients to which the provider has high affinity, because physical resources will appear emptier than they really are. As example, $\phi = 0.4$ used in the experiments will bias the prediction of a completely affine client to be 40% lesser than the actual one, and 40% higher for those clients to which there is no affinity. This way the admission rate of clients to which there is high affinity will be higher than the rate for clients to which there is low affinity.

This value is only chosen for showing the tendency of the graphs. Higher values of ϕ would decrease the revenue and increase both the average affinity and the number of SLA violations. Lower values of ϕ would have the opposite effect.

4.2.3. Selective violation of SLAs ($SLAViol^{Aff}$ and $SLAViol^{QoS}$)

Analogously to $SLAViol^{RM}$, we propose to violate first the SLAs of clients with low priority: to pause temporarily the tasks of the clients to which there is low priority (using virtualization facilities). For each SLA_i in a set of M SLAs that are being executed in the overloaded physical resource, the following formula is calculated:

$$NR(SLA_i) = \sum_{j=0}^{i-1} Rev(vt_j) * P(C_j) + \sum_{j=i+1}^M Rev(vt_j) * P(C_j) + Rev(vt_i + \delta) * P(C_i). \quad (7)$$

In Eq. (7), $P(C_i)$ is the value of the affinity or the QoS of the client C_i and δ is the time during which the SLA_i will be violated. The SLA whose $NR(SLA_i)$ value is the maximum will be violated during δ time.

Eq. (7) considers both revenue and client priority for choosing the SLA to be violated, because it would not be a good idea, for example, to cancel an SLA that reports a big revenue only because its associated client has 2% less priority than clients that own other SLAs with very low revenue or penalty.

4.2.4. Dynamic scaling of resources ($DynScl^{Aff}$ and $DynScl^{QoS}$)

This policy extends $DynScl^{RM}$ by adding the client priority to artificially bias the monitoring data of the resources. The resources needed for enforcing SLAs from low-priority clients are artificially decreased by a percentage that is linearly proportional to their priority. The resources needed for enforcing SLAs from high-priority clients are artificially increased by a percentage that is linearly proportional to their priority. Consequently, the system will act as if low-priority VMs tend to have free resources to transfer to high-priority VMs.

Table 2
List of figures in the experimentation section.

Number	Description
4	Revenue and average affinity when using $PrMax^{RM}$ and $PrDsc^{Aff}$ policies
5	Revenue and average affinity when using different $Ovrs$ policies
6	Number of violations and their affinity when using different $Ovrs$ policies
7	Proportion of violations by QoS range when using $Ovrs^{QoS}$
8	Revenue and average affinity of violations for $SLAViol^*$
9	Proportion of violations by QoS range with $SLAViol^{QoS}$
10	Revenue and average affinity of violations for $DynScal^*$
11	Proportion of violations by QoS range with $DynScal^{QoS}$
12	Revenue when triggering different $RtMigr$ policies
13	Proportion of violations by QoS range with $RtMigr^{QoS}$

Table 3
Values of the parameters for the simulations.

Parameter	Value (s)
ϕ	0.4
QoS ranges	{Gold, Silver, Bronze}
Maximum price for Gold	1.5 * maximum price for Silver
Maximum price for Silver	1.2 * maximum price for Bronze
Clients asking for Gold QoS	16.7%
Clients asking for Silver QoS	33.3%
Clients asking for Bronze QoS	50%
Distribution of the affinity between provider and clients	>0 for 25% of clients, =0 for the rest
Average affinity between provider and clients	0.21

4.2.5. Runtime migration of tasks ($RtMigr^{Aff}$ and $RtMigr^{QoS}$)

If $RtMigr^{RM}$ deals with the fact that the workload is unbalanced across the resource pool, $RtMigr^{Aff}$ and $RtMigr^{QoS}$ policies also deal with the fact that client priorities are unbalanced across the resource pool: if a physical machine is executing many tasks to which there is high priority and $SLAViol$ policy is triggered, a high priority task could be violated even if there are low priority tasks in other physical resources. Balancing the client priority across machines will minimize the aforementioned situations.

Analogously to $DynScal^{Aff}$ and $DynScal^{QoS}$, the $RtMigr^{Aff}$ and $RtMigr^{QoS}$ policies bias linearly the monitoring data from $RtMigr^{RM}$ (see Algorithm 1) for prioritizing the migration of high-priority VMs to physical resources where the average priority of VMs is low.

5. Evaluation

This section described the simulation environment as well as the results of the simulations for each policy introduced in the previous section. As a direct reference, Table 2 shows a summary of the figures that show the results of the simulation for the different policies that are introduced in this article.

5.1. Simulation environment

This section describes the experimental environment and its configuration values. We have used the EERM Simulator [11] to execute and evaluate the policies that are introduced in this article. The EERM Simulator is a fine-grained Cloud market simulator that simulates the complete cycle of our Cloud SLA negotiation and enforcement model: service discovery, SLA negotiation between provider and client, execution of Web Services or batch jobs and monitoring of the resources. It supports many features of Cloud Computing, such as elasticity of resources or migration of VMs. The EERM Simulator integrates the Drools [26] Rule Engine to allow configuring the SLA management policies with respect to the BLOs (e.g. the policies described in this article). We have used a simulated environment because it allows generating more data with limited resources in short time. This will allow to evaluate the models more accurately.

The simulated data centers are sized to represent a small-medium company that wants to externalize part of its resources for quicker amortization of the infrastructure costs. However, our research also focuses on big data centers that rent their space to companies. The policies are scalable for both types of providers, because they are applied at the level of each individual hardware node, without data or message dependencies between nodes.

The constant values and the parameters of the simulation described are arbitrary because there are no real market traces to extract data from. Different real market scenarios could require different values, but the contribution of this article is to show how Client Classification reports benefit *qualitatively*, not quantitatively. In other words, this article shows how a given policy can increase the revenue or improve the QoS to the preferential clients; this article does not intend to show whether the constant values in the policies are optimal, because they would vary depending on the real market status. In our future work, the provider will automatically adjust its parameters for self-adapting to changing market environments. The observed trends are more important than the specific values. The rest of this Section 5.1 explains the chosen values for the aforementioned parameters, which are also summarized in Table 3.

In the market, clients try to buy resources to host their Web Services. They send resource requests that contain $\{QoS, C, \vec{S}, \Delta t\}$, in which $QoS = \{Gold, Silver, Bronze\}$. For the same task in equal time and load conditions, the maximum price that the client is willing to pay for Gold QoS is 50% higher than for Silver QoS and 80% higher than for Bronze QoS.

The Web workload pattern corresponds to the access log of an ISP within our university domain¹ (see Fig. 3), and varies in function of the hour of the day and the day of the week: there are more requests from Monday to Friday evening than during the late night or the weekend. However, the proportion of QoS and Affinity of the clients does not vary in time.

Every provider belongs to a different organization, and has an affinity higher than 0 for 25% of the clients in the market, and equal

¹ The exact data source cannot be disclosed due to confidentiality reasons.

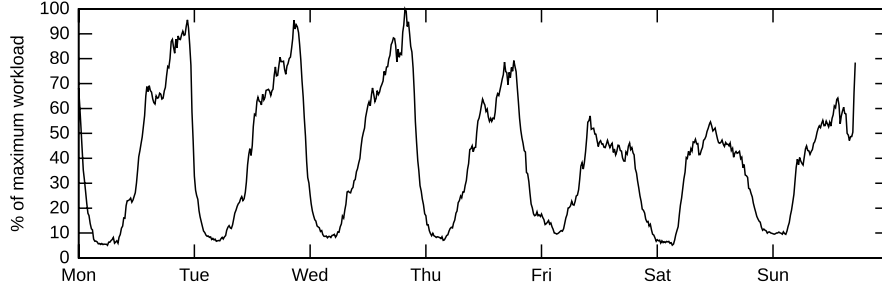
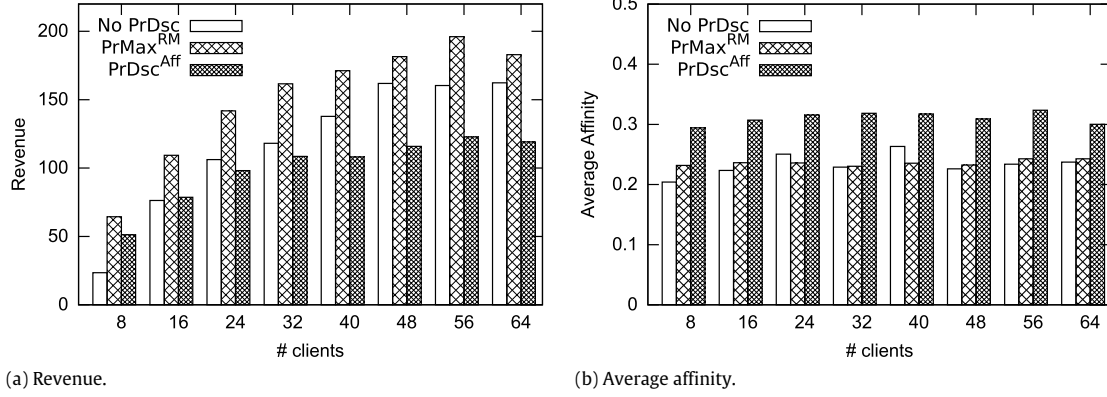


Fig. 3. Sample web workload pattern.

Fig. 4. Revenue and average affinity when using $PrMax^{RM}$ and $PrDsc^{Aff}$ policies.

to 0 for 75% of clients. The affinity of the clients ranges from 0 to 1 with a uniform distribution. Summarizing, the average affinity of all the clients is ~ 0.21 for every provider. Each client asks for Gold, Silver or Bronze QoS, independently of their organization. 1/6 of the clients ask for Gold QoS, 2/6 ask for Silver QoS, and 3/6 ask for Bronze QoS.

When the provider checks the request from the client, it applies Machine Learning techniques (Linear Regression, M5P, REPTree and/or Bagging) [33] to predict future workloads and verify whether the offered job can be executed correctly. A bad prediction could entail a violation of the SLA. The providers that accept the request return a revenue function $Rev(vt)$, which specifies the prices and penalties to pay for the execution of the service. Finally, the client chooses the provider with the lowest price or best time schedule for its interests, and sends back a confirmation.

The rest of this section demonstrates the validity of the policies introduced in Section 4. In each subsection, policies are incrementally added to the EERM rules repository in the same order as they were explained in Section 4. For demonstrating its validity, they are simulated in a scenario in which four different Cloud providers sell their services in a market during a week. Each provider has its own characteristics:

1. A provider that executes all the policies simulated so far. It prioritizes users to which the provider has high affinity.
2. Same as provider 1, but prioritizing tasks with high QoS.
3. Same as providers 1 and 2, but excluding the policy that is being introduced in the corresponding subsection. It is used for comparison purposes.
4. A provider that executes all the policies simulated so far, with client affinity as the objective. This formula is used instead of applying all the policies based on Revenue Maximization by comparative purposes, to show the real benefits of the introduced RM policy when compared with *Aff* policies.

It is important to evaluate how the providers behave and how effective the policies are in different scenarios. For example, if there are many providers and few clients, the prices and the load of the system will be low; if there are too many clients and the providers cannot host all of them, prices and the system workload will be high. To evaluate the policies in all the scenarios, all the experiments are repeated with different offer/demand ratios.

5.2. Experimental results

5.2.1. Price maximization and discrimination ($PrMax^{RM}$ and $PrDsc^{Aff}$)

Fig. 4(a) compares the revenue of the providers that are competing in the market. Every provider has different policies for pricing: *NoPrDsc* policy (which implements a fixed-pricing policy), $PrMax^{RM}$, and $PrDsc^{Aff}$. The $PrDsc^{QoS}$ policy is not considered because it does not make sense: Gold tasks must not be cheaper than Silver tasks, and Silver tasks must not be cheaper than Bronze tasks.

The x axis shows the number of clients in each experiment, which varies for showing the performance of the policies in different offer/demand ratios. The y axis represents the revenue of the different providers. Each column group represents the obtained results of the providers in the experiments.

Fig. 4(a) shows that, as expected, revenue is noticeably increased for the provider that applies $PrMax^{RM}$, compared to the provider that applies *NoPrDsc*. In the case of $PrDsc^{Aff}$ -provider, the revenue is noticeably decreased if compared with fixed-pricing and Revenue Maximization providers. It is demonstrated that the increment of the average affinity of the clients of $PrDsc^{Aff}$ penalizes the revenue. The need for compensating the impact in revenue of $PrDsc^{Aff}$ justifies the application of *Ovrs** policies.

Fig. 4(b) is structured similarly to Fig. 4(a), but its y axis shows the average affinity of the clients that used each resource. The average affinity is the addition of the affinities of all clients divided by the number of clients. The figure shows that the provider that

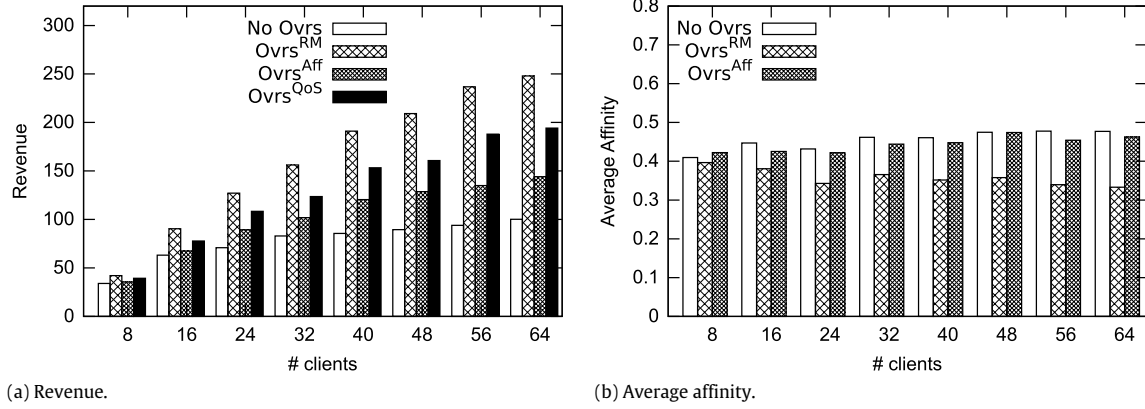


Fig. 5. Revenue and average affinity when using different Ovr policies.

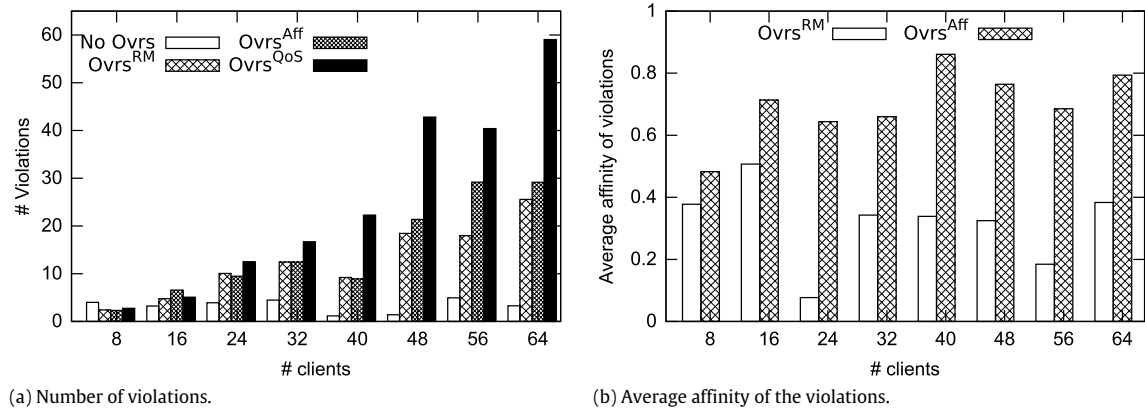


Fig. 6. Number of violations and their affinity when using different Ovr policies.

implements $PrDsc^{Aff}$ increases the average affinity of its clients up to 50%, compared to the other providers. The average affinity of clients in providers without $PrDsc^{Aff}$ is almost the same as the average affinity of all the clients in the market (~ 0.21).

5.2.2. Resource overselling (Ovr*)

Fig. 5 has a similar structure as Fig. 4: it shows both the revenue and the average affinity according to the policy combination in the provider.

Fig. 5(a) compares the revenue of the four providers described in Section 5.1: the provider labeled as *No Ovr* applies $PrDsc^{Aff}$ but does not apply any overselling policy; the other providers apply their respective dynamic pricing policies, as well as overselling policies based on Revenue Maximization (Ovr^{RM}), affinity discrimination (Ovr^{Aff}), and QoS range (Ovr^{QoS}), respectively. Fig. 5(a) shows that all the overselling policies have a positive impact on earnings. The provider labeled as *No Ovr* is the lower bound and the provider labeled as Ovr^{RM} is the higher bound. Ovr^{Aff} and Ovr^{QoS} stay in the middle of both: the clients are classified without renouncing the revenue completely. The revenue with Ovr^{Aff} is lower than the revenue with Ovr^{QoS} because Ovr^{QoS} prioritizes the Gold and Silver contracts, which report more revenue than Bronze ones.

To compare the usefulness of overselling based on affinity discrimination (Ovr^{Aff}), Fig. 5(b) also includes the results of a provider that performs $PrDsc^{Aff}$, but its overselling policy is driven by revenue instead of affinity (labeled as Ovr^{RM}). As the intention of Ovr^{QoS} is not to attract clients to which the provider has high affinity, this policy is not included in the figure. Fig. 5(b) shows that not considering the client affinity in the overselling policy decreases

the average affinity of the clients. It is not caused by any type of penalization, but it is a statistical fact: more clients enter the system, regardless of their affinity. Ovr^{Aff} maintains similar affinity levels as those of *No Ovr* but increasing the revenue of the provider, as in Fig. 5(a).

The drawback of overselling is that it considerably increases the number of SLA violations (Fig. 6(a)). There are two reasons: the associated error to the predictor component, and the permissiveness in terms of workload with clients to which the provider has high affinity. The latter makes the provider to violate a highest proportion of SLAs of high-affinity clients (Fig. 6(b)). The provider that applies Ovr^{QoS} reports the highest number of violations because the Gold and Silver SLAs have stricter requirements, which are more difficult to fulfill. Despite the increase in the number of violations, the proportion of violated SLAs over the total of allocated SLAs remains below 2% in the worst case.

Fig. 7 is a stacked chart that shows the percentage of each QoS ranges from the total of violations for the Ovr^{QoS} provider in several market simulations with different number of clients. It shows that the higher the QoS rank, the higher the percentage of violated SLAs. More violations of high-QoS SLAs do not mean that the QoS for Gold SLAs is lower than the QoS for Silver SLAs. It means that achieving the QoS requirements of Gold SLAs is difficult because the QoS requirements are high.

The drawbacks of Ovr^* can be minimized by applying the remainder policies for SLA enforcement at runtime: *SLAViol**, *DynScal** and *RtMigr**.

5.2.3. Selective violation of SLAs (SLAViol*)

The experiments performed in this section show only the results of simulations from 32 to 64 clients, because the load of the

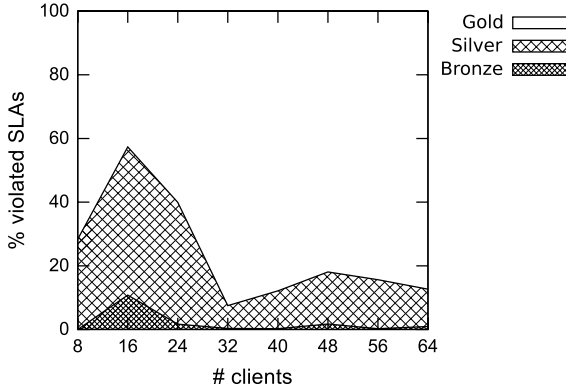


Fig. 7. Proportion of violations by QoS range when using $OvrS^{QoS}$.

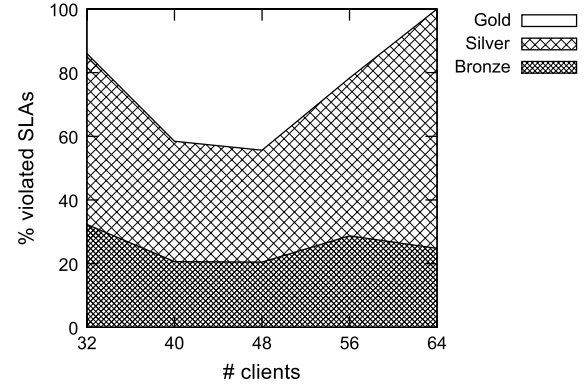


Fig. 9. Percentage of violations by QoS range with $SLAViol^{QoS}$.

system starts to be high from 32 clients, and the number of violations will be high enough to be representative.

The results of the experiments support the validity of all the $SLAViol^*$ policies. Fig. 8(a) shows that the provider that applies $SLAViol^{RM}$ increases the revenue up to 90% with respect to the other providers.

Fig. 8(b) compares the average affinity of the SLA violations of two providers. The average affinity here is the addition of the affinities of all the violated SLAs divided by the number of SLA violations. Both apply $PrDsc^{Aff}$ and $OvrS^{Aff}$, but one provider applies $SLAViol^{Aff}$ and the other does not. The figure shows that the average affinity of violations of the provider that applies $SLAViol^{Aff}$ is ~30%–50% compared to the provider that does not apply it, because $SLAViol^{Aff}$ violates SLAs from clients to which there is low affinity.

The results of prioritizing by QoS range are shown in Fig. 9: the proportion of high-QoS SLAs that is violated is considerably reduced with $SLAViol^{QoS}$, when compared with only applying $OvrS^{QoS}$ (Fig. 7).

5.2.4. Dynamic scaling of resources ($DynSca^*$)

Fig. 10(a) shows that the provider that applies $DynSca^{RM}$ increases its revenue by ~30%–100% with respect to the providers that do not apply this policy. Although Revenue Maximization is not the main BLO of $DynSca^{QoS}$, the revenue of that provider is also noticeably increased, because the number of SLA violations for Gold SLAs is reduced and, in consequence, the provider must pay less penalties.

Fig. 10(b) shows that $DynSca^{Aff}$ reduces the average affinity of the violations especially in scenarios in which the system load is not high. The reason is that there is not much leeway for finding free resources in high-load scenarios, even when the monitoring results are deviated to enforce Client Classification. Fig. 11, if

compared with Fig. 9, shows that the number of Gold SLAs that are violated is reduced by 50% when applying $DynSca^{QoS}$.

5.2.5. Runtime migration of tasks ($RtMigr^*$)

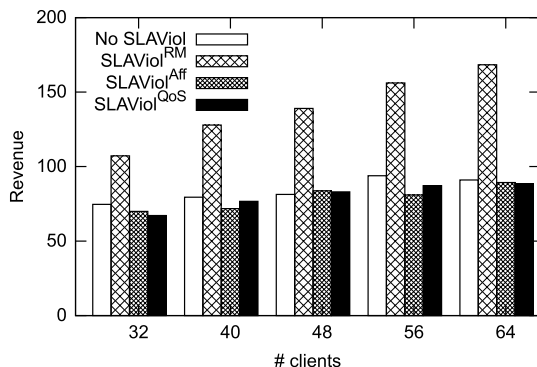
Fig. 12 shows that providers that apply $RtMigr^{RM}$ and $RtMigr^{QoS}$ increase their revenue compared to providers that do not apply them. The gain is proportionally similar to previous policies because the percentage of tasks with high revenue and penalties gets more balanced in all the physical resources.

$RtMigr$ policy combines well with $SLAViol$. Even if the migration of a task could simply transfer the violation to the target machine, the key fact is that $RtMigr$ diversifies the priorities of the tasks in every physical machine. In consequence, it is easy to find SLAs with low revenue in every overloaded machine. Otherwise, if $RtMigr$ was not applied there would be physical machines that only execute high-revenue tasks and the EERM would violate one of them even if there were low-revenue SLAs in the other resources. The same principle can be applied to client affinity and QoS.

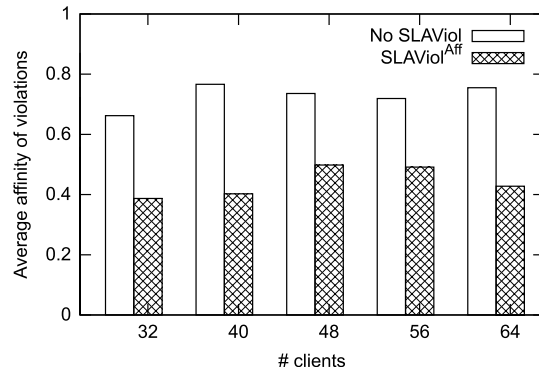
The effects of $RtMigr$ vary when applying $RtMigr^{Aff}$ or $RtMigr^{QoS}$. The average affinity of the violations is only reduced about 3%–5% on average in all the scenarios (so the figure that shows it has not been considered relevant to be included). The more the policies already implemented in the provider the less the percentage of improvement from new policies.

Fig. 13 shows the notable effects of live migration when prioritizing high-QoS tasks: the violations of Gold tasks are almost non-existent, and Silver violations are reduced. Fig. 13 also shows that the more the clients, the lower the effects of runtime migration. The reason is that migrated tasks would be also violated in the target machines because of the high load of the resources.

The specific value for the threshold over which $RtMigr$ is triggered (90% in the experiments) is not important in this work,



(a) Revenue.



(b) Average affinity of violations.

Fig. 8. Revenue and average affinity of violations for $SLAViol^*$.

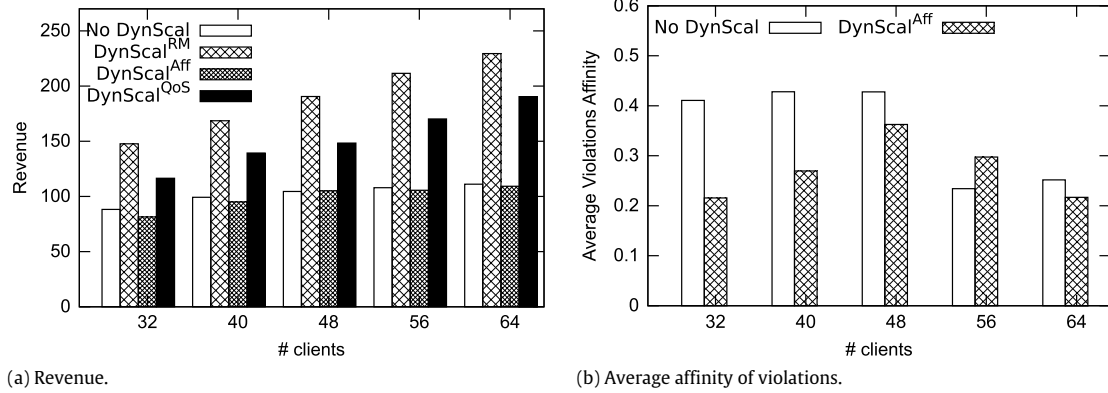


Fig. 10. Revenue and average affinity of violations for DynScaI*.

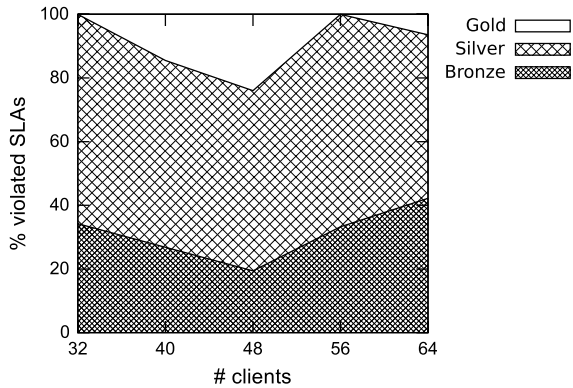


Fig. 11. Percentage of violations by QoS range with DynScaI^{QoS}.

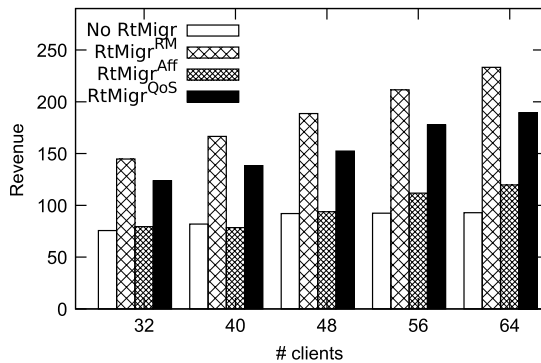


Fig. 12. Revenue when triggering different RtMigr policies.



Fig. 13. % of violations by QoS range with RtMigr^{QoS}.

because we want to show that applying *RtMigr* decreases the violations of high-priority SLAs. Finding the optimum value for this threshold autonomously is part of our future work.

6. Conclusions and future work

This article has introduced a set of policies that can be used for maximizing the achievement rate of the BLOs of a Cloud provider. Two BLOs have been considered: Revenue Maximization and Client Classification. Two facets can be used to classify the clients: client affinity and QoS.

First, we introduced a set of policies for Revenue Maximization: Price Maximization and Resources Overselling, which are triggered when the SLA is negotiated; Selective Violation of SLAs, Dynamic Scaling of Resources and Runtime Migration of SLAs, which are triggered when the VMs are running and the SLAs are enforced. The set of policies for Client Classification is built by modifying the Revenue Maximization policies for considering the client priority as a main objective; Revenue Maximization is kept as a secondary objective.

Revenue Maximization is suitable for most Cloud providers, because this is a common objective for ensuring the sustainability of businesses. Classification by QoS is suitable for a pure Cloud provider whose business is only based on selling its resources (it does not use them for its internal applications) and needs to provide different levels of QoS to gain maximum range of potential users, because it targets three types of clients: clients who need high QoS and are willing to pay high prices for that (because they may suffer economic losses), clients who do not have such QoS constraints and prefer to get cheaper resources with an acceptable rate of availability (for example, some Web pages for the general public), and an intermediate type of client (medium/high QoS, medium/cheap price). Classification by affinity is more suitable for organizations that host their own applications but want to sell part of their spare resources to amortize the cost of the infrastructure faster.

From experimental results, we conclude that the objectives of the policies are achieved: providers multiply their revenue when they apply Revenue Maximization rules; the percentage of priority users is increased when applying Client Classification policies. Also the rate of SLA violations is lower for users to whom there is high priority when applying Client Classification.

It is important to consider the introduced policies as a whole set. A Cloud provider should not consider applying only a subset of them because some policies have many drawbacks when triggered individually: for example, *PrDsc^{Aff}* reduces revenue and *Ovrs** increases the violations of SLAs. Applying the complete set of rules allows maximizing the benefits while minimizing the drawbacks.

The policies presented in this article rely on some constant values that could not lead to the optimal achievement of the BLOs. It is not the main objective in this article: the goal is to show that the policies increment the metrics of the BLOs. Finding the optimal values of the aforementioned constants brings a research opportunity for future work: adding dynamism to rules for allowing them to self-adapt at runtime to the changes in the environment and achieve the optimal results according to their own BLOs.

The objective of this article is not achieving the optimal welfare of the whole market. This article considers Cloud markets as a decentralized structure in which actors (clients and providers) may pursue different BLOs. We focus on the maximization of some BLOs from the viewpoint of each individual rather than finding a hypothetical scenario in which all the participants collaborate to attain a global optimal solution. This last scenario would be feasible in systems belonging to a single organization, but unrealistic in a market.

This article also intends to justify the roadmap for future implementations of Cloud markets. Current Cloud providers offer simple SLAs with few QoS and revenue options. The possibility of defining fine-grained SLAs would allow both clients and providers to mutually benefit: clients would find services that fit their requirements with more precision; providers would be able to apply policies that support a wider range of BLOs. This involves also changes in the Cloud market infrastructure: to set up neutral components to settle the SLA negotiation and enforcement process and guarantee trust in the market [4].

In our future research, we will investigate the validity of the rules in greater detail, by executing the tests in real platforms for checking how runtime migrations or dynamic scaling of resources behave in current virtualization systems.

Acknowledgments

This work is supported by the Ministry of Science and Technology of Spain under contract TIN2012-34557, by the European Union (FEDER funds) under contract TIN2012-34557, and by the Generalitat de Catalunya under contract 2009-SGR-980.

References

- [1] M.A. Rappa, The utility business model and the future of computing services, *IBM Syst. J.* 43 (2004) 32–42.
- [2] R. Buyya, C.S. Yeo, S. Venugopal, Market-oriented cloud computing: vision, hype, and reality for delivering it services as computing utilities, in: 10th IEEE Intl. Conf. on High Performance Computing and Communications, HPCC 2008, IEEE Computer Society, Dalian, China, 2008, pp. 5–13.
- [3] J. Altmann, C. Courcoubetis, G.D. Stamoulis, M. Dramitinos, T. Rayna, M. Risch, C. Bannink, Gridecon: a market place for computing resources, in: 5th International Workshop on Grid Economics and Business Models, GECON'08, Las Palmas, Spain, pp. 185–196.
- [4] D. Neumann, J. Stoesser, A. Anandasivam, N. Borissov, SORMA—building an open grid market for grid resource allocation, in: 4th International Workshop on Grid Economics and Business Models, GECON'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 194–200.
- [5] S. Garg, C. Vecchiola, R. Buyya, Mandi: a market exchange for trading utility and cloud computing services, *J. Supercomput.* 64 (2011) 1–22.
- [6] M. Macías, J. Guitart, Using resource-level information into nonadditive negotiation models for cloud market environments, in: 12th IEEE/IFIP Network Operations and Management Symposium, NOMS'10, Osaka, Japan, pp. 325–332.
- [7] M. Macías, O. Fito, J. Guitart, Rule-based SLA management for revenue maximisation in cloud computing markets, in: 2010 Intl. Conf. of Network and Service Management, CNSM'10, Niagara Falls, Canada, pp. 354–357.
- [8] L.M. Kaufman, Data security in the world of cloud computing, *IEEE Secur. Priv.* 7 (2009) 61–64.
- [9] M.E. Porter, Clusters and the new economics of competition, *Harv. Bus. Rev.* 76 (1998) 77–90.
- [10] Spotify, 2014. <http://www.spotify.com>.
- [11] Economically Enhanced Resource Manager, 2014. <http://www.sf.net/projects/erm>.
- [12] C.S. Yeo, R. Buyya, A taxonomy of market-based resource management systems for utility-driven cluster computing, *Softw. Pract. Exp.* 36 (2006) 1381–1419.
- [13] M.J. Freedman, C. Aperjis, R. Johari, Prices are right: managing resources and incentives in peer-assisted content distribution, in: Proc. 7th International Workshop on Peer-to-Peer Systems, IPTPS'08, Tampa Bay, FL, pp. 18–23.
- [14] S. Wee, Debunking real-time pricing in cloud computing, in: 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing 2011, Newport Beach, CA, USA, pp. 585–590.
- [15] T. Puschel, D. Neumann, Management of cloud infrastructures: policy-based revenue optimization, in: International Conference on Information Systems, ICIS 2009, Phoenix, Arizona, pp. 178–193.
- [16] M. Macías, J. Guitart, Client classification policies for SLA negotiation and allocation in shared cloud datacenters, in: Lecture Notes on Computer Sciences, LNCS, vol. 7150, 2011, pp. 90–104.
- [17] S. Aiber, D. Gilat, A. Landau, A. Sela, Autonomic self-optimization according to business objectives, in: Proceedings of the First International Conference on Autonomic Computing, IEEE Computer Society, Washington, DC, USA, 2004, pp. 206–213.
- [18] P. Herrero, J.L. Bosque, M. Salvadores, M.S. Perez, A rule based resources management for collaborative grid environments, *Int. J. Internet Protoc. Technol.* 3 (2008) 35–45.
- [19] J. Schiefer, S. Rozsnyai, C. Rauscher, G. Saurer, Event-driven rules for sensing and responding to business situations, in: Inaugural International Conference on Distributed Event-Based Systems, DEBS'07, ACM, Toronto, Ontario, Canada, 2007, pp. 198–205.
- [20] D. Weng, M. Bauer, Using policies to drive autonomic management of virtual systems, in: 2010 Intl. Conf. of Network and Service Management, CNSM'10, Niagara Falls, Canada, pp. 258–261.
- [21] A. Sulistio, K.H. Kim, R. Buyya, Managing cancellations and no-shows of reservations with overbooking to increase resource revenue, in: Intl. Symp. on Cluster Computing and the Grid, CCGRID 2008, IEEE Computer Society, Lyon, France, 2008, pp. 267–276.
- [22] P. Dube, Y. Hayel, L. Wynter, Yield management for IT resources on demand: analysis and validation of a new paradigm for managing computing centres, *J. Revenue Pricing Manag.* 4 (1) (2005) 24–38.
- [23] D.A. Menascé, V.A.F. Almeida, R. Fonseca, M.A. Mendes, Business-oriented resource management policies for e-commerce servers, *Perform. Eval.* 42 (2000) 223–239.
- [24] N. Poggi, T. Moreno, J. Berral, R. Gavalda, J. Torres, Self-adaptive utility-based web session management, *Comput. Netw.* 53 (10) (2009) 1712–1721.
- [25] M. Macías, O. Rana, G. Smith, J. Guitart, J. Torres, Maximizing revenue in grid markets using an economically enhanced resource manager, *Concurr. Comput.: Pract. Exp.* 22 (2010) 1990–2011.
- [26] Drools Rule Engine, 2014. <http://www.jboss.org/drools>.
- [27] Client classification and reclassification policy of Rabobank Polska Sa, 2014. <https://www.rabobank.com/en/locateus/eu/poland/mifid.html>.
- [28] Amazon EC2 instances, 2014. <http://aws.amazon.com/ec2/instance-types/>.
- [29] T. Puschel, N. Borissov, M. Macías, D. Neumann, J. Guitart, J. Torres, Economically enhanced resource management for internet service utilities, in: The 8th International Conference on Web Information Systems Engineering, WISE 2007, in: Lecture Notes in Computer Science, vol. 4831, Springer, Nancy, France, 2007, pp. 335–348.
- [30] J. Guitart, M. Macías, O. Rana, P. Wieder, R. Yahyapour, W. Ziegler, Market-Oriented Grid and Utility Computing, vol. 12, Wiley, 2009, pp. 261–284.
- [31] S. Ferretti, V. Ghini, F. Panzeri, M. Pellegrini, E. Turrini, QoS-aware clouds, in: 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD'10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 321–328.
- [32] I. Goiri, F. Julia, J.O. Fito, M. Macías, J. Guitart, Resource-level QoS metric for CPU-based guarantees in cloud providers, in: Proceedings of the 7th International Workshop on the Economics and Business of Grids, Clouds, Systems, and Services, GECON 2010, vol. 6296, pp. 34–47.
- [33] G. Reig, J. Alonso, J. Guitart, Prediction of job resource requirements for deadline schedulers to manage high-level SLAs on the cloud, in: 9th IEEE Intl. Symp. on Network Computing and Applications, Cambridge, MA, USA, pp. 162–167.
- [34] I. Goiri, F. Julia, J. Ejarque, M. de Palol, R. Badia, J. Guitart, J. Torres, Introducing virtual execution environments for application lifecycle management and SLA-driven resource distribution within service providers, in: 8th IEEE Intl. Symposium on Network Computing and Applications, NCA'09, Cambridge, MA, USA, pp. 211–218.
- [35] I. Goiri, F. Julia, J. Guitart, Efficient data management support for virtualized service providers, in: 17th Euromicro Conf. on Parallel, Distributed and Network-Based Processing, PDP'09, Weimar, Germany, pp. 409–413.
- [36] J. Torres, D. Carrera, V. Beltran, N. Poggi, K. Hogan, J. Berral, R. Gavalda, E. Ayguade, T. Moreno, J. Guitart, Tailoring resources: the energy efficient consolidation strategy goes beyond virtualization, in: 5th IEEE International Conference on Autonomic Computing, ICAC 2008, Chicago, Illinois, USA, pp. 197–198.



Mario Macías obtained his Master Degree in Computer Sciences at the Universitat Autònoma de Barcelona. After some years of working in IT companies, he joined Barcelona Supercomputing Center (BSC) in 2006 to work in European Research Projects and carry out his Ph.D. studies at Technical University of Catalonia (UPC). He combines his main job at BSC with his lectures about Object-Oriented programming at UPC.



Jordi Guitart received his M.S. and Ph.D. degrees in Computer Science from the Technical University of Catalonia (UPC), in 1999 and 2005, respectively. Currently, he is an associate professor at the Computer Architecture Department of the UPC and an associate researcher at Barcelona Supercomputing Center (BSC). His research interests are oriented towards innovative resource management approaches for modern distributed computing systems. He is involved in a number of European projects.