Exploiting SLAs through Application of Economic Analysis on Datacenters' Autonomic Management

Eduardo L. Falcão[1,2], Lucas Cavalcante[1], Rafael V. Falcão[1], José B. S. Nunes[1] Kaio K. M. Oliveira[1], Andrey Brito[1]

[1]Departmento de Sistemas e Computação, Universidade Federal de Campina Grande

[2]Centro Universitário Unifacisa, Campina Grande

{eduardolfalcao,lucascavalcante,rafaelvf,benardi}@lsd.ufcg.edu.br,

kaiokmo@lsd.ufcg.edu.br,andrey@computacao.ufcg.edu.br

Abstract. *Optimizing resource distribution in datacenters can achieve several economic benefits, including increasing energy efficiency, better workload performance, and higher levels of job acceptance. Although some studies propose strategies that increase resource utilization whilst meeting a certain QoS, there is a perspective yet to be exploited: the Service Level Agreements (SLA) amortization. We believe that keeping service levels far above the minimum agreed is a missed opportunity to apply more aggressive strategies to reduce allocated resource fragmentation and increase even further the benefits mentioned above. In this work, we propose a novel server rebalancing strategy based on economic systems, able to capitalize from the SLAs by keeping the harmony between the aggressive competition of Free Markets and the safe control of Regulated Markets. The proposed strategy is evaluated with a modified version of CloudSim Plus simulator. The outcomes show that the SLA exploitation strategy led to an economy of* 8.6% *up to* 23% *for different mechanism configuration.*

1. Introduction

There is increasing environmental concern about the energy needed to support datacenters in the cloud computing industry. In 2014, U.S. datacenters consumed 1.8% of the energy of the whole country [Shehabi et al. 2016]. However, this problem is not novel. There has been extensive investigation on the issue of energy efficiency from the perspective of virtual machine (VM) scheduling heuristics [Alahmadi et al. 2014, Song et al. 2014] and host consolidation [Hameed et al. 2016, Lee and Zomaya 2010, Usmani and Singh 2016]. Despite some of these strategies being able to save energy and improve resource utilization, some can lead to a performance degradation due to resource contention between VMs [Usmani and Singh 2016]. In some cases, the scarcity of resources reaches such degree that the Service Level Agreements (SLAs) may be violated.

In this sense, there are already several works [Alahmadi et al. 2014, Hameed et al. 2016, Usmani and Singh 2016] that avoid high levels of resource contention between VMs through different host rebalancing strategies, conciliating energy savings and Quality of Service (QoS). In addition to the conventional server rebalancing and consolidation strategies, which typically migrate VMs from overloaded hosts while maintaining a placement that favors host consolidation, our work **exploits the SLA** of VMs to its edge.

Whenever a client leases VMs from a public Infrastructure-as-a-Service (IaaS) cloud provider, she signs a virtual contract describing the responsibilities of the service provider — the Service Level Agreements. Although there is no strict model of which services and QoS levels different companies should address on their SLAs, there is a recurrent pattern of promising minimum levels of resource availability or minimum rates of performance. For instance, Microsoft Azure [Microsoft Azure 2019] and Amazon Web Services [Amazon Web Services 2019] promises availability levels of 99.99%, 99.95%, 99.9% or 99% for different classes of VMs. Instead of always trying to provide their clients with the best service availability/performance, we advocate that IaaS cloud providers could push this value to the minimum promised in their SLAs, reducing costs for both the client and the provider. Thus, we propose that the companies use this "safety margin" to obtain more utilization efficiency by employing more efficient, but riskier, resource management strategies. If the client is not satisfied with the risks posed, then she should pay a bit more for a more conservative SLA with higher levels of availability.

We achieve this goal by reducing the idleness of allocated resources, hereafter referred to as **allocated resource fragmentation**. However, since higher levels of resource contention increase the odds of violating the contract, our main novelty is the strategy employed to avoid a breach of contract. In our proposed SLA exploitation strategy, we focus on guaranteeing Service Level Objectives (SLOs) of performance. We evaluate the QoS by considering the **SLO compliance rate**: the percentage of time in which a VM committed no SLO violations. Roughly, VMs with SLO compliance rate higher than the minimum contracted are considered for server rebalancing, once they can occasionally experiment momentary SLO violations without breaking their contracts. The remaining period a VM can commit SLO violations without breaking the overall contract is called *credit time*. For instance, a VM with no SLO violations and whose contract promises to provide a minimum compliance rate of 99% would have 14 minutes of credit at the end of a day – a time that could be used to decrease allocated resource fragmentation by employing riskier resource provisioning strategies.

In this work, we leverage the *Economic Analysis Framework* [Huberman and Hogg 1995] to devise a novel resource provision strategy: SL-Tight. The basic and most used resource provision approach is a server allocating all the resources requested to its hosting VMs, which typically assures high QoS but may lead to resource underutilization. We call this conventional strategy the **SL-Loose** approach since the SLO compliance rates are far above the promised on the SLAs. By analogy, this is precisely the behavior of a government ruling a *Regulated Market* economy and, therefore, in the SL-Loose strategy, VMs are always hosted in what we call *regulated market servers*. In the **SL-Tight** approach, VMs without credit are also hosted in servers following the "regulated market" approach, so that there are no risks of contract breach. However, we propose that VMs with enough credit should be placed in servers following the "free market" approach, *i.e.*, servers in which VMs contend for resources likewise companies do in *Free Market* economic systems.

The main goal of this work is to assess the positive and negative impacts brought by the SL-Tight approach in relation to the SL-Loose: i) how much energy can be saved, ii) the impacts on workload performance, iii) the impacts on the SLO compliance rates, and iv) the impacts on the allocated resource fragmentation. Here we use the CloudSim

Plus simulator [Filho et al. 2017] in our assessment. The primary outcomes are the decrease of allocated resource fragmentation that generates, through host consolidation, an energy saving of 8.6% up to 23%, and that under some settings there is no contract breach even though SLAs has been exploited.

The rest of this paper is organized as follows. Section 2 presents the Economic Analysis Framework we rely on to conceive our novel resource provision mechanism. Our proposed SLA exploitation strategy is described in Section 3, followed by the problem and solution statements. A description of the simulator used is provided in Section 4. In Section 5 we present the experiment design. Simulation results and their corresponding analysis are presented in Section 6. The most relevant related work are presented in Section 7. Finally, in Section 8, we put forward the main conclusions.

2. Economic Analysis Framework

From an economic perspective, one may notice that cloud providers naturally behave as markets. A client must choose how much she wants to pay for a service, in accordance with her needs. Still, although the underlying infrastructure – the datacenter – is also analogous to a market, it is not treated as one. Therefore, in order to throw a glance at the datacenter resource provisioning using theories from the Economic Analysis Framework, the Regulated Market and Free Market Economies are briefly described next.

2.1. Regulated Market Economy

A Regulated Market is an economic system where the government zealously controls the forces of supply and demand, such as who is allowed to join the market and the prices to be practiced [Helms 2005]. In cloud datacenters, as each VM intends to consume resources, each resource provider, acting as the government, *regulates* the supply to not overflow what was agreed in the SLA. On the one hand, this regulation allows the cloud provider to manage the resource distribution comfortably and prevents neighboring VMs from interfering with each other [Ibidunmoye et al. 2015]. On the other hand, in some cases, a VM could be using fewer *goods*, creating a surplus for the cloud and although the provider is complying with the contract, having these idle and fragmented resources allocated leads to overall cloud inefficiency.

2.2. Free Market Economy

A Free Market is an economic system based on supply and demand laws. Applying it in a datacenter means that the resource provisioner will only guarantee that the VM will not starve for resources or be preempted for the benefit of a third-party. It also means that those competing for resources do so in equal terms.

Each VM, as an ultimate selfish entity, tries to consume all the resources it wants. As the competition is perfect, if two VMs are willing to consume all the resources of a server, each will get half. However, having a host share resources as a Free Market economy can lead to extreme resource depletion. Assume, for instance, a given host can supply 1 Gbit/s. If in a given moment, there are 1000 VMs, each will get 1 Mbit/s, a rate that could degrade the overall performance depending on the type of workload.

A Free Market economy host (**FM host**, for short) will not prevent the situation mentioned above because its government policy inhibits any interference on the competi-

tion. Still, a Regulated Market economy host (**RM host**) will not allow the placement of another consumer as it is unable to guarantee the SLA of all VMs.

3. SLA Exploitation

IaaS cloud providers could use a less conservative approach to ensure the agreed Service Level (SL), here called **SL-Tight**, instead of the most commonly used strategy, namely the **SL-Loose** approach, in which the service provider always tries to ensure the customer the *Optimal SL*. The basic idea is to provide, throughout the billing month, a tight SL, in the sense that the SL cumulatively provided gets closer to the minimum promised level (SLO). Figure 1 illustrates the *SLO compliance rate* provided by both strategies.

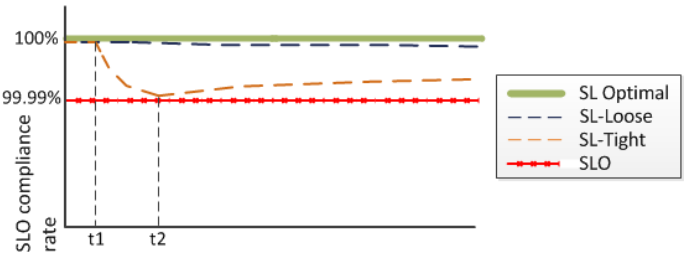

Figure 1. Strategies for ensuring the SLO.

Consider Figure 1. A specific VM accumulates a sufficient amount of credit time until time t_1, and after t_1 the VM is migrated to an FM host. During the time interval $[t1, t2[$ there is a steep drop in the SLO compliance rate, which could be, for instance, a consequence of resource contention. Finally, at t_2, it starts to increase again. This upsurge can be caused by a preemptive migration to an RM host or due to a decrease in resource contention in the FM host.

This problem is more formally defined next.

3.1. Problem Statement

With the goal of easing comprehension, we first present Table 1, containing the symbols used in this work and their corresponding descriptions.

A cloud datacenter is comprised by a set of hosts $\mathbb{H}$, where $\mathbb{H}_{RM} \subseteq \mathbb{H}$ denotes the set of RM hosts and $\mathbb{H}_{FM} \subseteq \mathbb{H}$ stands for the set of FM hosts. Let $\mathbb{V}$ be the set of VMs and $\mathcal{R} = \{iops, bandwidth, cpu, ram, storage\}$ the set of resources each machine (host or VM) has. Assume a given SL Objective $o \in \mathbb{O}$ is defined by a tuple $(r, \kappa_{min}, \delta)$, where $r \in \mathcal{R}$, $\kappa_{min} \in \mathbb{R}_{]0,1]}$ is the minimum compliance rate and $\delta \in \mathbb{R}$ is a constant defining the amount of demanded resources. Consider also that $\theta : \mathbb{O} \to \mathcal{R}$, $\mu : \mathbb{O} \to \mathbb{R}$ and $\rho : \mathbb{O} \to \mathbb{R}$ are functions that return the type of resource (r), the minimum compliance rate (κ_{min}) and the amount of requested resources (δ) of a given SLO.

To better understand what an SLO violation is, we formalize the allocation and workload utilization functions. For the sake of simplification, we assume that time is discrete and use step functions to formalize allocation as $\alpha : \mathbb{V} \times \mathcal{R} \times \mathbb{N} \to \mathbb{R}$ and the workload utilization function as $\omega : \mathbb{V} \times \mathcal{R} \times \mathbb{N} \to \mathbb{R}$. For instance, $\alpha(vm_A, iops, t)$ denotes the amount of IOPS allocated to VM A during time step t, and $\omega(vm_A, iops, t)$ denotes the IOPS utilization needed by the workload ran by VM A during time step t. Thus, the SLO violation function $\gamma : \mathbb{V} \times \mathbb{O} \times \mathbb{N} \to \{0, 1\}$ is

Symbol	Description
$\mathbb{H}_{RM}$	set of RM hosts
$\mathbb{H}_{FM}$	set of FM hosts
$\mathbb{V}$	set of VMs
$\mathcal{R}$	set of resources
κ_{min}	minimum compliance rate
δ	constant defining the amount of demanded resources
$\mathcal{O}$	set of SLOs
$\alpha(v, r, t)$	allocation of resource of type r for VM v in time t
$\omega(v, r, t)$	workload utilization of resource of type r, for VM v, in time t
$\gamma(v, o, t)$	occurrence of violation for SLO o and VM v, in time t
$\kappa(v, o, t)$	SLO compliance rate for SLO o, and VM v, in time t
$\Phi(r, \pi, t)$	allocated fragmentation of resource r, with placement π, in time t
$\tau(v, o, t)$	credit time of VM v, with respect to SLO o, in time t
$\mathcal{L}_{max}$	max amount of hosts that can be turned into an FM host
η	no. of VMs with enough credit to trigger a change from RM into FM host
λ	constant defining the duration of any VM migration
σ	safety time margin considered to avoid SLA breach
Δ	factor of change, in percentage, when scaling
$[\mathcal{S}_{min}, \mathcal{S}_{max}]$	range of proper values for workload utilization ($\omega(v, r, t)$)

Table 1: Table of symbols.

defined as:

$$\gamma(v, o, t) = \begin{cases} 1, & \text{if } \alpha(v, r, t) < \rho(o) \text{ and } \alpha(v, r, t) < \omega(v, r, t) \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where r is $\theta(o)$. Note that a momentary SLO violation is different from a contract breach.

The **SLO compliance rate** function is formalized as $\kappa : \mathbb{V} \times \mathbb{O} \times \mathbb{N}_{\geq 1} \to \mathbb{R}_{[0,1]}$. At the beginning of time t, considering we apply a mod operator so that t is always within the seconds of a month, the SLO compliance rate is computed by:

$$\kappa(v, o, t) = \frac{t - \sum_{i=0}^{t-1} \gamma(v, o, i)}{t}. \tag{2}$$

Thus, whenever $\kappa(v, o, t) < \mu(o)$ we say there is a breach of contract (or SLA violation).

A placement π is a tuple $(\mathbb{V}, h)$, comprised by a set $\mathbb{V}$ of VMs allocated to a given host h. Also, let $\mathbb{S} = \{SL - Loose, SL - Tight\}$ denote the set of strategies that can be used by a placement. Computing π using a strategy $s \in \mathbb{S}$, for each host $h \in \mathbb{H}$, generates Π, the set comprising the placements of the datacenter's VMs into hosts.

Let $\Phi : \mathcal{R} \times \Pi \times \mathbb{N} \to \mathbb{R}_{[0,1]}$ be a function where $\Phi(r, \pi, t)$ denotes the allocated fragmentation of resource r, in the end of time step t, when the placement π is applied. Then, whenever there is at least a unit of resource allocated, the normalized allocated resource fragmentation of a host is defined as follows:

$$\Phi(r, \pi, t) = \frac{\sum_{v \in \pi_1} max\left\{0, \alpha(v, r, t) - \omega(v, r, t)\right\}}{\sum_{v \in \pi_1} \alpha(v, r, t)}. \tag{3}$$

We believe that a host with no allocated resource fragmentation ($\Phi(r, \pi, t) = 0$) is optimal, as there exists no resource allocated that is idle. Thus, the challenge is to generate such a set of placements Π of VMs to hosts that at least a near-optimal allocated resource fragmentation is reached and the SLA is respected. We summarize this optimization problem with the following formulation:

$$\begin{aligned} \text{minimize} \quad & \Phi(r, \pi, t) \\ \text{subject to} \quad & \kappa(v, o, t) \geq \kappa_{min}, \\ & r \in \mathbb{R}, \pi \in \Pi, v \in \mathbb{V}, \\ & o \in \mathbb{O}, t \in \mathbb{N}. \end{aligned} \tag{4}$$

3.2. Solution Statement

The SL-Loose strategy applies the *simple resource provision* policy, in which a host accepts a VM only if the host has the number of resources (either CPU, RAM, IOPS, and bandwidth) requested by the VM. On the other hand, besides applying the *simple resource provisioner* for non-shared resources (RAM), the SL-Tight strategy also applies what we call the *shared resource provision* policy for shared resources (CPU, IOPS and bandwidth). In this policy, the only constraint that can preclude a host from accepting VMs is the RAM. The other resources do not impose any constraint because in FM hosts all available shared resources are divided equitably between all VMs, as described on Algorithm 1.

Algorithm 1 Resource Provisioner Shared

1: Sort VMs list by requested resource in ascending order.
2: Get the next VM in which the requested resources (δ) are not fulfilled.
3: If the host has enough resources to allocate δ to all VMs which are still requesting resources, allocate δ to all these VMs; go to 2.
4: If the host does not have enough resources to allocate δ to all VMs which are still requesting resources, divide all available resources between all these VMs.

To decrease allocated resource fragmentation, our solution, the SL-Tight, relies on the *shared resource provision* policy and on a novel server rebalancing heuristic, described next.

The SL-Tight strategy considers all VMs with a minimum amount of time credit to be placed in the FM hosts. The credit time of VM v with respect to SLO o in time t can be computed by the following function $\tau : \mathbb{V} \times \mathbb{O} \times \mathbb{N} \rightarrow \mathbb{R}$:

$$\tau(v, o, t) = \frac{\kappa(v, o, t) \cdot t \cdot (1 - \mu(o))}{\mu(o)} - t \cdot (1 - \kappa(v, o, t)) \tag{5}$$

In the SL-Tight, initially, $|\mathbb{H}_{FM}| = 0$ and $|\mathbb{H}_{RM}| = |\mathbb{H}|$. The strategy requires the cloud admin to define a limit $\mathcal{L}_{max} \in \mathbb{N}$ establishing the maximum amount of hosts that can be turned into FM hosts. Besides, the cloud admin must also choose a constant $\eta \in \mathbb{N}$ defining the amount of VMs with enough credit time that should be reached to consider converting an RM host into an FM host. In our heuristic, for the sake of simplicity, once turned into an FM host, a server will never be RM again.

The server rebalancing algorithm migrates VMs from $\mathbb{H}_{RM}$ to $\mathbb{H}_{FM}$ and vice-versa, based on the credit time of VMs. VMs with more credit time are iteratively allocated to FM hosts,

prioritizing hosts with higher resource availability. Here we assume the migration of any VM is performed in constant time λ. Thus, to avoid contract breach, in a given time t, only VMs that satisfies the condition $\tau(v, o, t) > \lambda + \sigma$, where σ is a safety margin, are considered for migration. We consider live migrations: while migrating from an RM to an FM host, the migrating VM still runs on the RM host and, besides, it has already provisioned resources from the FM host.

In a similar form, VMs that are about to violate their corresponding SLAs are migrated to RM hosts. Any VM v with a SLO o is considered to be a candidate to migration iff $\kappa(v, o, t) \leq \mu(o)$. However, since VM migrations are not instantaneous, we consider for migration any VM v with time credit lesser than or equal to the average migration time plus a safety margin, $\tau(v, o, t) \leq \lambda + \sigma$. Also, note that VMs with less credit time are prioritized.

Furthermore, we also consider before migrations trying RM hosts consolidation. The main idea is to move VMs from hosts with higher resource availability to hosts with lower resource availability so that, eventually, one of these hosts with low resource availability becomes empty, allowing its power off.

Finally, when the VMs are in an FM host, the scaling will be enabled so that VMs with idle resources reduce their allocated capacity and VMs needing more resources could perform upscaling. However, VMs already violating their SLOs ($\tau(v, o, t) \leq \lambda + \sigma$) are not allowed to downscale since this would worsen the situation. In this work, scaling is done for the IOPS resource, although the idea also applies to other shared resources such as bandwidth. The scaling variables are $\Delta \in [0, 1]$, $\mathcal{S}_{min} \in [0, 1]$, and $\mathcal{S}_{max} \in [0, 1]$. Recalling that $\alpha(v, iops, t)$ and $\omega(v, iops, t)$ are the amount of IOPS allocated and the amount of IOPS needed by the workload, for VM v in time t, then, if $\omega(v, iops, t) < \mathcal{S}_{min} \cdot \alpha(v, iops, t)$, the allocation of the next simulation time will be decreased by Δ, *i.e.*, $\alpha(v, iops, t + 1) = \alpha(v, iops, t) - \alpha(v, iops, t) \cdot \Delta$. This downscaling would release the allocation of idle resources, enabling overloaded VMs to benefit by offloading. Similarly, if $\omega(v, iops, t) > \mathcal{S}_{max} \cdot \alpha(v, iops, t)$, the allocation of the next simulation time will be increased by $2 \cdot \Delta$, *i.e.*, $\alpha(v, iops, t + 1) = \alpha(v, iops, t) + \alpha(v, iops, t) \cdot 2\Delta$. We consider this double-factor because such VMs may be violating or about to violate their SLO, and thus, a quick reaction is required.

4. CloudSim Plus Simulator

To evaluate our proposed strategy we opted for simulations, in particular, for the CloudSim Plus framework [Filho et al. 2017]. Our modified version of CloudSim is available at https://git.lsd.ufcg.edu.br/lucascavalcante/cloudsimplus.

In CloudSim Plus, a datacenter is comprised of a set of hosts, that in our case can be RM or FM. The *broker* acts as an abstraction of the user: it manages the submission and termination of *VMs* as well as the submission of *cloudlets* to VMs. A cloudlet is a simple task that can be executed by a VM. Cloudlets consume VMs' resources (CPU, RAM, etc), and VMs consume hosts' resources. Thus, each cloudlet carries workload information such as the amount of IOPS instructions to execute. The broker is also responsible for mapping VMs to hosts according to a given allocation policy that, typically, considers the VMs' resources requirements and the resource availability on the hosts.

There are two policies that hosts and VMs use to define how its resources are scheduled: *time shared* and *space shared*. The time shared policy allows a single resource to be allocated to multiple entities, each having the possession of such resources during a fraction of time, while in the space shared, a portion of resources is allocated only to a single entity. Furthermore, hosts provision resources to VMs and VMs provision resources to cloudlets with the aid of the *resource provisioner*. Provisioners can be either *simple* or *shared* (see Section 3.2).

The *SLA monitor* is the component that computes the credit time of each VM using the SLO compliance rate, according to equations 1 and 5. *Migration policies* define the placement of VMs to hosts. In the SL-Tight strategy, the algorithm considers the VMs' credit time and hosts' resource fragmentation.

To measure the energy consumption of a host there is the concept of *Power Model*. The model requires values defining the power consumption (W) of given hardware when the CPU level is at 0%, 10%, ..., 100%. In order to generate realistic results we conducted some experiments on a single server (Intel(R) Xeon(R) CPU E5-2620 v3), in which every 5 minutes the CPU usage level was raised by 10%. The results obtained are the following: $\{$0% = 45W/s, 10% = 65W/s, 20% = 75W/s, 30% = 80W/s, 40% = 85W/s, 50% = 88.33W/s, 60% = 90W/s, 70% = 95W/s, 80% = 95W/s, 90% = 96.66W/s, 100% = 100W/s$\}$. Such power model is used in our simulations.

Utilization models define the utilization patterns of VM's resources by cloudlets. In this work it was used the *full utilization model*, the *absolute utilization model* and the *normal utilization model*. In the full model, a cloudlet consumes all the allocated capacity, but in the absolute, it consumes a constant amount of resources. When following the normal model, a cloudlet consumes resources according to a normal distribution with specified mean and standard deviation.

5. Experiment Design

To evaluate the improvements brought by the SL-Tight strategy, we define two scenarios: i) baseline scenario and ii) impacts of resource fragmentation. Roughly, we want to understand the impacts of the SL-Tight on energy consumption, on workload running time, on VMs' SLO compliance rate, and on the allocated resource fragmentation.

There are some simplifications common to both scenarios. All hosts belong to a single datacenter and provision resources with the *simple provisioner*, except for FM hosts, which provision IOPS through a *resource provisioner shared* to reduce resource fragmentation. Besides, hosts are set up with space shared policies for all the resources except for the CPU – this means that a host accepts more VMs than its CPU capacity could handle, but each VM has a portion of the time of the CPU. In other words, the CPU availability never precludes a host from accepting VMs, but the other resources (RAM, IOPS, and bandwidth) do. VMs schedule cloudlets with space shared policy, and we set up each cloudlet requirement so that VMs can handle only a single cloudlet. Further, each cloudlet is set to finish after a 30-day interval if it stays in an RM host. When placed in an FM host, it can finish earlier, if it consumes more resources than in the RM host (this depends on the workload utilization model), or later, if the resource contention precludes it from consuming as many resources as it was able to consume in an RM host.

In both designs, the scaling parameters are fixed in $\mathcal{S}_{min} = 0.9$, $\mathcal{S}_{max} = 0.99$ and $\Delta = 0.01$ – an aggressive setting that tries to avoid VMs to have more than 10% of allocated resource fragmentation. We try out the most commonly used service levels, $\kappa_{min} \in \{0.999, 0.9995, 0.9999\}$. Finally, for the sake of simplification, the migration time is fixed in $\lambda = 30$s, and the safety margin is fixed in $\sigma = 10$s.

5.1. Baseline Scenario

For this setup, the infrastructure is comprised of two hosts and two VMs. We experiment scenarios with SL-Loose and SL-Tight to check how the SLA exploitation strategy performs in comparison to standard approaches. VMs are set with 100 IOPS SLO requirements, and they are launched with the worst fit allocation policy. Hosts are set with 150 IOPS capacity, and thus, the RM host can host a single VM, while the FM can host two VMs. What differs in each case is the IOPS utilization model each cloudlet is set up.

5.1.1. Case I

One cloudlet is set with the full utilization model, and thus it consumes all the 100 IOPS allocated to the VM. Meanwhile, the other cloudlet is set with an absolute utilization model of 25 IOPS. Such setup allows host consolidation and, in addition, it optimizes the workload performance in the FM host since the cloudlet with the full model will leverage the idle resources of the other one.

5.1.2. Case II

Both cloudlets are set up with an absolute utilization model of 76 IOPS. Therefore, although an FM server can host both VMs, they will always be experiencing momentary SLO violations. Our goal here is to check whether the SL-Tight breaks the overall SLA.

5.2. Impacts of Resource Fragmentation

In this design, we assess the impacts of the SL-Tight approach on VMs with different allocated resource fragmentation. Thus, a more significant setting is considered – the datacenter is comprised by 10 hosts and 100 VMs evenly distributed among hosts. Half of this set of VMs are configured with a *normal utilization model* with mean equal to 25 and the other half with mean on 75, all of them with a standard deviation of 5.

Each host has 1000 IOPS capacity and 20GB of RAM, but each VM requests only 100 IOPS capacity and 1GB of RAM. Therefore, the FM hosts can allocate twice as many VMs than RM hosts, since the IOPS is allocated with a shared provisioner. In this case, the unique constraint in an FM host would be the RAM, because it uses a simple provisioner.

Another factor we want to evaluate is the impact of letting more hosts being turned into FM hosts ($\mathcal{L}_{max}$). Whenever the SL-Tight is enabled, $\eta = 11$, meaning that 11 VMs with enough time credit to migrate can trigger the transformation of an RM into FM. Recall also that when an RM host becomes FM its IOPS provisioner is set to shared instead of simple, and all VMs on this host try to upscale/downscale when overloaded/underloaded. For this design we ran experiments with $\mathcal{L}_{max} \in \{1, 3\}$. For simplicity, $\kappa_{min} = 0.999$. Finally, we try out a narrow and a wide safety margin: $\sigma \in \{10, 10 + \lambda\}$.

6. Results and Analysis

6.1. Baseline Case I - saving energy and improving workload performance

Figure 2 shows the energy consumption of the datacenter over time, with SL-Tight and SL-Loose strategies, and with different SLO levels (κ_{min}).

In this experiment, each server could host 2 VMs. When hosting a single VM, each server was using half of its CPU capacity, consuming 88.33W/s. This situation is depicted by the continuous red line (SL-Loose). When the SL-Tight is enabled (dashed blue line), one host uses its full CPU capacity consuming 100W/s and the other one is powered off. In addition, consolidation (steep drop of dashed line) is achieved faster for lower κ_{min} values. This happens because the higher is κ_{min}, the lower will be the amount of time credit a VM accumulates per second. In this case, all VMs were committing no SLO violation once there is no resource contention, and thus they become eligible to migrate after 0.51, 1.02 and 5.14 days, for $\kappa_{min} \in \{0.999, 0.9995, 0.9999\}$.

Also, in the last five days there is a further decrease in energy consumption. This happened because the VM with full IOPS utilization (VM A) was placed together with the VM with absolute

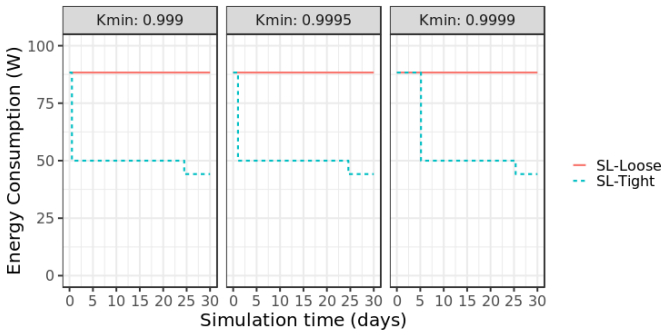

Figure 2. Case I: energy consumption of datacenter.

utilization of 25 IOPS (VM B) in the FM host. In the FM host, the scaling enables VM B to reduce its IOPS allocation from 100 to 25 without compromising its performance. This action allows VM A to improve its performance from 100 (in the RM host) to 125 IOPS (in the FM host), leading to a faster workload termination (5 days earlier). Considering the average US energy cost of \$0.1042 per kWh [U.S. Energy Information Administration 2019], the SL-Loose approach would cost \$66.26 (635.97 kWh) and the SL-Tight could reduce it up to \$38 (368.52 kWh) for $\kappa_{min} = 0.999$, an economy of 42.6%.

6.2. Baseline Case II - SLO analysis

Figure 3 shows the SLO compliance rate of VMs that always commit SLO violations when placed in the FM host.

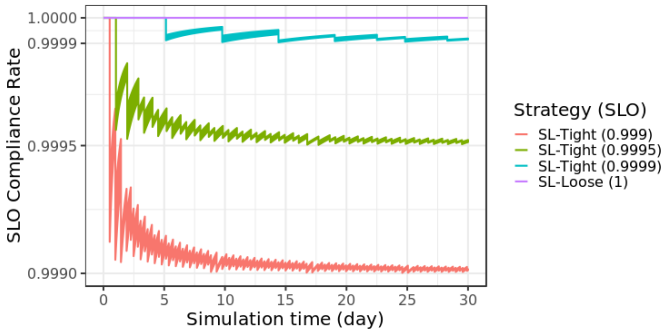

Figure 3. Case II: SLO compliance rate of VMs.

Case II shows it is possible for VMs to commit SLO violations but still comply with the SLA. The SLO compliance rate of all three κ_{min} never gets lower than the contracted. This is achieved by migrating VMs about to break their SLAs from FM hosts to RM hosts. Note that the steep drop of lines in Figure 3 denotes the SLO violation of VMs in the FM host, while the increase stands for their stay on the RM host. Finally, we may also observe that VMs with higher κ_{min} increase their credit time slower, and for this reason, they migrate to FM host less frequently. Thus, a shortcoming of FM servers hosting VMs in high resource contention scenarios is that VMs with low κ_{min} will experience a high number of migrations.

6.3. Impacts of Resource Fragmentation

Figure 4 presents the minimum, median and maximum SLO compliance rates of VMs with workload utilization mean equal to 75 IOPS, for scenarios with $\kappa_{min} = 0.999$, $\sigma \in \{10, 40\}$ and for

$\mathcal{L}_{max} \in \{1, 3\}$. Since all VMs with workload utilization mean equal to 25 IOPS committed no SLO violation for having low levels of demand, they are not included in the plot so that the details of results of the remaining VMs do not get masked due to summarization.

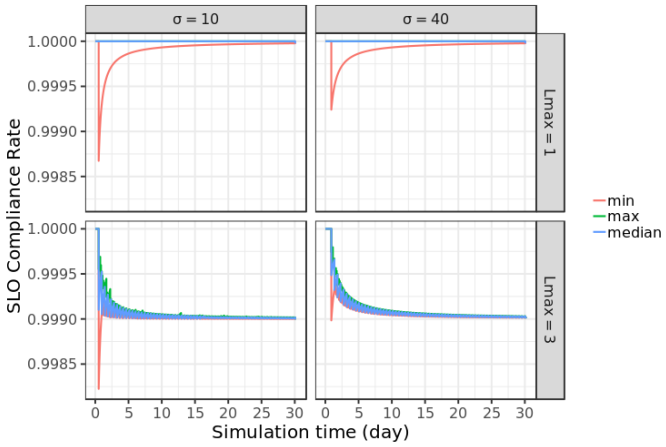

Figure 4. SLO compliance rates of VMs with mean IOPS equals to 75.

The first conclusion we can draw from Figure 4 is that depending on the parameters there may happen a contract breach. However, just a few VMs violate their SLAs and for a tiny period of time: 5, 8 and 6 VMs out of 100 VMs, for $(\sigma = 10, \mathcal{L}_{max} = 1)$, $(\sigma = 10, \mathcal{L}_{max} = 3)$ and $(\sigma = 40, \mathcal{L}_{max} = 3)$, respectively.

Even though there may happen brief SLA violations, some procedures can mitigate this issue. IaaS providers could conceive their agreements with some service credit mechanism [Muller 1999] where the customer accumulates service credit whenever the provider fails to comply with the agreed SLO, credits which could later be spent. Another mitigation is using conservative values for σ and $\mathcal{L}_{max}$, which is discussed next.

By comparing the columns of Figure 4 one can note that increasing the safety margin σ has a positive impact on the SLO compliance rate of VMs indeed— the higher is σ, the lesser is the VM stay in the FM host. Oppositely, the higher is $\mathcal{L}_{max}$, the higher are the risks of SLO violations. Note that when $\mathcal{L}_{max} = 1$ only 19 VMs experienced the FM host while for $\mathcal{L}_{max} = 3$ all the VMs have been in an FM host. This is why, for $\mathcal{L}_{max} = 3$, the maximum values got closer to the median and min SLO compliance rates.

Table 2 presents the energy cost and consumption of each scenario. The SL-Loose strategy cost \$333.38 (3199 kWh).

Table 2: Energy cost and consumption of the datacenter.

	$\sigma = 10$	$\sigma = 40$
$\mathcal{L}_{max} = 1$	\$304 (2920 kWh)	\$304 (2923 kWh)
$\mathcal{L}_{max} = 3$	\$256 (2462 kWh)	\$275 (2641 kWh)

In terms of energy, increasing $\mathcal{L}_{max}$ has higher economic impacts than decreasing σ. The most economical scenario is the one set up with $\sigma = 10$ and $\mathcal{L}_{max} = 3$, an economy of 23%.

Figure 5 shows the aggregated allocated resource fragmentation of all VMs in all hosts for each scenario. Since the allocated resource fragmentation has high variability, we ran a moving

average with a two-hour sample window. Here we normalize the allocated resource fragmentation of each VM dividing it by the VM's resource capacity.

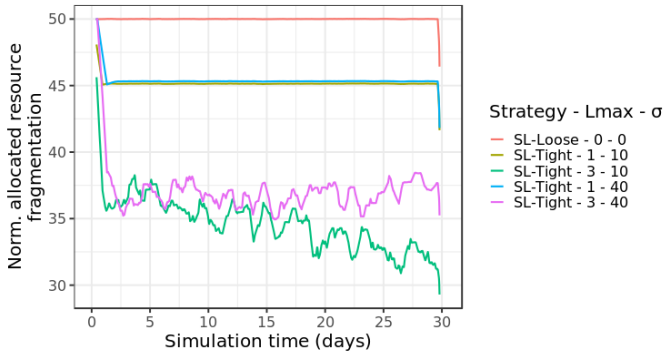

Figure 5. SLO compliance rates of VMs with mean IOPS equals to 75.

As expected, the average allocated resource fragmentation for the SL-Loose is 50, since half of VMs consumes 25% of their capacity and the other half consumes 75%. The SL-Tight approach with $\mathcal{L}_{max} = 1$ decreases on average 5% of the allocated resource fragmentation, and σ has a negligible impact in this case. However, for $\mathcal{L}_{max} = 3$ the value of σ has a considerable impact: when $\sigma = 40$ the allocated resource fragmentation was reduced to an average of 37%, and for $\sigma = 10$ the average allocated resource fragmentation is 34.6%.

7. Related Work

Two well-known IaaS providers use a similar scheduling strategy: AWS and Google. In AWS, users can lease VMs with different service levels (on-demand, reserved, and spot instances) [Amazon Web Services 2019], and Google maintains a cluster manager named Borg in which jobs with different priority levels can be scheduled (production, batch, and best effort) [Verma et al. 2015]. The idea is that VMs or jobs with lower service levels can be preempted to give room for more profitable ones. Although such a strategy increases the income, it does not exploits the SLAs as our strategy proposes.

Carvalho et al. (2017) state that there are three aspects to optimize the number of VMs allocated: capacity planning, admission control, and scheduling. They propose a capacity planning method that employs analytical models over the workload experienced in the past to find the minimum capacity required to meet availability SLOs in the future. The creation of multiple service classes allows better use of resources, yielding higher profitability. Instead of proposing new SLAs, our strategy exploits existing SLAs by reducing resource fragmentation through server rebalancing, allowing consolidation.

Svärd et al. (2015) rely on the fact that VM scheduling is an online problem, yielding suboptimal results over time, to propose a continuous datacenter consolidation heuristic. In their approach, any VM arrival/exit event triggers a set of algorithms that use VM profit as a priority level to decide which VMs should be migrated or suspended to rearrange the datacenter. Silva et al. (2019) also use a priority strategy, but taking the SLOs and availability delivered for each VM into account when making decisions. Although Silva's approach is similar to ours, the key difference is the strategy we use to push the experienced SLO to the contracted one — they suspend VMs while we migrate VMs to hosts with overbooking.

In the taxonomy proposed by Bittencourt et al. (2018), this work's scheduler can be classified as having the objective of *minimizing datacenter costs* in the provider perspective. However,

such a taxonomy can be augmented with a "SLA exploitation" category, in which this work and [Silva et al. 2019] would fit.

8. Concluding Remarks

Consolidation heuristics has been extensively adopted to reduce energy consumption by datacenters. However, only Silva et al. (2019) does it from the perspective of SLA amortization. Keeping SLO compliance rates far above the minimum agreed is a waste of resources. This slack could then be leveraged for reconfiguring the system into more efficient configurations. In this work, we propose a strategy to exploit VMs' SLA based on Economic Systems. VMs join the Free Market to amortize their SLOs rate and migrate back to a Regulated Market when the SLA is at risk.

With the aid of a simulation model, it was observed that such a strategy can lead to energy savings and improve workload performance. In scenarios in which VMs have different resource fragmentation, the SLA exploitation strategy may lead to SLA contract breach depending on the parameter setup. An aggressive setup caused punctual SLA violations for 8% of VMs but produced an economy of 23% in energy consumption. On the other hand, a conservative setting caused no SLA violations and produced an energy saving of 8.6%.

Although simulations show positive results, some aspects are yet to be investigated. For instance, one could analyze the number of migrations and devise heuristics to optimize this task. A VM classifier could monitor the VMs' resource utilization and try to forecast the migration time. Thus, upon classification, some VMs could be discarded for migration. Another idea is creating different levels of risk environments by controlling the level of overbooking in a given host.

Acknowledgment

This project was developed by UFCG under the scope and support from Lenovo Brazil with incentive of the Brazilian Informatics Law (Brazilian Federal Law #8248/91).

References

[Alahmadi et al. 2014] Alahmadi, A., Alnowiser, A., Zhu, M. M., Che, D., and Ghodous, P. (2014). Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud. In *Proc. of the 2014 Intern. Conference on Computational Science and Computational Intelligence - Volume 02*, CSCI '14, pages 69–74. IEEE Computer Society.

[Amazon Web Services 2019] Amazon Web Services (2019). Amazon EC2 Service Level Agreements. `https://aws.amazon.com/ec2/sla/?nc1=h_ls`.

[Bittencourt et al. 2018] Bittencourt, L. F., Goldman, A., Madeira, E. R., da Fonseca, N. L., and Sakellariou, R. (2018). Scheduling in distributed systems: A cloud computing perspective. *Computer Science Review*, 30:31 – 54.

[Carvalho et al. 2017] Carvalho, M., Menascé, D. A., and Brasileiro, F. (2017). Capacity planning for IaaS cloud providers offering multiple service classes. *Future Generation Computer Systems*, 77:97 – 111.

[Filho et al. 2017] Filho, M. C. S., Oliveira, R. L., Monteiro, C. C., Inácio, P. R. M., and Freire, M. M. (2017). Cloudsim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 400–406.

[Hameed et al. 2016] Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P. P., Kolodziej, J., Balaji, P., Zeadally, S., Malluhi, Q. M., Tziritas, N., Vishnu, A., Khan, S. U., and Zomaya, A. Y. (2016). A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing*, 98(7):751–774.

[Helms 2005] Helms, M. M. (2005). *Encyclopedia of Management*. GALE Group, 1st edition.

[Huberman and Hogg 1995] Huberman, B. A. and Hogg, T. (1995). Distributed computation as an economic system. volume 9, pages 141–152.

[Ibidunmoye et al. 2015] Ibidunmoye, O., Hernández-Rodriguez, F., and Elmroth, E. (2015). Performance anomaly detection and bottleneck identification. *ACM Comput. Surv.*, 48(1):4:1–4:35.

[Lee and Zomaya 2010] Lee, Y.-C. and Zomaya, A. (2010). Energy efficient utilization of resources in cloud computing systems. 60:268–280.

[Microsoft Azure 2019] Microsoft Azure (2019). SLA for Virtual Machines. `https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_8/`.

[Muller 1999] Muller, N. J. (1999). Managing service level agreements. *International Journal of Network Management*, 9(3):155–166.

[Shehabi et al. 2016] Shehabi, A., Smith, S., Horner, N., Azevedo, I., Brown, R., Koomey, J., Masanet, E., Sartor, D., Herrlin, M., and Lintner, W. (2016). United states data center energy usage report, berkeley, california. Technical report.

[Silva et al. 2019] Silva, G., Lopes, R., Brasileiro, F., Carvalho, M., Morais, F., Mafra, J., and Turull, D. (2019). Escalonamento justo em infraestruturas de nuvem com múltiplas classes de servico. In *Proc. of the 2019 Brazilian Symposium of Computer Networks and Distributed Systems*.

[Song et al. 2014] Song, W., Xiao, Z., Chen, Q., and Luo, H. (2014). Adaptive resource provisioning for the cloud using online bin packing. *IEEE Transactions on Computers*, 63(11):2647–2660.

[Svärd et al. 2015] Svärd, P., Li, W., Wadbro, E., Tordsson, J., and Elmroth, E. (2015). Continuous datacenter consolidation. In *IEEE 7th International Conference on Cloud Computing Technology and Science*, pages 387–396.

[U.S. Energy Information Administration 2019] U.S. Energy Information Administration (2019). Electric Power Monthly. `https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=epmt_5_6_a`. Online; accessed December 2019.

[Usmani and Singh 2016] Usmani, Z. and Singh, S. (2016). A survey of virtual machine placement techniques in a cloud data center. *Procedia Computer Science*, 78:491 – 498. 1st Intern. Conference on Information Security I& Privacy 2015.

[Verma et al. 2015] Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., and Wilkes, J. (2015). Large-scale cluster management at google with borg. In *Proc. of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 18:1–18:17, New York, NY, USA. ACM.