Client Classification Policies for SLA Enforcement in Shared Cloud Datacenters

Mario Macías and Jordi Guitart

Barcelona Supercomputing Center and Universitat Politecnica de Catalunya
Jordi Girona 29, 08034 Barcelona, Spain
{*mario.macias, jordi.guitart*}*@bsc.es,* {*mario, jguitart*}*@ac.upc.edu*

***Abstract*—In Utility Computing business model, the owners of the computing resources negotiate with their potential clients to sell computing power. The terms of the Quality of Service (QoS) and the economic conditions are established in a Service-Level Agreement (SLA). There are many scenarios in which the agreed QoS cannot be provided because of errors in the service provisioning or failures in the system. Since providers have usually different types of clients, according to their relationship with the provider or by the fee that they pay, it is important to minimize the impact of the SLA violations in preferential clients. This paper proposes a set of policies to provide better QoS to preferential clients in such situations. 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). Most of the policies use key features of virtualization: Selective Violation of the SLAs, Dynamic Scaling of the Allocated Resources, and Runtime Migration of Tasks. The validity of the policies is demonstrated through exhaustive experiments.**

I. Introduction

The Utility Computing business model is increasing its acceptance [1] thanks to the burst of Cloud Computing [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 a fee in function of the usage. The terms of the Quality of Service (QoS) to be provided and the economic conditions are established in a Service-Level Agreement (SLA). Utility Computing allows users to benefit from economies of scale by minimizing the space and maintenance costs. In this scenario, companies that own their computing resources may decide to hire out the spare resources of their data centers to external users [3]. The price that external users pay to use the resources contributes to amortize the cost of the data centers. However, a binary classification of the users as internal/external is not accurate enough in many situations. For example, headquarters of a big company may classify the users of its data centers according to different levels: users from the headquarters that owns the resources are completely internal, users from other companies are completely external, and users from other headquarters of the same company have an intermediate range. 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 users in between would pay a reduced fee that does not report profit, but encourages each location to only use resources from external locations when strictly necessary.

Another example of intermediate users are those from trusted entities that share their computing resources to share risks and deal with peaks of workload without the need to overprovision resources. Examples of trusted entities are different companies from the same business cluster [4].

Another criterion to classify clients is the QoS that they have purchased. For example, Spotify [5] 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 services and QoS: unlimited streaming hours, high quality of sound or available downloads. The provider must consider the purchased QoS when allocating the resources and managing the SLAs.

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 allocation and management of resources. This paper suggests applying Client Classification 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 have purchased: the higher the QoS range, 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.

Considering the aforementioned, our contributions are:

1) Proposal of innovative approaches to enforce SLAs

by pursuing a main Business-Level Objective (BLO): differentiation of users according of their Affinity/QoS relationship with the provider.

2) Demonstration of the validity of the model through fine-grained experiments that demonstrate how a provider can reach its BLOs without penalizing its preferential users. The results are evaluated in terms of revenue, client affinity, QoS, and SLA fulfillment.

The experiments have been performed with the Economically Enhanced Resource Manager (EERM) simulator [6]; a fine-grained, customizable Cloud market simulator that applies several Business policies and allows users to define new custom policies. Both the policies and the simulations are targeting the infrastructure layer within Cloud Computing (Infrastructure as a Service, IaaS).

The remainder of this paper is structured as follows. After the discussion of the related work, Section III describes the scenario in which Client Classification is applied as well as the proposed rules for Client Classification: their motivation and their concrete implementation. Next, Section IV describes the simulation environment and section V shows the experimental results that demonstrate de validity of the rules. At the end, Section VI describes the conclusions of this paper and states the future research lines.

II. Related Work

This paper extends our previous work in SLA Management for maximizing BLOs [7] and Client Classification Policies for SLA Negotiation and Allocation [8]. First [7] we introduced policies to maximize the revenue of providers in Cloud Markets: dynamic pricing, overselling of resources, dynamic scaling of resources, migration of Virtual Machines (VMs), etc. Next [8] we modified dynamic pricing and overselling policies to classify clients in SLA negotiation and allocation. In Dynamic Pricing, the provider applies price discounts that are proportional to the affinity of the clients. In overselling, the provider estimates optimistically or pessimistically the workload of the client in function of its affinity or QoS: if there are many preferential clients in the system, the provider oversells prudentially to decrease the risk of SLA violation. We demonstrated that these policies increase the percentage of preferential clients in the system. This paper introduces new ones that are applied at runtime for providing high QoS to preferential users.

Aiber et al. [9] 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 [10] promotes cooperation between resources for their optimization by means of a set of rules. Schiefer et al. [11] 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. [12] propose an autonomic management system of VMs that relies 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 paper extends this approach by adding other reactive actions, such as migration of resources or cancellation of tasks, and is not just limited to guarantee QoS but also Business Metrics.

Client Classification is usual in many businesses such as banking services [13]. 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, Quality of Service, etc. In Cloud Computing, Amazon Elastic Computing Cloud (EC2) provides a set of predefined VM instances [14], each one with different performance profiles (CPU load, Memory, etc.), but a fixed Quality of Service: they promise that their machines have an annual availability of 99.5%. This approach may be economically suitable for huge resource providers, but not for smaller providers. With this paradigm, small providers should overprovide resources for minimizing risks and provide high availability. 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. This paper extends the approach of Amazon by introducing and evaluating many policies for Client Classification combined with other features of Cloud Computing: dynamic elasticity of resources, migration of VMs across the resources pool, or cancellation of tasks.

Previous papers introduced some policies similar to those introduced in this paper. Sulistio et al. [15] proposed overbooking strategies for mitigating the effects of cancellations and no-shows for increasing the revenue. In addition to that, we consider the possibility of under-usage of the reserved resources of the client. Dube et al. [16] establishes 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 our proposal about establishing Gold, Silver and Bronze ranges and optimizing their QoS performance giving priority to the contracts that report the

highest economic profit. We extend this work by combining the QoS ranges with many other policies, such as Pricing or Selective Violation of SLAs. Another main difference between this paper and the work from Sulistio et al. and Dube et al. is that the main BLO of our work is the Client Classification instead of the Maximization of the Revenue.

Püschel et al. [17] propose a scheme for Client Classification by means of price discrimination, different priorities in job acceptance and differentiation in Quality of Service. They adopt the architecture of the EERM, which supports the optimization of SLA Negotiation and Allocation by dealing with both economic and technical information of Cloud Markets. We extend the work of Püschel et al. by adding new policies for SLA Management at runtime, and deeper validation of them by means of a tailored simulator of Clients, Cloud Market, EERM and Resource Fabrics.

III. Preliminary definitions

A Cloud Market has two main actors: Clients and Providers. Clients try to buy resources in the Market to host their services, by sending resource requests to providers to start a negotiation. Each provider owns a set of N physical machines. Each physical machine can host several VMs that execute single tasks, such as Web Services or Batch Jobs. The QoS terms of a task are described in $SLA = \{Rev(vt), C, \overrightarrow{S}, \Delta t\}$:

- $Rev(vt)$ is a revenue function to quantify how much the provider earns after finishing correctly or incorrectly a task. 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, MR the Maximum Revenue, MPT the Maximum Penalty Threshold, and MRT the Maximum Revenue Threshold, Equation 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 Figure 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.
- C is the client information. Let id be the client identifier and $\overrightarrow{CD}$ a vector that handles the description of the client, then $C = \{id, \overrightarrow{CD}\}$. The information contained in $\overrightarrow{CD}$ must be decided by the System Administrator and applied consequently in the policies.

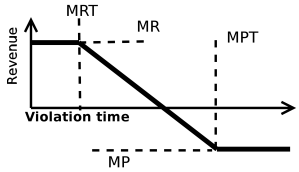

Figure 1: Revenue function of an SLA (Equation 1)

- $\overrightarrow{S}$ describes the QoS of the purchased service: throughput, response time, and so on.
- Δt is the time period requested to allocate the task.

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

Current Utility Computing market implementations [18], [19] provide common components for accounting and billing the usage of the resources. They also provide an SLA Enforcement component that continuously checks if the provider is fulfilling the agreed SLAs. In case an SLA is violated, the SLA Enforcement component imposes the appropriate sanctions to the provider (Equation 1).

A. Client Classification criteria

We propose the classification of clients according to the **priority** that the provider gives them. This priority can be described using 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 $= 0.25 \sim 0.75$ 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 may be different among different providers, depending on their business goals. How affinity is calculated is not important in this paper: the main topic is how to discriminate clients in function of their affinity.

Quality of Service: The same Cloud provider could host critical tasks and tasks that can tolerate low QoS in some situations. For example, e-commerce applications may need extra QoS guarantees to avoid losing money on service failure. 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 $Rev(vt)$ (Equation 1). We define three ranges of QoS, in descending order: Gold, Silver and Bronze. The higher the QoS range, the higher MR and the lower MP, MRT and MPT (lower values of these three values imply higher penalties).

This paper proposes policies that are applied when the SLAs are managed during service operation. During this

phase, the provider must be able to deal with unexpected events such as resource overloading or system failures. The goal is keeping the agreed QoS and minimizing the violations of SLAs from clients to which the provider has high affinity.

B. Rules for client classification at runtime SLA enforcement

To facilitate the reading of this paper, the names of the policies have been abbreviated according to the next notation: $PolicyName^{Criterion}$. $PolicyName$ is an abbreviation of the policy name. The abbreviations of all the policies are shown below, enclosed in parentheses next to their names. $Criterion$ is an abbreviation of the magnitude that is used for calculating the priority of the client: the affinity (Aff) or the Quality of Service (QoS). When the policies for Client Classification are compared with policies that prioritize the maximization of the revenue, the abbreviation for this last priority is RM (Revenue Maximization). As example, Price Discrimination policies that apply discount to clients according to their affinity are notated as $SLAViol^{Aff}$.

The proposed policies are:

Selective Violation or Cancellation of SLAs (SLAViol or SLACanc): Overselling [8] increases the violations of SLAs in the provider side. The Selective SLA Violation selects at every moment the SLAs to be partially violated (i.e., during some time) to minimize the violations of other SLAs from clients with higher priority. In some extreme scenarios, such as partial failure of the resource fabrics [20], some selected SLAs could be completely canceled for minimizing the impact in the BLOs achievement.

Dynamic Scaling of Resources (DynScal): The spare resources are dynamically assigned to high-priority clients whose SLA is not being fulfilled.

Runtime Migration of Tasks (RtMigr): When the load of the resources becomes high, high-priority services are migrated to other hardware resources with low average priority of tasks.

IV. Simulation Environment

This section describes the experimental environment and its configuration values. We have used the EERM Simulator [6] to execute and evaluate the policies that are introduced in this paper. The EERM Simulator is a fine-grained Cloud Market simulator, which simulates the complete cycle of a Cloud Resource sale and execution: services discovery, SLA negotiation process 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. In addition, it integrates the Drools [21] Rule Engine to allow configuring the SLA enforcement policies in function of the BLOs (e.g. the Client Classification policies described in this paper). We have used a simulated environment because it allows generating more data with limited resources in short time. That will allow to evaluate more precisely the models.

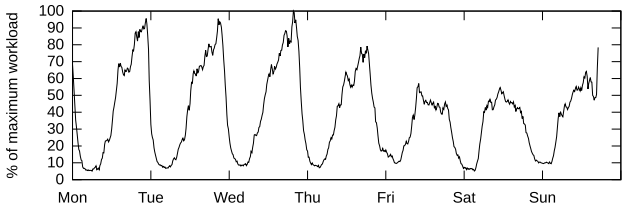

Figure 2: Sample pattern of web workload

The simulated data centers are sized to represent a small-medium company that wants to externalize part of its resources for quicker amortizing the infrastructure costs, as stated in Section I. However, our research also focuses big providers, since the policies are applied at the level of each individual hardware node. Regarding our approach, the only difference between big and small providers is that the former own a higher number of hardware 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 paper is to show how Client Classification reports benefit **qualitatively**, not quantitatively. In other words, the paper shows how a given policy can improve the Quality of Service to the preferential clients, but not whether the numeric values are optimum, because they would vary in function of 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 specific values.

In the market, clients try to buy resources to host their Web services. They send resource requests that contain $\{QoS, C, \overrightarrow{S}, \Delta t\}$, 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 is acquired from a real anonymous ISP (see Figure 2), 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 to the 25% of the clients in the market, and equal to 0 to the other 75% of clients. The affinity of the clients of the same organization than the provider ranges from 0 (non-inclusive) to 1 (inclusive) with an 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 [22] to predict future workloads and verify whether the offered job can be executed correctly. A bad prediction might 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 him a confirmation.

Section III-B explains the different policies that are introduced in this paper and demonstrates its validity through market simulations. In each subsection, a new policy is introduced and 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 introduced 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 (depending on the classification criteria), but excluding the policy that is being introduced in the corresponding subsection.
4) A provider that executes the same policies as Providers 1 and 2 but without client classification as a main BLO. Its priority is the maximization of the economic profit [7]. We do not evaluate its results, but we want to show the effects of its competition with the other three providers in the simulations.

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.

V. Rules definition and evaluation

A. Selective Violation of SLAs (SLAViol)

Our previous work [8] introduced price discrimination and overselling policies to increase the percentage of preferential clients that use the system. However, the provider could oversell too many resources due to errors when estimating the workloads and some SLAs will be violated indiscriminately. We propose to violate first the SLAs of clients with low priority: tasks of the clients to which there is low priority 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: $NR(SLA_i) = \sum_{j=0}^{i-1} Rev(vt_j) * P(C_j) + \sum_{j=i+1}^{M} Rev(vt_j) * P(C_j) + Rev(vt_i + \delta) * P(C_i)$, in which $P(C_i)$ 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. This formula considers both revenue and client priority for choosing the SLA to be violated.

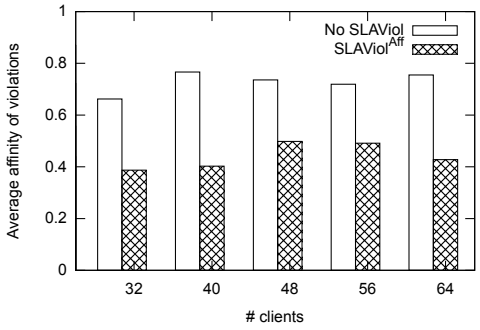

Figure 3: Average affinity of violations with $SLAViol^{Aff}$

As defined in Equation 1, each SLA has a grace period ($vt \leq MRT$) in which violating it will not involve a penalization. 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 distribute the selective violations across the SLAs in grace period (with some variance due to their client priority) and keep a compromise between the Client Classification and Revenue Maximization objectives.

The experiments performed in this section shows only the results of simulations from 32 to 64 clients, because the load of the 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 both $SLAViol^{Aff}$ and $SLAViol^{QoS}$. Figure 3 compares the average affinity of the SLA violations of two providers. The average affinity is the addition of the affinities of all the violated SLAs divided by the number of SLA violations. Both apply overselling and price discrimination policies, but one provider applies $SLAViol^{Aff}$ and the other does not. The figure shows that the average affinity of violations decreases ∼30-50% when applying $SLAViol^{Aff}$ because the provider chooses to violate SLAs from clients to which there is low affinity. The results of prioritizing by QoS range are shown in Figures 4 and 5, which show the percentage of each QoS range from the total of SLA violations in several market simulations with different number of clients. Figure 4 shows that the higher the QoS rank the higher the percentage of violated SLAs because high-QoS SLAs are harder to fulfil. Figure 5 shows that the proportion of high-QoS SLAs that are violated is considerably reduced with $SLAViol^{QoS}$.

A special case of $SLAViol$ is the Selective Cancellation

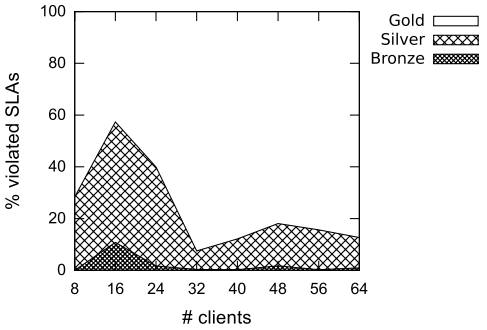

Figure 4: % of violations by QoS range without $SLAViol^{QoS}$

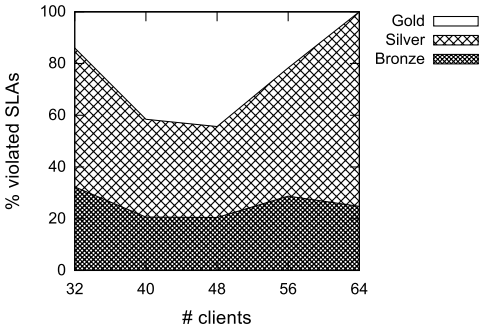

Figure 5: % of violations by QoS range with $SLAViol^{QoS}$

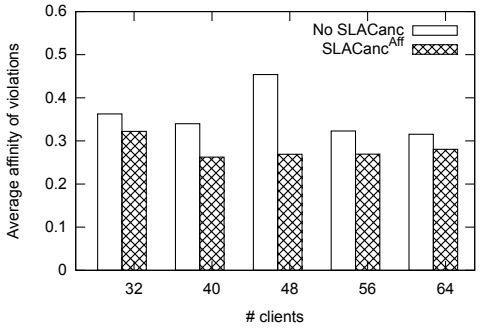

Figure 6: Average affinity of violations with $SLACanc^{Aff}$

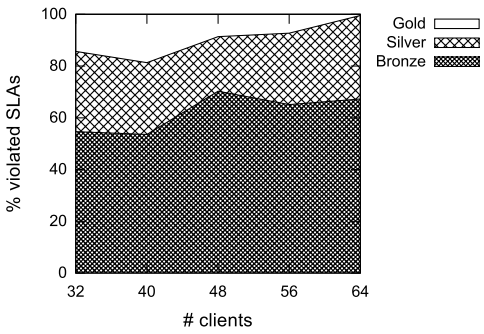

Figure 7: % of violations by QoS range with $SLACanc^{QoS}$

of SLAs ($SLACanc$): instead of pausing the lowest-priority VMs, $SLACanc$ policy cancels them completely. Figures 6 and 7 show the results of an experiment where $SLACanc$ is applied: both the average affinity of the violations and the percentage of violations of high-QoS SLAs decreases noticeably in providers that respectively apply $SLACanc^{Aff}$ and $SLACanc^{QoS}$. However, $SLACanc$ must be applied with extreme caution because it increases enormously the number of violations, specially to clients with low affinity (up to 2000% in the experiments). Applying $SLACanc$ would lead to decreasing the reputation of the provider [23]. $SLACanc$ must be only applied in special cases, such as reorganizing tasks after a partial failure of the system, similar to the Amazon EC2 outage in April 2011 [20].

B. Dynamic Scaling of Resources (DynScal)

$DynScal$ uses the potential of elasticity in virtualization: hardware resources are transparently reallocated within VMs at runtime [24]. When an SLA is violated, the EERM transfers resources from the VMs that are not using all their assigned resources to the VM whose SLA is violated. This policy uses the client priority as a factor to decide which VM the resources are subtracted from: the monitoring data about the resources usage of VMs from a low-priority client is artificially decreased by a percentage that is linearly proportional to the priority, and increased similarly in VMs from high-priority clients. In consequence, the system will act as if low-priority VMs tend to have free resources to transfer to high-priority VMs. Depending on the BLOs of each provider, the scale of resources could be proportional to the priority in a non-linear distribution.

Figure 8 shows that $DynScal^{Aff}$ 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. Figure 9, if compared with Figure 5, shows that the number of Gold SLAs that are violated is reduced by 50% when applying $DynScal^{QoS}$.

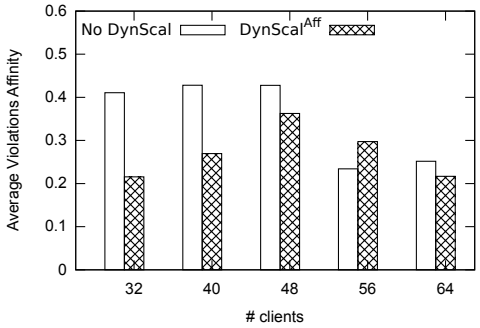

Figure 8: Average affinity of violations with $DynScal^{Aff}$

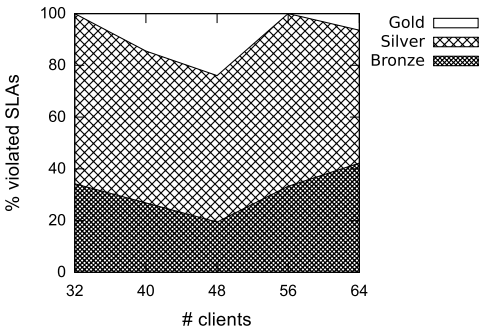

Figure 9: % of violations by QoS range with $DynScal^{QoS}$

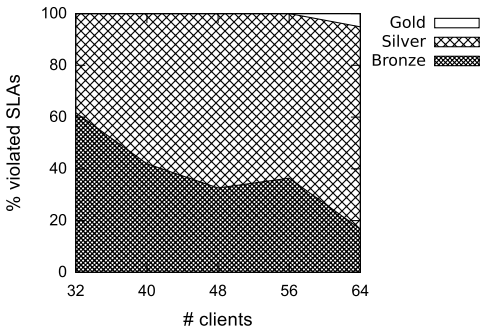

Figure 10: % of violations by QoS range with $RtMigr^{QoS}$

C. Runtime Migration of Tasks (RtMigr)

This policy uses another facility of virtualization: the live migration of VMs across the resources pool with performance penalty near to zero [25]. If the load of a physical machine exceeds a threshold of 90% of its maximum capacity, the $RtMigr$ policy is triggered: the system tries to find another physical machine in the pool where the average priority of its running tasks is lower than the machine that triggered $RtMigr$ and, if found, it looks in the overloaded machine for the task with the highest client priority and migrates it to the target machine.

$RtMigr$ policy combines well with $SLAViol$. Even if the migration of a task could simply translate 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 low-priority SLAs in every overloaded machine. Otherwise, if $RtMigr$ was not applied there would be physical machines that only execute high-priority tasks and the EERM would violate one of them even if there were low-priority SLAs in the other resources.

The concrete value for the threshold (90% in the experiments) is not important in this work, because we want to show that applying $RtMigr$ decreases the violations of high-priority SLAs. Finding the optimum value for this threshold is part of our future work.

The effects of $RtMigr$ vary when applying $RtMigr^{Aff}$ or $RtMigr^{QoS}$. The average affinity of the violations is only reduced about 3-5% in average in all the scenarios (so the figure that shows it has not been considered interesting). The cause is that the more policies implemented in the provider the less percentage of improvement (which tends to a limit) is obtained by adding new ones. On the other hand, Figure 10 shows the effects of live migration when prioritizing high-QoS tasks: the violations of Gold tasks are nearly 0, and Silver violations are reduced. Figure 10 also shows that the more clients, the lesser effects of runtime migration. The reason is that migrated tasks would be also violated in the target machines because the high load of the resources.

Another advantage of $RtMigr$ is the reduction of the number of violations. Figure 11 shows that the violations

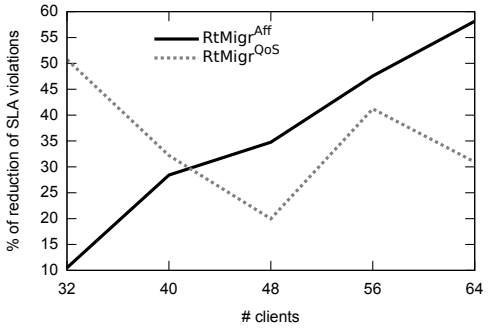

Figure 11: % of reduction of violations with $RtMigr$

are reduced almost linearly with the number of clients when using $RtMigr^{Aff}$. That does not mean that the total of violations is lower with 64 clients than with 32 clients, because although the percentage of reduction is high, the total number of violations is also higher.

VI. Conclusions and Future Work

We have introduced a set of policies for managing SLAs during service operation in a Cloud provider, classifying the clients according to two facets: client affinity and QoS. These policies have been evaluated through experiments that show how each policy can improve the prioritization of clients when combined by the other policies. After the application of all the policies, the EERM increases the number of high-priority clients (high-affinity or Gold and Silver, depending on the chosen type of priority), and keeps a good compromise of quality with them by violating a low percentage of high-priority SLAs. Some of the policies also reduce the number of violations.

We conclude that Client Classification policies in SLA enforcement achieve their objectives: the percentage of violated SLAs is lower for users to which there is high priority. 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). Classification by affinity is more suitable for organizations that mix internal and external applications on their resources.

The policies presented in this paper rely on some constant values that may not lead to the optimum achievement of the BLOs, which is not the main objective in this paper. Our aim is to show how the policies decrement both the number of SLA violations and the priority of the clients that the SLAs belong to. Finding the optimum 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 optimum results according to their own BLOs.

In future work lines, 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.

ACKNOWLEDGEMENTS

This work is supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contract TIN2007-60625, by the Generalitat de Catalunya under contract 2009-SGR-980, and by the European Commission under FP7-ICT-2009-5 contract 257115 (OPTIMIS).

REFERENCES

[1] M. A. Rappa, "The utility business model and the future of computing services," *IBM Syst. J.*, vol. 43, no. 1, pp. 32–42, 2004.

[2] R. Buyya, C. S. Yeo, and 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)*. Dalian, China: IEEE Computer Society, September 2008, pp. 5–13.

[3] I. Foster, "The anatomy of the grid: Enabling scalable virtual organizations," *Cluster Computing and the Grid, IEEE International Symposium on*, vol. 0, p. 6, 2001.

[4] M. E. Porter, "Clusters and the new economics of competition," *Harvard Business Review*, vol. 76, no. 6, pp. 77–90, Nov-Dec 1998.

[5] Spotify. [Online]. Available: http://www.spotify.com

[6] Economically Enhanced Resource Manager (last visit: Aug. 2011). [Online]. Available: http://www.sf.net/projects/eerm

[7] M. Macias, O. Fito, and 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, October 2010, pp. 354–357.

[8] M. Macias and J. Guitart, "Client classification policies for SLA negotiation and allocation in shared cloud datacenters," in *To be published in Proceedings of the 8th International Workshop on the Economics and Business of Grids, Clouds, Systems, and services (GECON 2011)*, December 2011.

[9] S. Aiber, D. Gilat, A. Landau, and A. Sela, "Autonomic self-optimization according to business objectives," in *Proceedings of the First International Conference on Autonomic Computing*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 206–213.

[10] P. Herrero, J. L. Bosque, M. Salvadores, and M. S. Perez, "A rule based resources management for collaborative grid environments," *Int. J. Internet Protoc. Technol.*, vol. 3, no. 1, pp. 35–45, 2008.

[11] J. Schiefer, S. Rozsnyai, C. Rauscher, and G. Saurer, "Event-driven rules for sensing and responding to business situations," in *Inaugural International Conference on Distributed Event-Based Systems (DEBS 07)*. Toronto, Ontario, Canada: ACM, 2007, pp. 198–205.

[12] D. Weng and 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, Oct. 2010, pp. 258–261.

[13] Client classification and reclassification policy of rabobank polska sa (last visit: Feb. 2011). [Online]. Available: http://goo.gl/AKu86

[14] Amazon EC2 instances (last visit: Feb. 2011). [Online]. Available: http://aws.amazon.com/ec2/instance-types/

[15] A. Sulistio, K. H. Kim, and 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)*. Lyon, France: IEEE Computer Society, May 2008, pp. 267–276.

[16] P. Dube, Y. Hayel, and L. Wynter, "Yield management for IT resources on demand: analysis and validation of a new paradigm for managing computing centres," *Journal of Revenue and Pricing Management*, vol. 4:1, pp. 24–38, 2005.

[17] T. Püschel, N. Borissov, M. Macias, D. Neumann, J. Guitart, and J. Torres, "Economically enhanced resource management for internet service utilities." in *WISE*, ser. Lecture Notes in Computer Science, vol. 4831. Springer, 2007, pp. 335–348.

[18] Catnets. [Online]. Available: http://www.catnets.uni-bayreuth.de

[19] D. Neumann, J. Stoesser, A. Anandasivam, and N. Borissov, "SORMA - building an open grid market for grid resource allocation," in *4th international conference on Grid economics and business models (GECON'07)*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 194–200.

[20] R. Tehrani, "Amazon EC2 outage: what the experts tell us," *Customer Interaction Solutions*, vol. 29, no. 12, p. 1, May 2011.

[21] "Drools rule engine." [Online]. Available: http://www.jboss.org/drools

[22] G. Reig, J. Alonso, and 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, July 2010, pp. 162–167.

[23] M. Macias and J. Guitart, "Influence of reputation in revenue of grid service providers," in *2nd International Workshop on High Performance Grid Middleware (HiPerGRID'08)*, Bucharest, Romania, Nov. 2008.

[24] I. Goiri, F. Julia, J. Ejarque, M. de Palol, R. Badia, J. Guitart, and 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, July 2009, pp. 211–218.

[25] I. Goiri, F. Julia, and 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, February 2009, pp. 409–413.