An Autonomic Approach to Risk-Aware Data Center Overbooking

Luis Tomás and Johan Tordsson
Department of Computing Science, Umeå University, Umeå, Sweden
{luis,tordsson}@cs.umu.se

Abstract—Elasticity is a key characteristic of cloud computing that increases the flexibility for cloud consumers, allowing them to adapt the amount of physical resources associated to their services over time in an on-demand basis. However, elasticity creates problems for cloud providers as it may lead to poor resource utilization, specially in combination with other factors, such as user overestimations and pre-defined VM sizes. Admission control mechanisms are thus needed to increase the number of services accepted, raising the utilization without affecting services performance. This work focuses on implementing an autonomic risk-aware overbooking architecture capable of increasing the resource utilization of cloud data centers by accepting more virtual machines than physical available resources. Fuzzy logic functions are used to estimate the associated risk to each overbooking decision. By using a distributed PID controller approach, the system is capable of self-adapting over time – changing the acceptable level of risk – depending on the current status of the cloud data center. The suggested approach is extensively evaluated using a combination of simulations and experiments executing real cloud applications with real-life available workloads. Our results show a 50% increment at both resource utilization and capacity allocated with acceptable performance degradation and more stable resource utilization over time.

Index Terms—Admission Control, Overbooking, Resource Utilization, Scheduling, Control Theory, Fuzzy Logic Programming

1 Introduction

One of the main features provided by clouds is *elasticity*, which allows users to dynamically adjust resource allocations depending on their current needs. Thus, customers only have to pay for what they use. Although this characteristic is one of the main advantage from the client viewpoint, it may have an impact on the cloud infrastructure providers side: capacity planning becomes challenging when the exact number of Virtual Machines (VMs) that each running service is going to need in the future is unknown [1]. Consequently, if the objective is to make an efficient use of available resources, overestimating the required capacity results in poor resource utilization and lower income from consumers. On the contrary, underestimating may lead to performance degradation and/or crashes.

Additionally to elasticity issues, there are other factors contributing to lower the cloud data center utilization: (1) cloud providers commonly only offer predefined VM sizes, which have fixed amount of CPU, memory, disk, etc.; (2) cloud applications do not use the same amount of hardware resources all the time; and (3) users are usually bad at estimating the requirements of their applications. This low resource utilization is a big concern for cloud data center providers as data centers consume lot of energy and are being used in a rather inefficient way – energy consumption does not decrease linearly with resource usage. One way cloud providers can mitigate these resource utilization problems is by overbooking. However, overbooking techniques always expose the infrastructure to a risk of resource congestion upon unexpected situations and consequently to SLA violations. The main challenge is *how to decide the appropriate level of overbooking that can be achieved without impacting the performance* of the cloud services [2]. Admission control techniques are thus needed to handle this trade-off between increase of resources utilization and risk of performance degradation, determining whether a new service should be admitted into the data center by evaluating the associated long term risk.

Our initial work on this problem include scheduling for better server utilization [3] and admission control for capacity planning [4], getting an initial understanding of the overbooking problem and the risk evaluation, respectively.

Based on that acquired knowledge we present here a risk-aware autonomic framework for resource overbooking within cloud data centers to increase resource utilization in a safe and balanced way. In a nutshell, the system autonomously (re)adjusts depending on how stable and predictable the system and current workload are, facilitating a quicker reaction to unpredicted situations and a faster response to possible performance degradations. The system automatically adapts to different data center sizes, type and knowledge of the new incoming workload, etc., and achieves a more balance utilization among the different capacity dimensions (CPU, memory, I/O) thanks to the use of proportional-integral-derivative (PID) controllers [5] that adjust the suitable level of risk to safely achieve the desired level of resources utilization. A PID controller is a generic control loop feedback mechanism that calculates the difference between a measured variable (in our case resource utilization) and the desired set point and attempts to minimize it by readjusting the control inputs (the

risk thresholds). It involves three separate parameters corresponding to the present error (P), the accumulated error (I), and the prediction of future errors (D).

The main contributions of this paper are:

- design and tuning of a PID Controller for controlling the servers behavior,
- use of a distributed PID controller that changes the level of risk that the data center is able to face depending on how the system is behaving,
- a method to select the most representative server in to the data center to accordingly establish the risk thresholds accepted by the admission control,
- and evaluation of our proposal using real workloads (web servers).

2 Data center utilization problems

Achieving efficient utilization of hardware resources has been a research goal since long in data centers, and involves techniques at different scales: single processors (such as prefetching techniques [6]), networks on chip (NoCs) (such as virtualization for NoCs [7]), routing algorithms [8], network multiplexing [9], or time sharing systems [10]. These efforts are not only centered on increasing the utilization but also to keep a more balanced usage among nodes [11], and/or delivering fairness among users, projects or virtual organizations [12].

For data centers, the efficient use of resources is mainly motivated by hardware and operational costs [13], and lately also by power consumption and environmental concerns [14], becoming a critical issue for large scale data centers. The very large scale and multi-tenant nature of cloud infrastructures offers great potential for efficient multiplexing of different services and applications, allowing a much higher resource utilization [15]. The cloud paradigm also introduces new obstacles for efficient resource management. There are several recent studies that highlight the overall low utilization in those cloud data centers. One major example is the analysis of Google traces [16], which concludes that only 53% of the available memory is used whereas CPU utilization is as low as 40% on average. Similarly, Barroso et al. [17] report 10-50% as common levels for CPU utilization in a study of 5000 servers observed along 6 months.

Elasticity of applications is one of the most notably characteristics of cloud infrastructures but at the same time one of the more challenging regarding resource management. VMs belonging to a cloud service can dynamically be (de)allocated based on changes in service workload [1]. This *horizontal elasticity* or scale up/down is a common pattern in interactive multi-tier services, where the amount of resources needed by the service strongly depends on the number of users currently served. However, from a cloud provider perspective, deciding how much resources to dedicate to an elastic service constitutes a challenging problem, since not even the service owner knows what amount of resources may need at certain time into the future. If the cloud provider accepts a new service by estimating that is going to need a certain capacity (variable over time) and finally the service is using less than expected, the infrastructure is underutilized as some resources were reserved for that service but never used. On the other hand, if the service needs more capacity than estimated, the service or data center can get too congested and start to behave poorly. This has also a significant impact on resource utilization on one hand, and potential Service Level Agreements (SLAs) violations on the other.

Another aspect that significantly affects cloud data centers utilization is *vertical elasticity*. This utilization problem stems from the fact that cloud applications are encapsulated into VMs with a fixed amount of hardware resources (CPU, memory, I/O, etc.), whilst applications do not use the same amount of them all the time. This can either be related to different behavior due to internal phases of application operation, or be caused by variations in incoming workload. Therefore, users need to provision the worst-case capacity as the upper bound, even though that amount of CPU, memory, or I/O is only used for a short period of time.

In addition to elasticity, there are other issues that reduce resource utilization. Cloud providers usually offer a bunch of pre-defined VMs sizes with a specific (fixed) amount of resources, such as CPU, disk, memory, etc., that the customer is not allowed to customize. This *T-shirt problem* [18] forces users to choose a VM with enough of the most critical resource (CPU, memory, etc.) while over-provisioning other resources. Additionally, users tend to over-provision their needs for the sake of safest executions, aggravating the waste of resources problem. A related study of parallel computing workloads evidence this fact by showing that more than half of all jobs use less than 20% of users requested capacity [19]. These utilization problems also lead to an imbalance usage of the different capacity dimensions even if nominal usage is totally balanced.

3 Overbooking

The above listed cloud infrastructure characteristics, in combination with users' tendency to over-provision resources, significantly impact the overall resource utilization, as reviewed in [16]. However, data centers can also take advantage of those characteristics to accept more VMs than the number of physical resources the data center allows. This is known as *resource overbooking* or resource overcommitment. More formally, overbooking describes resource management in any manner where the *total available capacity is less than the theoretical maximal requested capacity*. This is a well-known technique to manage scarce and valuable resources that has been applied in various fields since long ago. Well known examples include network bandwidth allocation [20] and batch scheduling for parallel computers [21]. As cloud environments regards, overbooking of resources appears as a suitable solution to increase utilization as machines

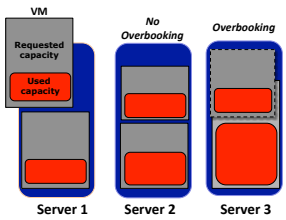


Fig. 1: Conceptual illustration of overbooking showing how two small enough applications (encapsulated within their respective VMs) can be collocated without disturbing each other.

may not be fully utilized even when the total capacity requested (not the real used) exceeds the total capacity.

Figure 1 shows a conceptual overview of cloud overbooking, depicting how two virtual machines (gray boxes) running one application each (red boxes) can be collocated together inside the same physical resource (Server 1) without (noticeable) performance degradation. This is possible as long as the total real capacity being used by both applications is less than the real capacity available taking into account all capacity dimensions: CPU, memory, I/O, etc.

In order to address the three main issues affecting utilization, a two-pronged overbooking strategy is applied. At a higher level, we propose *admission control* strategies that allow the acceptance of more VMs than the actual data center capacity. This mainly addresses the horizontal elasticity issues but also helps other issues – T-shirt problem, vertical elasticity and users overestimations. At lower level, we propose *scheduling* techniques to decide which VMs are better to collocate. These techniques mostly focus on T-shirt and vertical elasticity problems. Therefore, admission control techniques are used for long-term capacity planning, whilst scheduling techniques are focused on avoiding performance degradation due to short-term workload fluctuations.

3.1 Challenges

However, increasing resource utilization in cloud data centers by including overbooking entails a list of challenges that must be addressed: dealing with many different sources of information, inaccurate or missing information, uncertainty in predictions, trade-offs between resource utilization and risks being taken, application-specific impact of overload situations, etc.

There is a vast amount of critical information to be considered when performing overbooking actions, such as hardware available in the data center, the set of currently admitted services and their variation in resource usage over time, available information about new service requests, etc. Taking all that information into account in resource management is a challenging problem even in absence of overbooking.

In such overbooking scenarios, admission control precision greatly impacts scheduling decisions and performance: accepting too few or too many VMs in the data center makes either makes scheduling trivial or it becomes too hard (if not impossible) to find a good allocation, respectively.

The impact of potential overload situations is rather hard to assess and application-specific. For instance, high throughput applications and real time applications behave very differently and should be treated in different ways. To exemplify, for MapReduce services, high bandwidth and throughput may be preferred (reducing the shuffling and execution phase, respectively), whereas for interactive services, low latency and response times are the key performance indicators. Moreover, most applications run, albeit slower, if allocated too little CPU, whereas provisioning too little memory is prone to make applications crash.

Furthermore, some applications are more suitable of being collocated with other VMs. For instance, workloads presenting bursty (peaky) behavior are more suitable to be overbooked as highlighted in [22]. They only occasionally use all the resources that they are entitle to. However, the involved risks are also bigger as the system has to avoid that peaks of one workload coincide with those of other workloads. As VMs are multidimensional (CPU, memory, I/O), while one application is using (more intensively) one resource type (such as CPU), other may be using another, e.g., memory. They can thus be collocated without disturbing each other. As an example, collocating CPU-bound and network-bound or memory-bound applications [23], [24] is of great advantage.

All in all, the main challenge for overbooking is to decide how much excess capacity is appropriate to allocate to minimize the risk of performance degradation [2].

4 Related Work

This work primarily focuses on how to increase resource utilization at cloud data centers by a combination of admission control, scheduling and prediction techniques. A brief overview of the related work in these fields is given next.

4.1 Overbooking

We propose overbooking techniques as a solution to poor resource utilization in cloud data centers. The overbooking concept is not new and has been studied and applied in various fields since long ago [25]. Examples of this diversity are bandwidth allocation [20], airline yield management [26] and parallel computer scheduling [21]. Within data center management, Urgaonkar et al. [23] try to safety overbook cluster resources. This is performed by guaranteeing applications performance using feedback control. Their work however assume that users are capable of providing information regarding the degree of overbooking that their applications may tolerate. This information is strongly coupled to the underlying physical infrastructure, as well as the other applications that are to be collocated. Our experiments

with RUBiS and RUBBoS described in Section 7 illustrate this difficulty.

Additionally, the utilization problem caused by cloud infrastructure vendors only offering fixed size VMs (the t-shirt problem) also increases the needs for overbooking. This problem is illustrated by Gmach et al. [18]. They demonstrate that for a given set of workloads, if vendors use the t-shirt model almost twice the number of physical servers is required compared to the time share model – without using resource overbooking.

4.2 Admission control

In order to solve these utilization problems, admission control techniques that increase resource utilization are needed. However, admission control within cloud environments is not trivial due to different reasons, specially the elastic nature of services and uncertainty about future needs/behaviors. Among recent approaches to this multi-faceted problem, Konstanteli et al. [27] present a probabilistic method for a combined scheduling and admission control problem formulated using mixed-integer non-linear programming. They model the elastic service demand by using cumulative distribution functions.

There are also various approaches for admission control at application level. For instance, Leontiou et al. [28] propose an adaptive feedback scheme with an application queue model to prevent overload of cloud services. Kjaer et al. [29] use an online feed-forward control system for web server resource allocation avoiding performance disturbances rejecting individual requests. Ashraf et al. [30] propose a combination of noise-filtering and load predictions for session-based admission control in multi-tier servers. Nevertheless, unlike our work, these efforts are not focused on increasing data center utilization but at preserving applications performance.

4.3 Scheduling with collocation

Once the services are accepted in a data center, a scheduler is in charge of finding a suitable physical resource to allocate them. In that process, it is decided which VMs are suitable to be collocated for improved utilization and stable performance. On this topic, He et al. [31] present a multivariate probabilistic model where physical hosts for VMs are selected based on three capacity dimensions (CPU, memory, and I/O). However this approach is centered on replaning the scheduling decisions rather than overbooking the resource usage. A similar approach is presented by Meng et al. [15], who propose a joint VM provisioning approach that allocates and consolidates VMs based on estimates of the aggregated VM capacity requirements. However, their work only takes into account CPU usage and assumes perfect predictions about future workload behavior. Another more recent example of consolidation at cloud data centers is presented by Mastroianni et al. [32]. Unlike the previous work, it does not only consider CPU resources but also RAM usage. Similarly to our proposal, they try to keep a balanced and efficient usage of CPU and memory resources. Nevertheless, we also consider the network capacity dimension and focus on the admission control problems. Therefore, both approaches could complement each other.

There are other overbooking approaches that consider traffic localization in order to reduce the network bandwidth consumption when collocating VMs, such as the one proposed by Wo et al. in [33]. They present a greedy approach that perform traffic-aware VM placement to increase the rate of accepted requests. However they focus on a revenue optimization model that assumes linear performance degradation in overloaded situations. On this topic it is useful to use (anti)affinity rules as proposed by Larsson et al. in [34] or Broeckhove et al. [24]. The main aim of these works is to avoid repeating poor performance and to increase the chances of good collocations when there is some meta-knowledge about the VMs to collocate – whether due to previous collocation or known issues about their performance. A recent example of an over-commit scheduler is PULSAR [35] that extends OpenStack Nova filter scheduler to allow overcommitment actions. It varies the overcommitment ratio over time in an adaptive way depending on the workloads behavior, but only considering CPU.

4.4 Prediction methods

Every overbooking system (as well as admission control techniques when dealing with elastic services) need insight in future resource usage and service requirements to avoid performance degradation due to overloaded physical resources. The literature on resource behavior modeling and predictions within highly distributed systems, such as grids and clouds, is very rich. A survey of several prediction techniques is presented in [36]. Examples of these techniques include adaptive methods [37], state-space models [38], exponential smoothing [39], and use of control schemes for self-tuning for improved forecasts [40].

4.5 Risk assessment and SLA management

In addition to resource usage predictions, admission control techniques need risk assessment methods to take their decisions, regardless of the overbooking. Just like overbooking techniques, risk evaluation has been studied since long. There are examples of risk evaluation and SLA management applied to grids, such as work by Djemame et al. in [41], where the probability of SLA failure is evaluated and used in the negotiation process.

Risk assessment has also been used within consolidation environments, such as the work presented by Verma et al. [42]. They presented two consolidation methods to achieve power savings, which try to reduce the risk of performance degradation by considering the correlation between collocated applications.

There are also more recent studies centered on evaluating the risk of resource overbooking in cloud environments. Ghosh et al. analyze these risks and based on

these a threshold-based overbooking schema is proposed to estimate the level of overbooking achievable, but only CPU is taken into account [22].

Others study the trade-off between overbooking and risk of performance degradation by closely relating these to SLA management. Breitgand et al. [43] present an extension to standard available SLAs that also includes the probability of successfully deploying additional VMs. Their framework uses cloud effective demand as a metric to estimate the total physical capacity required for performing the overbooking, but unlike our work, this framework is only based on CPU usage. Another example of admission control techniques based on SLAs is presented by Wu et al. [44], who propose admission control and scheduling algorithms for Software as a Service (SaaS) providers to efficiently use Infrastructure as a Service (IaaS) providers to maximize the profit whilst at the same time improving customer satisfaction. Their work does not focus on improving the resource utilization (IaaS level) but to use the cloud provider that best suits the current needs of the service provider (SaaS level), minimizing the needed budget and maintaining the performance. Their work complements ours as the two approaches operate at different levels.

4.6 Recovery methods

Neither predictions nor risk assessment can be exactly accurate in all scenarios in cloud environments. For this reason, recovery and autonomic readjusting mechanisms are needed when problematic overload situations occur (reactive) or to avoid them before they happen (proactive). On this topic, Beloglazov et al. [45] propose a Markov chain model and a control algorithm for detecting overloading problems in physical servers as a part of a dynamic VM consolidation. In case of a problematic situation arises, the system migrates some VMs to less loaded resources. Recovery methods are also used to avoid SLA violations, as presented by Bobroff et al. [46]. They proposed a pro-actively virtual machine migration algorithm based on time series forecasting and bin packing heuristics. However, unlike us, they do not focus on the admission control problems and only consider CPU utilization. Another example is the Sandpiper engine [47] which detects overloaded nodes and performs the needed migration actions in order to reduce the performance degradation. There are other recovery methods based on control theory, such as service level awareness suggested by Klein et al. in [48]. In their work the application behavior is adjusted along time depending on the status of the system. Notably, such migration and service level changing approaches are not only an alternative to overbooking techniques but also a complement that may help avoiding performance degradation upon unexpected situations or mispredictions.

5 Overbooking Framework

To handle the data center resource utilization problems and overbooking challenges discussed, we have implemented an autonomic overbooking framework. Figure 2 shows a system overview of the framework. A preliminary version focusing on scheduling was presented in [3] where the basic scheduling and collocation capabilities were implemented but just with simple admission control methods. This work was later extended with a risk-aware admission control module [4], gaining some insight about parameters affecting overbooking, most prominently, target utilization.

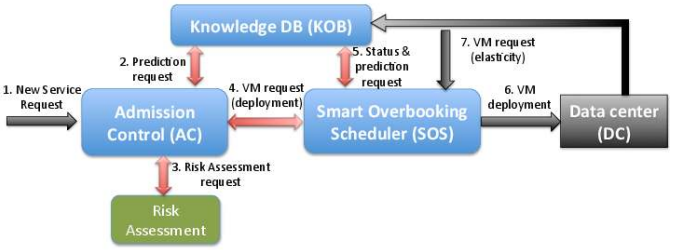


Fig. 2: Overbooking framework architecture highlighting overall operation.

In this work we focus on building an autonomic framework that provides better application performance, avoiding overpassing total capacity at any of the dimensions, and not only at the whole data center level but also in every single node into the system. In order to achieve that, besides using the risk assessment detailed in Section 6, a distributed PID controller approach (one controller at each server) is implemented to make the system fully autonomic. The system is now capable of keeping the utilization stable at each server in the data center, regardless of changing conditions (size of the data center, type and amount of cloud applications, VM sizes, etc.).

5.1 Admission Control

The Admission Control (AC) module is the cornerstone in the overbooking framework. It decides whether a new cloud application should be accepted or not, by taking into account the current and predicted status of the system and by assessing the long term impact, weighting improved utilization against the risk of performance degradation. To make this assessment, the AC needs the information provided by the Knowledge DB, regarding predicted data center status and, if available, predicted application behavior. This information is then injected into the Risk Assessment module, which by means of fuzzy logic programming [49], provides the information needed about the associated risk that would be taken in case of accepting the new service [50]. More information about the admission control algorithms is presented in next section (Section 6). Once the risk assessment returns the associated risk value of accepting the new service, the AC decides whether it is an acceptable risk or not.

5.2 Knowledge DB: Monitoring and Profiling Tools

The Knowledge DB (KOB) module measures and profiles the different applications' behavior, as well as the

Algorithm 1 Worst-Fit Overbooking Scheduling

Let TNC denote the Total Node Capacity and OBF the overbooking capacity
Let $R_{n_i}C_{used}$ = Real node n_i capacity in use
Let $margin$ = allowed overpassed capacity
$OBF = \frac{(UsageRequested - RealUsage)}{min(UsageRequested, RealCapacity)}$
Allocated = false
NS = Sort Nodes N by OBF
for each $n_i \in NS$ **do**
  if $Prediction(R_{n_i}C_{used}) + AppProfile < TNC + margin$ **then**
    Allocated = True
    AllocateVM at Node n_i
  end if
end for
if $Allocated == false$ **then**
  AllocateVM at Node with highest OBF
end if

resources' status over time. This module gathers information regarding CPU, memory, and I/O utilization of both virtual and physical resources. The KOB module has a plug-in architectural model that can use existing infrastructure monitoring tools, e.g., the Libvirt library [51] or Nagios [52], as well as shell scripts. These are interfaced with a wrapper that stores information in the KOB. The same tools can be used for application performance profiling, but more appropriately, more fine-grained monitoring should be used, such as LTTng2 [53].

In addition to interface with existing monitoring tools, a simulation and emulation module has been implemented in the KOB. Workload profiling has been made offline and based on this, resource executions are emulated in simulated servers. This enables reproducible experiments according to the workload profiles and allows experiments in larger environments. Moreover, the simulation and emulation model can be mixed with the real one, specifying which servers are to be simulated and which ones are real.

5.3 Smart Overbooking Scheduler

The Smart Overbooking Scheduler (SOS) allocates both the new services accepted by the AC and the extra VMs added to deployed services by scale-up, also deallocating the ones that are not needed [1]. As elasticity algorithms are outside the scope of this paper, the used autoscaling algorithms are just simple threshold-based methods that increase the number of VMs if the utilization of the running ones is over a certain threshold and vice versa for scaling down.

Basically, the SOS module selects the best node and core(s) to allocate the new VMs – based on the established policies. These decisions have to be carefully planned, specially when performing resource overbooking, as physical servers have limited CPU, memory, and I/O capabilities. Thus, some scheduling decisions may result in performance degradation.

Node selection is the first step to be performed by the scheduler. We presented in [3] a worst-fit algorithm that schedules a VM to the least overbooked server with the aim to improve overall utilization and reduce the overbooking impact. Algorithm 1 summarizes the steps to find a suitable server.

Algorithm 2 Admission Control

Let req be the incoming service request
Let $getFuzzyRiskAssessment$ the function that obtains the associated risk of accepting an app (see Section 6.1)
Let $Threshold[CPU, mem, IO]$ the current risk thresholds of the data center (see Section 6.2)
$App\_Risk[CPU, mem, IO]$ = $getFuzzyRiskAssessment(req)$
if $App\_Risk[CPU, mem, IO] <= Threshold[CPU, mem, IO]$ **then**
  Accept req
else
  Reject req
end if

Once the server where the service is to be deployed is selected, the next step is to find the best core for it. Without server overbooking this process is simply to find the idle resources. However, in the presence of overbooking, there are no, or too few, idle cores. Under such circumstances deciding what VMs to collocate in the same core(s) is of great important as some applications are more suitable for being collocated than others. The core selection is now managed by the KVM scheduler [54] but is to be improved in the near future with knowledge about VM affinity and interference.

6 Risk-Aware Admission Control

Admission control decisions must be carefully considered in order to achieve a high utilization without exposing the system to potential SLA violations. However, achieving this trade-off is far from being trivial. As highlighted in [23], taking overbooking actions by only considering average resource requirements or only one capacity dimension, e.g., CPU, can result in significantly reduced performance. Therefore, a long term evaluation is needed to better assess the future impact that the overbooking actions may entail.

How to estimate the risk for a given service request is unclear, specially under the uncertainty of future behavior of both currently deployed services and the new ones to be deployed. Based on this, we argue that fuzzy logic is a good method to evaluate the risk associated with a new incoming request. We hence implemented a risk assessment module based on fuzzy logic programming [49].

The basic functionality of the admission control is detailed in Algorithm 2. It first evaluates the risk associated to the new incoming request by calling the fuzzy risk assessment module. Once the associated risk is known, the admission control obtains the current (new) risk thresholds for the whole data center. Finally, it is checked, for each capacity dimension, if the risk of accepting the new incoming request is below the currently acceptable level and if so, the request is accepted. The process to calculate the service acceptance risk and the data center risk thresholds is explained next.

6.1 Fuzzy Risk Assessment

The risk assessment module provides the Admission Control with the information needed to take the final

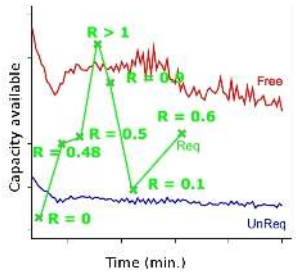


Fig. 3: Illustration of risk over time.

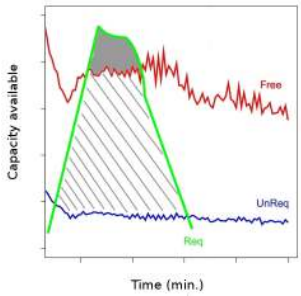


Fig. 4: Area calculation for peak risk assessment.

decision of accepting or rejecting the service request, as a new request is only admitted if the final risk is bellow a pre-defined level (*risk threshold*).

A generic metric for the risk of accepting a new service has been previously presented in [4], [50]. This metric forms the basis for a set of different admission controls algorithms, enabling the long term evaluation of overbooking actions under uncertain conditions. The inputs for this risk assessment module are:

- Req - CPU, memory, and I/O capacity required by the new incoming service.
- $UnReq$ - the difference between total data center capacity and the capacity requested by all running services.
- $Free$ - the difference between total data center capacity and the capacity used by all running services.

The values for $Free$ and $UnReq$ are predicted from previous monitored utilization using exponential smoothing [55]. For the Req, the values are obtained from previous knowledge of the service (if possible) or by taking into account the information specified by the cloud customer. Consequently, if there is available information about the incoming workload, this is used for resource usage predictions in the admission control.

Figure 3 illustrates risk calculation based on these parameters. The red line is the expected available CPU, memory, or I/O ($Free$), the blue line shows $UnReq$, and the green line is Req. For the sake of more intuitive representation, we suppose that the system has information about the incoming service (green line). The risk at each point in time is then estimated as follows:

$$risk_i = \begin{cases} 0 & \text{if } Req_i < Unreq_i \\ \frac{Req_i - Unreq_i}{Free_i - Unreq_i} & \text{if } Unreq_i < Req_i < Free_i \\ 1 & \text{if } Req_i > Free_i. \end{cases} \quad (1)$$

Once risk is assessed for each point in time, we calculate an aggregate risk of accepting the service over time. The average of all risk points in time are used as an indicator of the likelihood of having problems, and the risk associated with the peaks are used to estimate the impact of the possible problems. A peak is defined as the area of the green line over the red line (Figure 4). The risk value for the peaks is estimated in two steps:

1) Estimate the risk for each peak as the fraction of the area over the $Free$ line (the top gray zone in Figure 4) over the total area over the $UnReq$ line (the gray area and the striped gray area).
2) Aggregate the risk of all peaks using the *product* fuzzy logic disjunction operator (see [50]):
$$p(x, y) = x + y - x * y, \quad (2)$$
where $p(x, y)$ is the aggregated risk for the peaks x and y, where x and y are obtained in step 1. This represents a realistic aggregation, but in [50] we also presented optimistic and pessimistic aggregators that follows the Gödel and Łukasiewicz fuzzy logic operators, respectively:
$$opt(x, y) = max\{x, y\} \quad (3)$$
$$pess(x, y) = min\{x + y, 1\}. \quad (4)$$

Finally these values are mixed to obtain the final risk value containing information about both likelihood and impact of degradation problems. More information about this risk calculation process including fuzzy functions can be found in [50], [4], where an evaluation of the risk assessment module behavior is presented.

6.2 Risk Threshold Controller

Calculating the risk of admitting a new service includes many uncertainties. Furthermore, choosing an acceptable risk threshold has an impact on data center utilization and performance. High thresholds result in higher utilization but the expense of exposing the system to performance degradation, whilst using lower values leads to lower but safer resource utilization. In [4] we demonstrated the advantages of using a more optimistic approach (taking more risks) when overbooking CPU and I/O capacity, and a more realistic approach for the memory. The rationale for this is that problems resulting from CPU or I/O congestion are less critical than the ones coming from running out of memory. Therefore, the different risk degrees presented can be combined according to the situation, considered capacity dimensions, knowledge about the incoming service, etc.

With the solution presented at [4], the risk assessment module gets no feedback about the current status and behavior of the system, the current workload mixture, the data center size, etc. Our previous work [4] shows that theses factors significantly affect AC performance. The AC therefore needs a mechanism to autonomously readjust its behavior to the current needs, being aware of the system misbehaving, and therefore reacting quicker to performance degradation, and at the same time favoring a more balance resource utilization among the capacity dimensions.

In order to address this issue, we propose here a control theory approach that dynamically (re)adjusts risk thresholds depending on the system behavior and the desired utilization levels, allowing the admission control to learn over time depending on current system behavior. We evaluate a proportional-integral-derivative (PID) controller [5] that readjusts risk thresholds, i.e., change the level of risk that the system is willing to face, to

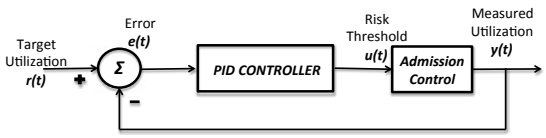


Fig. 5: PID Controller conceptual view.

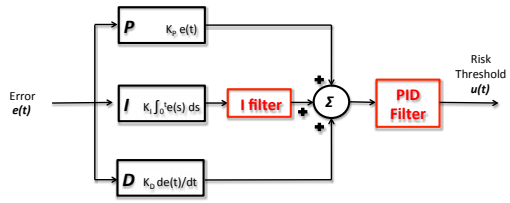


Fig. 6: PID Controller with filters.

achieve a desired level of resource utilization for each dimension (CPU, memory, and I/O) in a safer way. The weighted sum of those values is used to adjust the risk thresholds.

A conceptual view of the PID controller is depicted in Figure 5. In this figure, r denotes the desired utilization level, y the measured level, e the difference between the current utilization level and the target one, and finally u is the increment or decrement in the actual risk threshold needed to achieve the target utilization. The u value at time t can be obtained from Equation 5:

$$u(t) = K_P e(t) + K_I \int_0^t e(s)ds + K_D \frac{de(t)}{dt}. \quad (5)$$

In this equation, increasing the P component (K_P) leads to faster response but also to overshooting and oscillation problems. Increasing I component (K_I) reduces stationary errors but at the expense of larger oscillations. Finally, the D (K_D) component reduces the oscillations but may lead to slower response. Only setting these values turned out to be insufficient to achieve a good performance (stable and fast response). We hence included some filters to make the overall behavior more stable. To this end, the output of the I component and the final value (P + I + D) have been filtered (see Figure 6).

The *I filter* aims to reduce the impact of I when the desired utilization level has not been achieved, for instance, during system initialization or when there are too few service requests to saturate the system. This filter mainly avoids overshooting caused by long periods of low utilization that are not the result of admission control decisions. As this was not enough to keep the level of resource utilization stable enough, the filter for the final output of the controller was included (*PID filter*). This filter keeps threshold levels between certain values, i.e., between 0.2 and 0.8 instead of the [0,1] range. Limiting the spectrum of feasible values for the risk threshold reduces fluctuations caused by fast switching between accepting too many and too few services.

6.2.1 Distributed PID Controller

The above discussed PID Controller works properly if the performance is measured at the data center level, obtaining a smooth utilization fluctuations (close enough to the target one) for each congested capacity dimension. However, the utilization of each server may vary from the accumulated utilization – even after applying load balancing techniques. This effect cannot be totally avoided as load imbalance is also caused by the current workload characteristics. To reduce load imbalance we propose a distributed controller approach where each physical server has its own PID controllers, one for each capacity dimension.

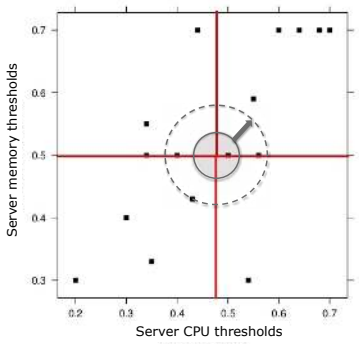


Fig. 7: Admission control risk thresholds selection.

Based on theses insights, we modify our design of the AC to have global threshold values for the whole data center based on the distributed PID controllers values. Different approaches may be used to obtain the global thresholds, such as: minimum, maximum, average or median value of all servers thresholds, but none of them correlate the different dimensions. We follow a similar approach to the one presented in the ExPERT framework [56] to select a representative threshold that correlates the three dimensions. ExPERT calculates a Pareto-frontier of scheduling strategies and selects the one that best fulfills a user-provided utility function. We apply the same concept to select the most reliable thresholds – in our case the ones closest to the median values. A conceptual view is presented in Figure 7.

The idea is to find the values closest to the median thresholds but correlating the three dimensions, taking all of them into account at once. More formally, we select as global thresholds the ones belonging to the server whose accumulated distance to the median value for each capacity dimension is minimal:

$$i \in N / \forall i' \in N, \sum_{X=\{CPU,mem,IO\}} |\widetilde{threshold}_X - threshold_{X_i}| < \sum_{X=\{CPU,mem,IO\}} |\widetilde{threshold}_X - threshold_{X_{i'}}| \quad (6)$$

In the simplified 2-dimension case plotted in Figure 7, each point represents the CPU and memory thresholds for a server. The selected point would be the one with approximated value $(0.5, 0.5)$ (the one inside the gray circle) since is the one closest to both median values (where red lines meet). If that point did not exist, then the gray circle is enlarged until one point is inside and thus selected. In fact, one of the main differences with ExPERT [56] is that we take into account a 3-dimensional

model, whereas ExPERT uses only two for cost-time optimization, besides the fact that we are aiming for the median value instead of the minimum or maximum one. This method of choosing the representative risk thresholds for the data center balances utilization in all capacity dimensions. If capacity is imbalanced, e.g., CPU utilization is greater than memory, the admission control can act on this fact and admit applications that request more capacity of the type that is further from the target utilization level (memory in the previous example).

7 Experiments

We have previously evaluated the impact of various aspects of overbooking [4] by means of a simulated infrastructure. The conclusion was that differences in data center sizes, workload mixes (ratio of bursty and steady applications), risk thresholds, etc., all have an impact on the achieved utilization and on the number of problematic situations, also leading to imbalanced server utilization. Therefore, unless these aspects are properly considered, resource utilization could be lower and less stable than desired and the performance degradation may increase. Here, we significantly extend our evaluation as follows:

- The new distributed PID controller approach is evaluated (both by simulation and in the real system).
- The performance of a the real system in evaluated in-depth, where application performance degradation due to the overbooking actions is quantified.

7.1 Testbed

The performance provided by our overbooking framework is firstly evaluated by simulating the physical resources belonging to a data center and emulating the execution of several applications and services. We generate a mix of services and applications that to the best of our knowledge are representative within cloud environments and consequently relevant for real cloud providers (see Section 7.2) and then emulate their behavior by carrying out discrete event simulations of the resources that execute them. The basic simulated cloud infrastructure used in our experiments consists of 16 nodes where each one of them has 32 cores. These cores simulate the execution of the workloads presented in Section 7.2 by following their usage profile.

Regarding the experiments using a real environment, we perform these on a 32 AMD Opteron(TM) Cores (2.1 GHz), with 56 GB of memory, where we deploy and monitor the VMs running the requested services. Following the Amazon model, we use 4 different types of VMs, whose characteristics are detailed on Table 1.

TABLE 1: Virtual Machines sizes.

	# CPUs	Memory (GB)	Bandwidth (Mbit/s)
S	1	1.7	1000
M	2	3.4	2000
L	4	6.8	4000
XL	8	13.6	8000

7.2 Workload

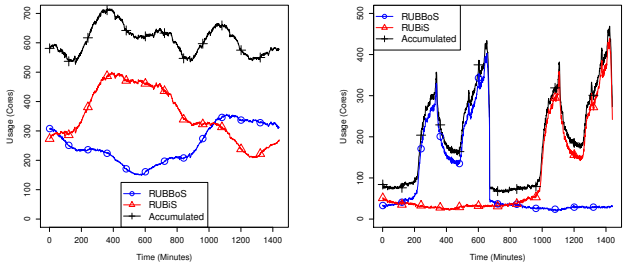
We have used and modeled different types of applications and services to try to recreate a realistic cloud workload. The modeling process is based on running the real application, monitoring its behavior and finally generation a load profile based on the monitored information. In order to generate a representative cloud workload we mix different types of applications. These are grouped in the following two classes:

- *Interactive workload*: commonly web server applications that are provisioned for a very long time and with no deadline specified. The number of user requests varies over time, which basically defines the utilization patterns (how bursty or steady the application is). The pattern of this kind of workload can be extracted from real available traces, such as the Wikipedia ones [57].
- *Batch workload*: incoming applications with different behavior and needs, with durations ranging from minutes to months. These applications may present bursty or steady performance, also in each capacity dimension (CPU, memory, or I/O). They can be generated by scripts or by using existing benchmarks. These applications are mixed and submitted by following a Poisson distribution.

This mixture of different workload groups has been chosen to have different kind of applications, with steady and bursty behaviors and/or unknown performance, etc. This way we can study horizontal and vertical elasticity problems regarding resources utilization.

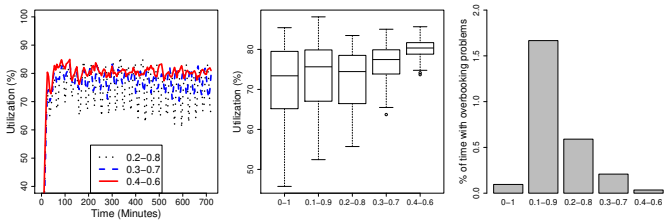
For the interactive workload we have installed and used two different types of web services. The first one is *RUBiS* [58], an auction site prototype modeled after eBay.com whilst the second one is *RUBBoS* [59], a bulleting board benchmark modeled after an online news forum like Slashdot. For the workload, we generate the number of requests that they need to serve along time, by using information extracted from publicly available web traces from the Wikipedia [57] and FIFA [60] web sites. Figure 8 shows the selected one day pattern for RUBiS and RUBBoS with the workload time-shifted 12 hours, thus each experiment runs for 24 hours.

For the dynamic workloads, we have modeled two kind of behaviors applicable to each VM dimension – bursty and steady. An example of the bursty behavior may be an interactive service whereas an example of a stable one could be a CPU-bound map-reduce job. To this end, we have made use of the 3node test from GRASP benchmarks [61] (for the steady CPU behavior) and several shell scripts to generate burstiness in the different capacity dimensions. It must be noted that some applications may present both behaviors at the same time, e.g., steady CPU behavior whilst being bursty regarding memory and/or I/O.



(a) Wikipedia traces. (b) FIFA traces.

Fig. 8: Interactive workload: Web servers showing resource requirements (requests along time), as well as the accumulated load.



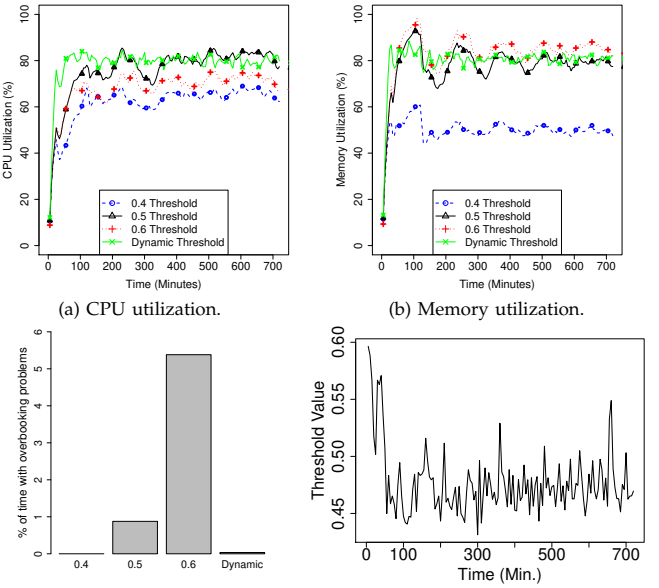
(a) Average usage over time. (b) Usage distribution. (c) % of time with an overloaded node.

Fig. 9: Suitable risk threshold for PID output filter.

7.3 Distributed PID Controller Evaluation

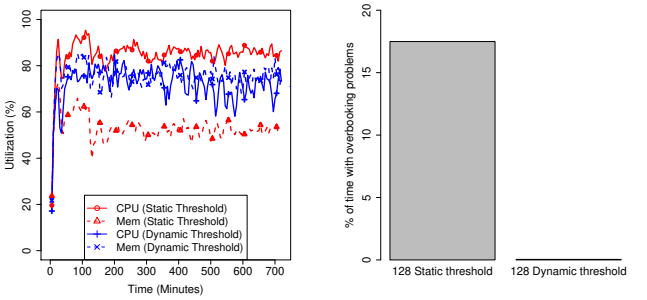
Previous experiments presented in [4] showed that the relationship between risk values and probability of overload situations is affected by external factors such as data center size or workload burstiness. This highlights the need for an autonomic system that readjusts itself depending on the current system behavior – influenced by external factors, such as the amount of users that the different services has to serve, commonly not known in advance. This fact motivates the use of feedback to adjust the level of risk that the overbooking system is willing to face over time. Once we decided to use the PID controller presented in Section 6.2, we firstly configured its parameters (K_p, K_i, K_d, and I filter) by running some tuning experiments. Then, we configured the PID output filter, since it has a great impact into the system stability, as presented in Figure 9. As depicted in Figure 9 (a) and Figure 9 (b), the larger the amplitude for the risk thresholds is, the larger the resource utilization oscillations are. Moreover, allowing bigger risk thresholds (e.g., 0.8 or 0.9) more frequently leads to resource congestion. This is illustrated in Figure 9 (c), that shows the percentage of time when one of the servers did not have enough capacity to properly process its allocated load.

Once the PID controller tunning is done, we proceed to compare the performance of the distributed PID controller approach with the method presented in [4] where the risk thresholds were statically defined and fixed over time. These comparisons are presented in Figures 10 and 11. Figures 10 (a) and (b) show that resource utilization is more stable and balanced among the capacity dimensions when risk thresholds are dynamically adjusted depending on the current load.



(a) CPU utilization. (b) Memory utilization. (c) Frequency of problems. (d) Risk threshold evolution (Avg.: 0.48).

Fig. 10: Static vs. dynamic risk thresholds.



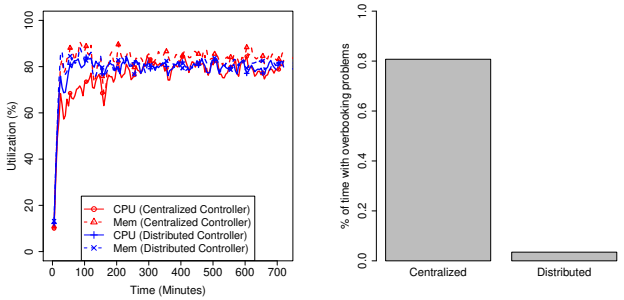
(a) Utilization: 128 cores. (b) Frequency of problems: 128 cores.

Fig. 11: Data center size: Static vs. dynamic risk thresholds.

Furthermore, the number of problematic situations is remarkable decreased (almost avoided, see Figure 10 (c)) thanks to reconfiguration of the acceptable risk level. As Figure 10 (d) depicts, only small modifications of the risk threshold are needed to achieve this performance. Figure 10 (c) also shows a non-linear increment of overload situations when risk thresholds increase and illustrates that how no problematic situations occur at all when risk is less than 0.4.

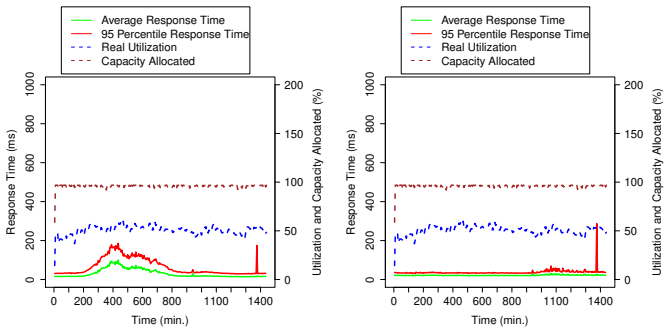
We also evaluate the distributed controller approach when the data center size is reduced to 128 cores. This experiment previously showed bad performance (imbalance utilization among different dimensions) when having static thresholds [4]. In Figure 11 it can be seen that utilization is increased and more balanced over the different dimensions thanks to risk threshold adaptation. At the same time the number of problematic situations is significantly reduced, from more than 15% of the time to less that 0.05%.

Finally, we also study the differences between the distributed PID controller approach and using a centralized (single) PID controller for the whole data center. The results for that comparison are presented in Fig-



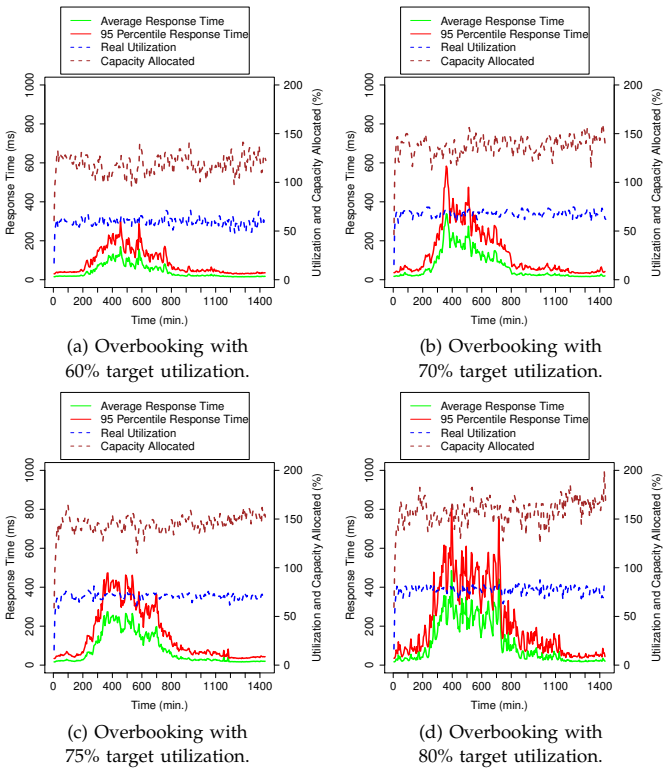
(a) Resource tilization. (b) Frequency of problems.

Fig. 12: Centralized vs. distributed PID controller.



(a) RUBiS. (d) RUBBoS.

Fig. 13: RUBiS and RUBBoS performance along time following Wikipedia workload without overbooking.



(a) Overbooking with 60% target utilization.

(b) Overbooking with 70% target utilization.

(c) Overbooking with 75% target utilization.

(d) Overbooking with 80% target utilization.

Fig. 14: RUBiS performance with different target utilization following Wikipedia workload.

ure 12. There, it can be seen that utilization is more stable and converges quicker when using distributed PID controllers (Figure 12 (a)). Moreover, the amount of problems is largely reduced (Figure 12 (b)). Thus, the distributed PID controller achieves better utilization and also reduces the risk of impacting the performance.

7.4 Overbooking impact on applications

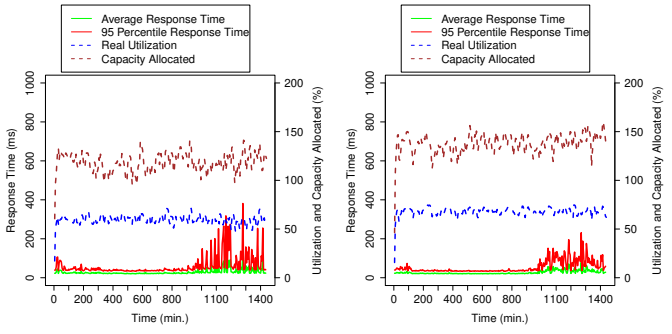
After we have studied by means of simulations the advantages of our distributed PID controller approach, we evaluate the impact of overbooking on applications performance, as although the total capacity is not used, some side effects may appear due to uncertainty and overbooking actions. Figure 13 presents the response time (average and 95 percentile) for the RUBiS and RUBBoS services running in the real server. This figure also presents the server utilization over time and the total capacity allocated (the one requested by the users). These two interactive services are selected as representative since they are more prone to be disturbed by the overbooking actions. A set of deadline-constrained applications are used to create the batch workload. These applications can tolerate performance fluctuations during their executions as long as they finish on time.

Figure 13 shows the performance of the RUBiS and RUBBoS services when no overbooking actions are taken, whilst Figures 14 and 15 respectively shows the same information with different levels of overbooking. As it can be seen in these figures, for RUBiS the response time (average and 95 percentile) increases with the number of user requests served. The peak occurs around minute 400 and the high load lasts roughly to minute 800 (see Figure 8). The same trend can be seen in all the plots, but due to the overbooking impact, the response time is noticeable bigger for the plots in Figure 14.

However, as overbooking impact is related to the level of utilization that we want to achieve (overbooking pressure), unless the target utilization is high, a noticeable increment in the resource utilization is obtained without any important impact on the performance provided by those two services (blue lines in Figures 14 (a), (b) and (c)). If higher utilization is pursued, it is achieved but at the expense of high performance degradation (slower response time). This effect is most prominent when the load inside the servers is close to their saturation point, which for both RUBiS and RUBBoS is around 500 clients when using an XL VM without overbooking.

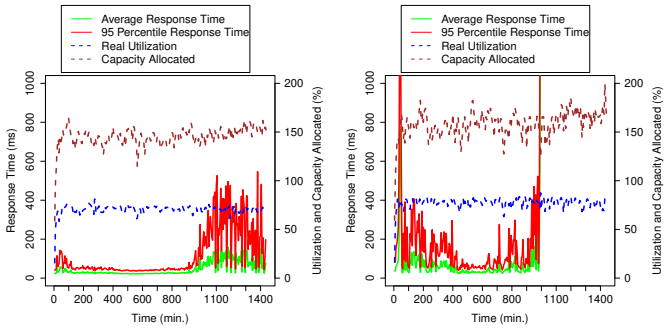
From these figures we can estimate that 75% is a suitable utilization target for RUBiS, with an acceptable impact on performance (Figure 14 (c)), increasing the utilization by a factor of 1.5 (blue line, raising from around 50% to roughly 75%). However, if the target utilization is increased a bit more (80% utilization target, Figure 14 (d)), the response time is noticeable increased even during the non overloaded periods (see period from 0-200 minutes or 800-1200). Although these response times are still acceptable, they may lead to SLA violations in case of unexpected increases in load.

Similar conclusions can be made for RUBBoS (Figure 13 (b) and Figure 15). However, this service is even more prone to suffer from overbooking as the correlation



(a) Overbooking with 60% target utilization.

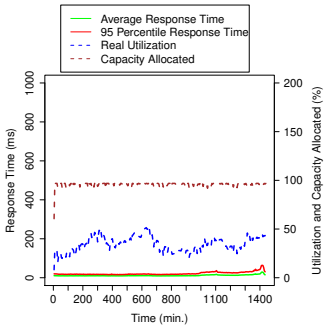
(b) Overbooking with 70% target utilization.



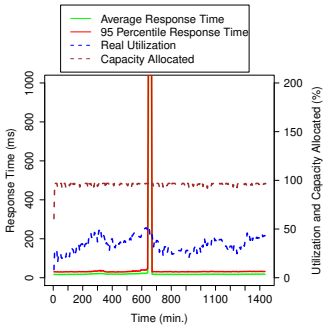
(c) Overbooking with 75% target utilization.

(d) Overbooking with 80% target utilization.

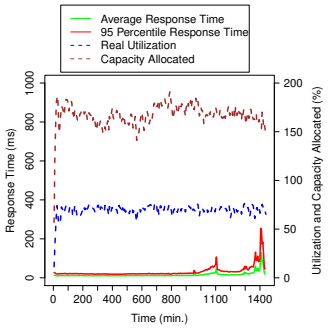
Fig. 15: RUBBoS performance with different target utilization following Wikipedia workload.



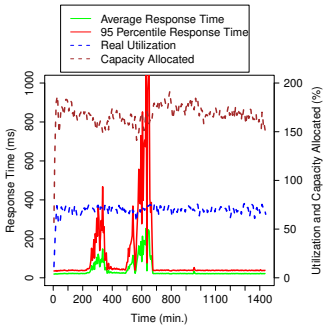
(a) RUBiS without Overbooking.



(b) RUBBoS without Overbooking.



(c) RUBiS with Overbooking:. 75% target utilization.



(d) RUBBoS with Overbooking: 75% target utilization.

Fig. 16: RUBiS and RUBBoS performance along time following FIFA workload.

between number of clients and response time is not linear. As the figures depict, the average response time is not affected at all, but the 95 percentile fluctuates much more than for the RUBiS. This trend in also depicted in the non overbooking plot, although at a smaller scale.

The RUBBoS performance is still acceptable if target utilization level is below 75%, as response times remain below a second, mostly staying below half a second. However, we also note that whilst RUBiS is still working with 80% utilization level (response time remains below 1 second), the RUBBoS service stops providing acceptable response times from minute 1000 onwards (Figure 15 (d)). For this service, 75% overbooking level may result in some SLA violations (delayed/dropped requests) and thus it is safer to only pursue 70% utilization. From these observations we conclude that overbooking can highly increase the whole data center utilization but need to be kept at a certain level in order not to impact service performance. Notably, the suitable overbooking level is application-specific, specially when pursuing utilization ratios over 70%.

Finally, Figure 16 presents a more extreme scenario, where the number of users requests per second for RUBiS and RUBBoS follows the FIFA traces (note that there is a 12 hours time shift between the RUBiS and RUBBoS peaks, depicted in Figure 8 (b)). Figure 16 shows that, although the number of users increases faster, the requests are still processed within an acceptable response time by RUBiS (Figure 16 (a) and (c)). RUBBos response times are more affected than RUBiS ones but this behavior is also represented in the non-overbooking scenario (Figure 16 (b)). In this figure, the average and 95 percentile response times are significantly increased coinciding with the peak in user requests (around minutes 650 to 700). This also highlights the fact that both services have different reactions to congestion: whilst RUBiS response time proportionally increases with the load, RUBBoS response time is kept low up to a certain point (level of congestion) where the response time is remarkable increased. A comparison summary of the overbooking impact on the RUBiS and RUBBoS services is presented at Table 2.

8 Conclusions and Future Work

This work proposes overbooking to address the utilization problems that cloud data centers face due to the elastic nature of cloud services. Overbooking has to be carefully planned in order not to impact application performance. We present an overbooking framework that performs admission control decisions based on fuzzy logic risk assessments of each incoming service deployment request. A set of distributed PID controllers are implemented to avoid performance degradation and to increase and keep the utilization evenly distributed among the servers. Feedback control is used to adapt the level of overbooking (risk threshold) that the cloud data center is able to tolerate at any time.

Our experiments show that data center utilization is not only increased in overall but also harmonized across hardware capacity dimensions and servers. From our extensive evaluation using web server benchmarks and

TABLE 2: Summary of the tests.

Workload	Service	Overbooking level	Avg. response time (ms.)	95th% response time (ms.)	Utilization (%)	Capacity Allocated (%)
WIKIPEDIA	RUBiS	No Overbooking	30.99	64.11	50.69	95.92
		60	42.51	86.18	59.66	118.24
		70	71.00	136.91	68.11	137.72
		75	79.12	151.89	72.56	147.36
		80	112.98	215.91	77.30	158.30
	RUBBoS	No Overbooking	21.90	37.61	50.69	95.92
		60	29.27	58.88	59.66	118.24
		70	27.14	53.42	68.11	137.72
		75	46.70	115.28	72.56	147.36
		80	692.36	1192.00	77.30	158.30
FIFA	RUBiS	No Overbooking	11.61	22.01	34.80	95.98
		75	15.81	31.83	70.79	167.24
	RUBBoS	No Overbooking	47.98	80.22	34.80	95.98
		75	37.42	96.19	70.79	167.24

real-life workloads, we conclude that on average, resource utilization and allocated capacity can be increased by 50% with acceptable performance degradation.

For future work we plan to compare and combine mitigation methods for unexpected misbehaviors, such as reducing the service level of some services to avoid performance degradation. We also plan to further advance into the autonomic behavior of the overbooking system by enabling the possibility of self-adapting the target utilization level depending on the applications behavior and how sensitive they are to overbooking pressure. Another future research direction is to study affinity functions that aid the scheduling system in deciding which applications to collocate, reducing the performance degradation when overbooking.

ACKNOWLEDGMENTS

This work was supported in part by the Swedish Research Council under grant number 2012-5908 for the Cloud Control project. We thank Ahmed Ali-Eldin who provided valuable feedback on workload traces and elasticity. We also acknowledge Carlos Vázquez and Ginés Moreno for their help with the fuzzy logic programming implementation of the risk assessment. We finally recognize Anders Robertsson's help regarding PID controller configuration and tuning.

REFERENCES

[1] A. Ali-Eldin, J. Tordsson, and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *Proc. of Network Operations and Management Symposium (NOMS)*, 2012, pp. 204–212.

[2] A. Sulistio, K. H. Kim, and R. Buyya, "Managing cancellations and no-shows of reservations with overbooking to increase resource revenue," in *Proc. of Intl. Symposium on Cluster Computing and the Grid (CCGrid)*, 2008, pp. 267–276.

[3] L. Tomás and J. Tordsson, "Improving Cloud Infrastructure Utilization through Overbooking," in *Proc. of ACM Cloud and Autonomic Computing Conference (CAC)*, 2013.

[4] ——, "Cloudy with a chance of load spikes: Admission control with fuzzy risk assessments," in *Proc. of 6th IEEE/ACM Intl. Conference on Utility and Cloud Computing*, 2013, pp. 155–162.

[5] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[6] T.-F. Chen and J.-L. Baer, "Effective hardware-based data prefetching for high-performance processors," *IEEE Transactions on Computers*, vol. 44, no. 5, pp. 609–623, 1995.

[7] J. Flich, S. Rodrigo, J. Duato, T. Sodring, A. Solheim, T. Skeie, and O. Lysne, "On the potential of noc virtualization for multicore chips," in *Proc. of Intl. Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, 2008, pp. 801–807.

[8] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 12, pp. 1320–1331, 1993.

[9] B. B. Chen and P.-B. Primet, "Scheduling deadline-constrained bulk data transfers to minimize network congestion," in *Proc. of IEEE 7th Intl. Symposium on Cluster Computing and the Grid (CCGRID)*, 2007, pp. 410–417.

[10] A. J. Bernstein and J. C. Sharp, "A policy-driven scheduler for a time-sharing system," *Communications of the ACM*, vol. 14, no. 2, pp. 74–78, 1971.

[11] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Proc. of IEEE/ACM 12th Intl. Conference on Grid Computing*, 2011, pp. 26–33.

[12] P.-O. Östberg, D. Espling, and E. Elmroth, "Decentralized scalable fairshare scheduling," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 130 – 143, 2013.

[13] A. Corradi, M. Fanelli, and L. Foschini, "VM consolidation: A real case based on OpenStack Cloud," *Future Generation Computer Systems*, vol. 32, no. 0, pp. 118 – 127, 2014.

[14] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, 2012.

[15] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing," in *Proc. Intl. Conference of Autonomic Computing (ICAC)*, 2010, pp. 11–20.

[16] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale:Google trace analysis," Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. ISTC–CC–TR–12–101, Apr. 2012, http://www.istc-cc.cmu.edu/publications/papers/2012/ISTC-CC-TR-12-101.pdf.

[17] L. A. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[18] D. Gmach, J. Rolia, and L. Cherkasova, "Selling t-shirts and time shares in the cloud," in *Proc. of Intl. Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2012, pp. 539–546.

[19] W. Cirne and F. Berman, "A comprehensive model of the supercomputer workload," in *Proc. of Intl. Workshop on Workload Characterization*, 2001, pp. 140–148.

[20] R. Guerin, H. Ahmadi, and M. Naghshineh, "Equivalent capacity and its application to bandwidth allocation in high-speed networks," *Selected Areas in Communications*, vol. 9, no. 7, pp. 968–981, 1991.

[21] G. Birkenheuer, A. Brinkmann, and H. Karl, "The gain of overbooking," in *Proc. of Job Scheduling Strategies for Parallel Processing*, ser. LNCS, 2009, vol. 5798, pp. 80–100.

[22] R. Ghosh and V. K. Naik, "Biting off safely more than you can chew: Predictive analytics for resource over-commit in iaas cloud," in *Proc. of 5th Intl. Conference on Cloud Computing*, 2012, pp. 25–32.

[23] B. Urgaonkar, P. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in shared hosting platforms," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 239–254, 2002.

[24] J. Broeckhove and K. Vanmechelen, "Network aware scheduling for virtual machine workloads with interference models," *IEEE Transactions on Services Computing*, vol. 99, no. PrePrints, 2014.

[25] R. Householder, S. Arnold, and R. Green, "On cloud-based oversubscription," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 8, no. 8, pp. 425–431, 2014, http://arxiv.org/abs/1402.4758v2.

[26] J. Subramanian, S. Stidham, and C. J. Lautenbacher, "Airline yield management with overbooking, cancellations, and no-shows," *Transportation Science*, vol. 33, no. 2, pp. 147–167, 1999.
[27] K. Konstanteli, T. Cucinotta, K. Psychas, and T. Varvarigou, "Admission control for elastic cloud services," in *Proc of IEEE 5th Intl. Conference on Cloud Computing (CLOUD)*, 2012, pp. 41–48.
[28] N. Leontiou, D. Dechouniotis, and S. Denazis, "Adaptive admission control of distributed cloud services," in *Proc. of Intl. Conference on Network and Service Management (CNSM)*, 2010, pp. 318–321.
[29] M. A. Kjaer, M. Kihl, and A. Robertsson, "Resource allocation and disturbance rejection in web servers using SLAs and virtualized servers," *IEEE Transactions on Network and Service Management*, vol. 6, no. 4, pp. 226–239, 2009.
[30] A. Ashraf, B. Byholm, and I. Porres, "A session-based adaptive admission control approach for virtualized application servers," in *Proc. of Intl. Conference on Utility and Cloud Computing (UCC)*, 2012, pp. 65–72.
[31] S. He, L. Guo, M. Ghanem, and Y. Guo, "Improving resource utilisation in the cloud environment using multivariate probabilistic models," in *Proc of 5th Intl. Conference on Cloud Computing (CLOUD)*, 2012, pp. 574–581.
[32] C. Mastroianni, M. Meo, and G. Papuzzo, "Probabilistic consolidation of virtual machines in self-organizing cloud data centers," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 215–228, 2013.
[33] T. Wo, Q. Sun, B. Li, and C. Hu, "Overbooking-based resource allocation in virtualized data center," in *Proc of 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW)*, 2012, pp. 142–149.
[34] L. Larsson, D. Henriksson, and E. Elmroth, "Scheduling and monitoring of internally structured services in cloud federations," in *Proc. of IEEE Intl. Symposium on Computers and Communications (ISCC)*, 2011, pp. 173–178.
[35] D. Breitgand, Z. Dubitzky, A. Epstein, O. Feder, A. Glikson, I. Shapira, and G. Toffetti, "Pulsar: An adaptive utilization accelerator for iaas clouds," in *IEEE International Conference on Cloud Engineering (IC2E)*, 2014.
[36] M. Dobber, R. van der Mei, and G. Koole, "A prediction method for job runtimes on shared processors: Survey, statistical analysis and new avenues," *Performance Evaluation*, vol. 64, no. 7-8, pp. 755–781, 2007.
[37] H. Jin, X. Shi, W. Qiang, and D. Zou, "An adaptive meta-scheduler for data intensive applications," *Intl. Journal of Grid and Utility Computing*, vol. 1, no. 1, pp. 32–37, 2005.
[38] M. Kalantari and M. K. Akbari, "Grid performance prediction using state-space model," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 9, pp. 1109–1130, 2009.
[39] L. Tomás, A. Caminero, C. Carrión, and B. Caminero, "Exponential Smoothing for Network-aware Meta-scheduler in Advance in Grids," in *Proc of 6th Intl. Workshop on Scheduling and Resource Management on Parallel and Distributed Systems (SRMPDS)*, 2010, pp. 323–330.
[40] L. Tomás, A. C. Caminero, C. Carrión, and B. Caminero, "Network-aware meta-scheduling in advance with autonomous self-tuning system," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 486 – 497, 2011.
[41] K. Djemame, J. Padgett, I. Gourlay, and D. Armstrong, "Brokering of risk-aware service level agreements in grids," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 13, pp. 1558–1582, 2011.
[42] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, 2009, pp. 28–28.
[43] D. Breitgand, Z. Dubitzky, A. Epstein, A. Glikson, and I. Shapira, "SLA-aware resource over-commit in an IaaS cloud," in *Proc. of 8th Intl. Conference on Network and Service Management (CNSM)*, 2012, pp. 73–81.
[44] L. Wu, S. K. Garg, and R. Buyya, "SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1280 – 1299, 2012.
[45] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1366–1379, 2013.
[46] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *10th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2007, pp. 119–128.
[47] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, no. 17, pp. 2923–2938, 2009.
[48] C. Klein, M. Maggio, K.-E. Årzén, and F. Hernández-Rodriguez, "Introducing service-level awareness in the cloud," Lund University, Tech. Rep. ISRN LUTFD2/TFRT-7641-SE, Jul. 2013.
[49] P. Vojtás, "Fuzzy logic programming," *Fuzzy Sets and Systems*, vol. 124, no. 3, pp. 361–370, 2001.
[50] C. Vázquez, L. Tomás, G. Moreno, and J. Tordsson, "A fuzzy approach to cloud admision control for safe overbooking," in *Proc. of Intl. Workshop on Fuzzy Logic and Applications (WILF). Springer Verlag, LNCS*, 2013, pp. 212–225.
[51] libvirt: The virtualization API, Web page at http://libvirt.org/, Visited 2014-01-27.
[52] Nagios - The Industry Standard in IT infrastructure Monitoring, Web page at http://www.nagios.org/, Visited 2014-01-27.
[53] LTTng Project. Linux Trace Toolkit - next generation, Web page at http://lttng.org/lttng2.0, Visited 2014-01-27.
[54] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the linux virtual machine monitor," in *Linux Symposium*, 2007.
[55] P. S. Kalekar, "Time series Forecasting using Holt-Winters Exponential Smoothing," Kanwal Rekhi School of Information Technology, Tech. Rep., 2004.
[56] O. Ben-Yehuda, A. Schuster, A. Sharov, M. Silberstein, and A. Iosup, "Expert: Pareto-efficient task replication on grids and a cloud," in *Proc. of IEEE 26th Intl. Parallel Distributed Processing Symposium (IPDPS)*, 2012, pp. 167–178.
[57] Page view statistics for Wikimedia projects, Web page at http://dumps.wikimedia.org/other/pagecounts-raw/, Visited 2014-01-27.
[58] RUBiS: Rice University Bidding System, Web page at http://rubis.ow2.org/, Visited 2014-01-27.
[59] RUBBoS: Bulletin Board Benchmark, Web page at http://jmob.ow2.org/rubbos.html, Visited 2014-01-27.
[60] 1998 World Cup Web Site Access Logs - The Internet Traffic Archive, Web page at http://ita.ee.lbl.gov/html/contrib/WorldCup.html, Visited 2014-01-27.
[61] G. Chun, H. Dail, H. Casanova, and A. Snavely, "Benchmark probes for Grid assessment," in *Proc. of 18th Intl. Parallel and Distributed Processing Symposium*, 2004, pp. 26–30.

Luis Tomás is a postdoctoral researcher at Umeå University. He received his PhD Degree in Computer Science from University of Castilla–La Mancha in 2012. His current research interests are QoS support and autonomic resource management within highly distributed computing infrastructures such as grid and cloud environments.

Johan Tordsson is Assistant Professor at Umeå University, from which he also received his PhD in 2009. After a period as visiting postdoc researcher at Universidad Complutense de Madrid, he worked for several years in the RESERVOIR, VISION Cloud, and OPTIMIS European projects, in the latter as Lead Architect and Scientific Coordinator. Tordsson's research interests include autonomic resource management for cloud and grid computing infrastructures as well as virtualization.