



<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *6th IEEE International Conference on Utility and Cloud Computing*.

Citation for the original published paper:

Tomas, L., Tordsson, J. (2013)

Cloudy with a Chance of Load Spikes: Admission Control with Fuzzy Risk Assessments.

In: *6th IEEE International Conference on Utility and Cloud Computing* (pp. 155-162).

Proceedings of the 6th IEEE International Conference on Utility and Cloud Computing

<http://dx.doi.org/10.1109/UCC.2013.38>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-80399>

Cloudy with a Chance of Load Spikes: Admission Control with Fuzzy Risk Assessments

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

Abstract—Elasticity is key for the cloud paradigm, where the pay-per use nature provides great flexibility for end-users. However, elasticity complicates long-term capacity planning for cloud providers as the exact amount of resources required over time becomes uncertain. Admission control techniques are thus needed to handle the trade-off between resource utilization and potential overload. We define a set of admission control algorithms that combine risk assessment methods with a fuzzy aggregation framework. An experimental evaluation using a mixture of bursty and steady applications demonstrate that our algorithms can increase resource utilization by a factor of two while limiting overload problems to a few percent.

I. INTRODUCTION

The elastic nature of clouds [1] allows users to dynamically adjust resource allocations. For cloud infrastructure providers, capacity planning becomes a challenging problem as the exact number of Virtual Machines (VMs) needed at any time by the users is hard to estimate. Overestimating the required capacity may result in poor resource utilization and lower income from consumers, whereas underestimates can lead to performance degradation and/or crashes.

In addition to elasticity, there are multiple factors that contribute to low resource utilization in clouds. First, cloud providers commonly only offer predefined VM sizes (e.g., S, M, L, and XL) with a fixed amount of CPU, memory, disk, etc., which forces users to select the VM size that provides enough of the most critical resource type (such as CPU), while typically over-provisioning the others types [2]. Second, most cloud applications do not use the same amount of hardware resources all the time as they have different behavior during their different phases of execution (known as vertical elasticity). In such scenarios, users have to provision the worst-case capacity as the upper bound. Third, users tend to overestimate their resource needs for the sake of safe execution, as illustrated by a supercomputing workload study that concluded that half of all jobs used less than 20% of the requested capacity [3].

Cloud providers can mitigate these resource utilization problems through overbooking, i.e., resource management in any manner where the total available capacity is less than the theoretical maximal requested capacity. In our previous work [4], we demonstrate how resource overbooking can be used to increase provider utilization and revenue. Basic admission control strategies were presented to cope with horizontal elasticity issues while scheduling techniques were

proposed to handle the effects of vertical elasticity and predefined VM sizes. However, overbooking techniques always exposes the cloud provider to a risk of resource congestion and consequently of SLA violations as the future behavior of the workloads is not known. This is one of the main challenging issues when dealing with overbooking: how to decide the appropriate level of overbooking that can be achieved without impacting the performance under unexpected situations [5].

To handle this trade-off, admission control mechanisms can be used by cloud providers to determine whether a new user service request should be admitted into the data center or not. Measuring or evaluating the long term risk being taken by the admission control at performing resource overbooking is not trivial. There are multiple criteria that must be considered for this problem, e.g., the hardware available in the data center, currently admitted services requests, the variation in their resource usage over time, etc., as well as the uncertainty of future behavior of services. The impact of potential overload situations is also hard to assess. For example, most applications run, albeit slower, if allocated too little CPU, whereas provisioning too little memory commonly makes applications crash. Additional application characteristics include, e.g., that CPU capability cannot be treated in the same way for high throughput applications as for real time applications.

Taking all the above aspects into account in a single admission control algorithm is difficult. Even selection of which information to prioritize is non-trivial. To handle this uncertainty, we propose an admission control framework with a general risk assessment approach based on fuzzy logic. Our framework and fuzzy logic programming engine enables multiple admission control algorithms to be defined to take more aggressive (optimistic) or conservative (pessimistic) decisions, depending on, e.g., provider capability, knowledge about the new incoming workload, and/or preferred risk level.

The rest of the paper is organized as follows. Section II explains how to measure and evaluate risk and also describes our risk-based admission control algorithms. An experimental evaluation of these algorithms is presented in Section III. Related work is discussed in Section IV and our conclusions, including future research directions, are found in Section V.

II. ADMISSION CONTROL WITH RISK ASSESSMENT

Admission control must be carefully planned to achieve high utilization and avoid exposing the system to SLA violations. Critical information sources for admission control include the



Fig. 1: Illustration of uncertainty in admission control.

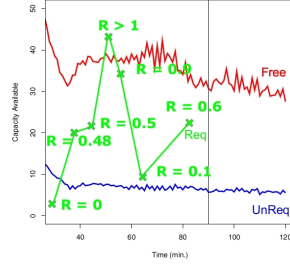


Fig. 2: How to evaluate the risks at one point.

hardware available in the data center, the set of currently admitted services requests and their variation in resource usage over time, all information about the new service request, etc. Taking all that information into account at once is not trivial. Selecting which information source has the greatest impact is difficult too, and changes depending on the scenario. Furthermore, information is sometimes missing. For instance, the historical burstiness of existing service requests can be recorded, but their future behavior is still unknown. A key aspect of all overbooking systems is therefore insight in future resource usage, but at the expense of taking risks due to the predictions inaccuracy or misbehavior. However, it is not clear how to estimate the risk for a given situation. An example of these difficulties is presented in Figure 1, where the red line illustrates the amount of expected available CPU, and $Req1$ and $Req2$ are two different CPU allocation requests. In this scenario, it is non-trivial to determine whether $Req1$ can be admitted, decide about admission for $Req2$, and also to determine which of these requests is the most risky one. These decisions are significantly affected by prediction accuracy both in terms of resource usage and workload characteristics (size and duration of peaks, etc.).

Consequently, we define a generic metric for the risk of accepting a new service that form the basis for different types of admission control algorithms and also can be used for other resource management tasks such as deciding the price for the allocation or for the penalties in case of SLA violation. The inputs for this risk assessment module are:

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

The values for $Free$ and $UnReq$ are predicted using exponential smoothing [4], [6]. Req is either obtained from previous knowledge of the service or specified by the cloud user. Figure 2 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 . The risk at each point in time is estimated as follows:

- $Risk = 0$ if $Req < Unreq$. Capacity is available and no overbooking is needed to accommodate the request.

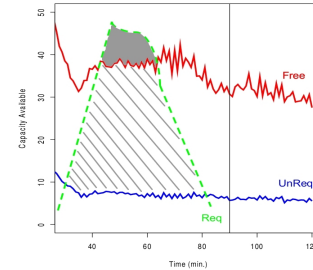


Fig. 3: Area calculation used for peak risk assessment.

- $Risk > 1$ if $Req > Free$. We expect to have insufficient capacity for the allocation, even with overbooking.
- $Risk \in (0, 1)$ if $Unreq < Req < Free$. Overbooking is needed for the allocation but sufficient available capacity is expected. The exact risk value is calculated as the euclidean distance between the three points.

As the green line illustrates, one request can have no risk at some points in time, low risk at others, and high risk later. We now look for an aggregate risk metric that captures changes in risk over time. We also want to take into account that the VM capacity dimensions behave differently, e.g., overbooking CPU resources can be done aggressively whereas memory overbooking must be conservative [7].

To calculate those risk values we estimate, on the one hand, the risk value associated to the average of all risk point values and, on the other hand, the risk value regarding the peaks, which are the more dangerous/difficult situations. Regarding the risk value for the average, the obtained average of all the risks, whenever below 1, is modified by different *linguistic modifiers* (explained at Section II-A) to obtain those different risk degrees (optimistic, realistic, and pessimistic). On the other hand, the peaks are the areas made by the green line when it is over the red one. This is shown in Figure 3, where the top gray zone represents a peak over the $Free$ line, and whose associated value is the percentage that this area represents over the total area (the gray area and the striped gray area) over the $UnReq$ line. There may be more than one peak, so their information (risk values) have to be somehow summarized together. To do that, different *disjunction operators* are used, as explained below, to also obtain different degrees when calculating the risk associated to the peaks. Finally, those two risk values (for the average and the peaks) are aggregated to estimate the final risk associated to the acceptance of the Req service. The same *disjunction operators* are used, joining together the optimistic value for the average with the optimistic value for the peaks – and the same for the realistic and pessimistic approaches. All those different operators and modifiers, as well as the main fuzzy logic programming implementation are detailed next.

A. Fuzzy Risk Aggregation

We use fuzzy logic formulations to combine the different risk values in flexible manners. In order to provide the needed mechanisms for calculating the average and peaks risks (and

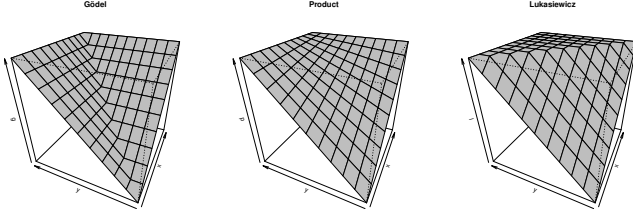


Fig. 4: Fuzzy logic disjunction from *Gödel* (optimistic), *Product* (realistic), and *Łukasiewicz* (pessimistic) logics.

to mix them), extended versions of the fuzzy logic disjunctions (optimistic, realistic, and pessimistic) are implemented, as illustrated in Figure 4.

The inputs to the fuzzy program are the curves associated with *Free*, *UnReq*, and *Req* values, as well as a fourth argument indicating which resource (Field) is considered (CPU, memory, I/O). Then, point by point, risk is assigned a value greater than 1 when the requested resource is over the free value, 0 when is below the unrequested value, and a number in the interval [0,1] when the requested point is between the other two values. This last value is obtained through linear interpolation.

Once all the information regarding the risk values for each point is calculated, the final values for the average and peaks risks are aggregated. For the average value, two fuzzy logic connectives (linguistic modifiers) are used, *@very* and *@approx*, in our case defined as x^2 and \sqrt{x} , respectively. The three level of risk assessments (pessimistic, realistic, and optimistic) regarding the average are obtained by appropriately modulating the final average (realistic approach) with the pessimistic and the optimistic modifiers (*approx* and *very*). To evaluate the final performance impact of all the peaks, they are combined all together by using the different versions of the disjunction operators presented in Figure 4. This is modeled according to Łukasiewicz, Product, and Gödel logics [8], which produce the values for the pessimistic, realistic, and optimistic scenarios, respectively.

Finally, both risk values (average and peaks) are combined in pairs by using the same disjunction operators. Thus, a pessimistic average value is combined with pessimistic peak value by using the pessimistic disjunction operator (Łukasiewicz), and similarly for the realistic and optimistic approaches. A summary of all these steps performed to obtain the final pessimistic, realistic and optimistic risk values is detailed in Algorithm 1. Our fuzzy framework is based on *Multi-Adjoint Logic Programming* (MALP) [9], [10] and implemented done using the FLOPER (*Fuzzy LOGic Programming Environment for Research*) system [11].

B. Risk-Aware Admission Control

Once we have those different ways (optimistic, realistic, and pessimistic) of evaluating the possible impact that accepting a new request may have into the system behavior, the admission control is in charge of using that information in order to take the final decision. Nevertheless, how to use that information

Algorithm 1 Risk assessment calculation steps for each VM dimension

- 1: Input: Let *Req*, *Free* and *UnReq* = list of predicted values for requested, available and unrequested capacity, respectively
 - 2: Input: Let *RiskDegree* the required truth degree of risk (pessimistic, realistic or optimistic)
 - 3: Let *RiskValue* = list of risk values for each point
 - 4: Let *PeakRisk* = list of risk values for each peak
 - 5: *modifiers(values, RiskDegree)* = be the logic linguistic modifiers to obtain the pessimistic, realistic and optimistic values
 - 6: *disjunction(values, RiskDegree)* = be the logic disjunction functions to obtain the pessimistic, realistic and optimistic values when joining two or more variables
 - 7: **for all** *i* in *Req* **do**
 - 8: *RiskValue_i* = *GetRiskValue*(*Req_i*, *Free_i*, *UnReq_i*)
 - 9: **end for**
 - 10: **for all** peaks in *Req*, *Free* **do**
 - 11: *PeakRisk_i* = *GetPeakValue*(*Req_i*, *Free_i*, *UnReq_i*)
 - 12: **end for**
 - 13: *AvgRisk* = *GetAvgRisk*(*RiskValue*)
 - 14: *AvgRisk_{RiskDegree}* = *modifiers*(*AvgRisk*, *RiskDegree*)
 - 15: *PeaksRisk_{RiskDegree}* = *disjunction*(*PeakRisk*, *RiskDegree*)
 - 16: *FinalRisk_{RiskDegree}* = *disjunction*(*AvgRisk_{RiskDegree}*, *PeaksRisk_{RiskDegree}*, *RiskDegree*)
 - 17: **return** *FinalRisk_{RiskDegree}*
-

also have an impact into the data center performance. As highlighted in [12], taking overbooking actions by only considering average resource requirements or only one dimension can result in significantly reduced performance. This is one of the main reasons why different logics for risk assessment were implemented [13]. We thus use a mixture of the logics depending on the capacity dimension. In this approach a more optimistic approach (taking more risks) is used when overbooking CPU and I/O dimensions, as we thought that the problems resulting from that are not that dangerous as the ones coming from not having enough memory available. Hence, a more pessimistic approach (using both pessimistic and realistic values) is used when overbooking memory, resulting in lower risk. This new technique is named *Capacity-Aware*. For this, and all other algorithms, a new request is only admitted if the final risk level (aggregated using Algorithm 1) is below a pre-defined threshold.

We also use risk assessment information together with the ideas presented in our previous work [4] to minimize the risk being taken even when using the more optimistic approaches. The main steps of the simple admission control method implemented in [4] are outlined in Algorithm 2. This algorithm uses a overbooking threshold to decide on admission. We combine this scheme with optimistic and realistic risk assessment and also with the capacity-aware algorithm. In summary, the different admission control algorithms are:

- *No Risk*: base case where no overbooking actions are taken.
- *ACSOS*: overbooking technique presented in [4] where long term risks are not evaluated (See Algorithm 2).
- *Pessimistic*, *Realistic* and *Optimistic*: conservative, regular, and aggressive risk aware admission control technique pre-

Algorithm 2 Overbooking Admission Control (ACSOS)

```
1: Let  $RC_{used}$  the predicted data center usage
2: Let  $AppProfile$  the predicted  $app$  usage
3: Let  $TC$  the total data center capacity
4: Let  $OBF$  indicated how overbookable is the data center
5: Let  $\alpha$  be the acceptable margin for data center overcapacity
6: if  $Prediction(RC_{used}) + AppProfile \leq TC + \alpha$  and
    $OBF > OBF_{threshold}$  then
7:   Accept  $App$ 
8: else
9:   Reject  $App$ 
10: end if
```

sented in [13], respectively, obtained as Algorithm 1 details by calling it with the desired truth degree.

- *CapacityAware*: risk aware admission control technique that differentiates VM dimensions. It applies Algorithm 1 with different truth degree values for the different dimensions, i.e., optimistic risk values are used for the CPU and I/O, whilst realistic values for memory.

- *ACSOS-Opt*: aggressive risk aware admission control technique plus ACSOS methods (Algorithm 2 + Algorithm 1 with optimistic value as truth degree).

- *ACSOS-Real*: risk aware admission control technique plus ACSOS methods (Algorithm 2 + Algorithm 1 with realistic value as truth degree).

- *ACSOS-Cap*: risk and capacity aware admission control technique plus ACSOS methods (Algorithm 2 + Algorithm 1 with optimistic and realistic truth degrees, i.e., ACSOS + CapacityAware).

III. EXPERIMENTS

This section presents the experiments performed to evaluate our risk-aware admission control mechanisms. First we describe our experimental framework and how workloads are modeled. Next, a serie of experiments evaluate the proposed admission control algorithms with respect to achieved resource utilization and resulting resource insufficiency.

A. Experiment framework

To evaluate the proposed admission control schemes, we reuse and extend our cloud resource overbooking framework [4], as illustrated in Figure 5. It consists of a risk-aware admission control module (AC), a Smart Overbooking Scheduler (SOS), and a Knowledge DB (KOB).

1) *Admission Control*: The Admission Control module (AC) decides whether a new service/application deployment request should be accepted or not. To take that decision, AC use the information provided by the (KOB) and evaluates the impact that accepting the new request will have both at short and long term system behavior, i.e., weighting improved utilization against the risk of overpassing the total capacity, potentially resulting in SLA violations. The AC invokes the risk assessment module to obtain the different values for the risks. These risk metrics are used to decide on admission or not according to the algorithms defined in Section II-B.

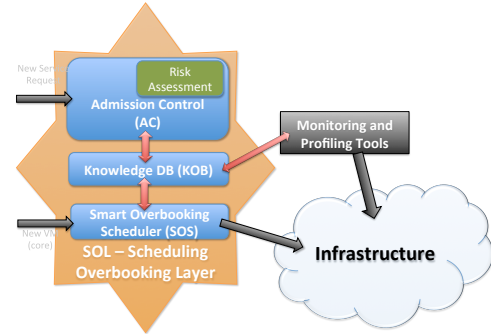


Fig. 5: Resource management framework overview.

2) *Smart Overbooking Scheduler*: When the AC decides to accept a new application, the SOS has to select the most suitable placement for it, i.e., decide in which node(s) to allocate the new VM(s). The scheduler aims to increase the node level utilization by overbooking each physical server within certain bounds, while avoiding performance degradations due to overpassing the total capacity. The scheduler achieves safe overbooking by co-allocating CPU-bound applications with network-bound or memory-bound ones (resource multiplexing) and by performing safe time-sharing, i.e., ensuring that peaks and lows in the cohosted VMs do not coincide. The scheduler uses a worst-fit style algorithm that combines the future resource usage (estimated using triple exponential smoothing [6]) with profiling of the application. The selected host for a new VM is the one with the largest residual capacity after adding the new VM (all capacity dimensions weighted). In our previous work, this scheduling algorithm demonstrated potential for significant improvements in resource utilization while avoiding overpassing the total capacity [4].

3) *Knowledge DB: Monitoring and Profiling Tools*: Decisions taken by the AC and SOS must be taken considering both current and estimated behavior of resources and workloads already deployed. To this end, the KOB module measures and profiles the different applications' behavior, as well as physical server and VM resource usage, taking into account all different dimensions (CPU, memory, and I/O). This module uses a plug-in architecture model to interface various existing monitoring tools, e.g., Nagios [14] for physical servers, the Libvirt library [15] for VMs, and LTTng2 [16] for applications.

B. Infrastructure and Workload Modeling

We emulate the behavior of workloads and carry out discrete event simulations of the resources that execute the workloads to study the demand imposed on our system: we simulate the physical resources belonging to the data center and emulate several applications with a range of workload characteristics. The cloud infrastructure simulated for testing our algorithms consists of 16 nodes where each one of them has 32 cores. Those cores simulate the execution of the workloads by following the profiled usage of them. We previously confirmed that our simulator performs similar to real servers [4]. We use four different types of VMs (S, M, L, and XL), with 1 to 8

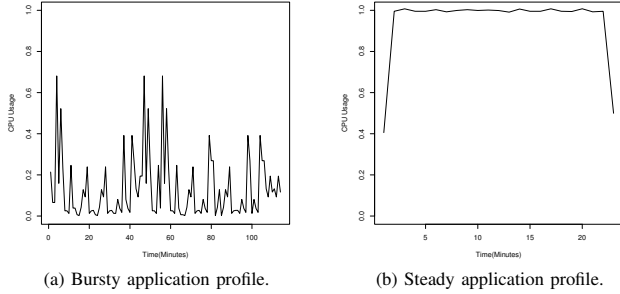


Fig. 6: Resource usage for bursty and steady applications.

cores, 1.7 to 14 GB of memory, etc.

We emulate two classes of workloads with different duration and behaviors: bursty applications (Figure 6 (a)) such as web servers and steady applications (Figure 6 (b)) such as CPU-bound map-reduce jobs. These two application profiles have been obtained by executing two different types of applications and use the LTTng2 monitoring tool to measure their resource usage over time. These two application types are equally mixed and submitted into the system according to a Poisson distribution with 20 requests per minute on average, more than enough to saturate the infrastructure.

C. Performance Evaluation

The performance evaluation of our admission control algorithms defined in Section II-B measures achieved utilization as well as the frequency and magnitude of overload situations.

The first comparison is depicted in Figure 7 where achieved CPU (Figure 7 (a)), memory (Figure 7 (b)) and accumulated (Figure 7 (c)) utilizations by each technique are shown. We observed that NoRisk and Pessimistic achieve significantly lower utilization. The ACSOS and other techniques that use a too aggressive approach (optimistic risk values) achieve higher utilization for one capacity dimension (memory) but at expense of lower utilization at the other (CPU). They thus present a more asymmetric behavior regarding utilization of the different capacity dimensions. In contrast, the Realistic and CapacityAware techniques present a more symmetric behavior regarding CPU and memory utilization but present more fluctuations along time. Finally, combining ACSOS and realistic risk assessment (ACSOS-Real and ACSOS-Cap) results in the best performance. These two methods are stable over time and also symmetric regarding utilization in different capacity dimensions. Moreover, using an optimistic value for the CPU dimension (ACSOS-Cap) allows a slight improvement in terms of stability and accumulated utilization.

However, the resource utilization increment may result in overload situations unless predictions are accurate and capacity is sufficient. Figure 8 shows the possible problems of increasing the utilization by those overbooking actions. On the one hand, Figure 8 (a) depicts the percentage of time that a node has overpassed its capacity – not the whole data center capacity, which only happens for the approaches using the optimistic risk calculations. On the other hand, Figure 8 (b)

TABLE I: Performance Summary (Figure 7).

	Average utilization	% problems	problem size
No Risk	38.56 %	0	0
Pessimistic	61.94 % (1.61)	0	0
Realistic	78.08 % (2.02)	1.07	0.26%
Optimistic	74.39 % (1.93)	8.80	0.33%
ACSOS	75.39% (1.96)	6.58	0.33%
CapacityAware	77.03 % (2.00)	0.88	0.30%
ACSOS-Opt	74.30 % (1.93)	6.40	0.32%
ACSOS-Real	77.87 % (2.02)	0.25	0.17%
ACSOS-Cap	78.11 % (2.03)	0.87	0.22%

shows the magnitude of the overload problem, at node level, with the y-axis illustrating the extra capacity required to process all workloads. In these figures, it can be seen that the two algorithms achieving the highest utilization (ACSOS-Real and ACSOS-Cap) also present among the least frequent and smallest overload problems. The Pessimistic approach does not present any problem as it is too conservative but its utilization is low compared with the others. When comparing Optimistic with ACSOS-OPT and Realistic with ACSOS-Real, we note that the combination of risk assessment and ACSOS results in fewer overload situations. Table I summarizes the utilization, overload frequency and magnitude for all algorithms. It must be noted that those utilization problems just involving single nodes can even be solved/alleviated through VM live migrations, since the whole data center is not saturated and there are other nodes that may received the migrated VM without overpassing their total capacity.

After investigating the utilization and overload results, we evaluate the impact of changing data center size, workload mixture, and risk thresholds.

1) *Data center size*: Figure 9 shows the performance impact when data center size is changed from 128, 256, 512, and 1024 cores. Again, Figure 9 (a) and (b) show CPU and memory utilization, respectively. Figure 9 (c) and (d) show the percentage of time that a node has overpassed its capacity and the magnitude of the problems when that happens, respectively. In the first two figures it is shown that the bigger the data center is, the less utilization fluctuates, as bursty applications behavior have a smaller impact. Moreover, as the data center gets bigger, the differences between utilization of the different resources (CPU, Memory, I/O) become smoother. Regarding performance degradation (two right figures), when the data center is big enough they are less frequent and have a smaller impact into the performance.

2) *Workload mixture impact*: The application type also impacts the performance. Bursty applications increases the amount of VMs that can be accepted, as well as the risks for overloads. An evaluation of this impact is summarized in Figure 10. With higher ratios of bursty applications, the overall utilization increases, whereas utilization in the most saturated resource dimension remains constant. However, bursty applications are usually also less predictable. The increased utilization thus results in more overload problems, as illustrated in figures 10 (c) and (d). However, the frequency and magnitude of these are only a few percent even with 75% of bursty applications.

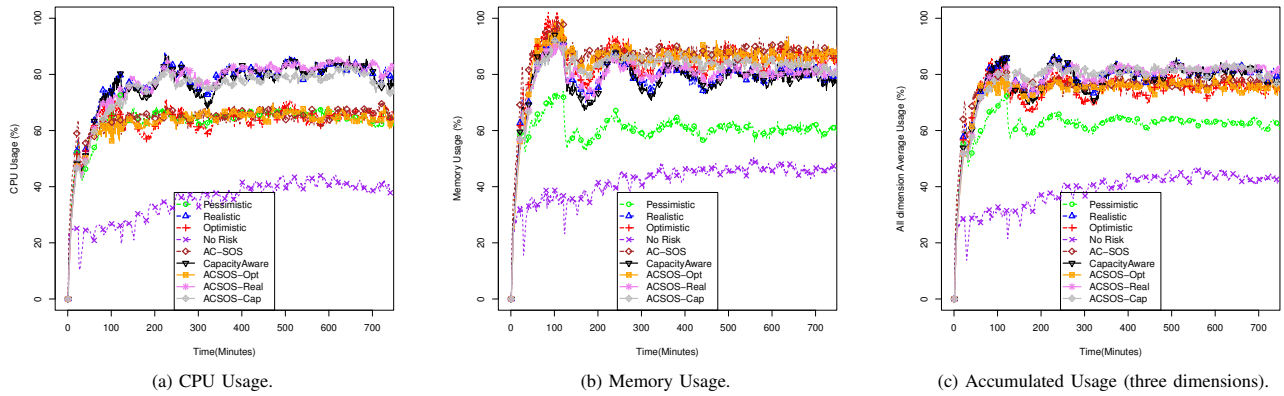


Fig. 7: Utilization comparison for the different admission control techniques.

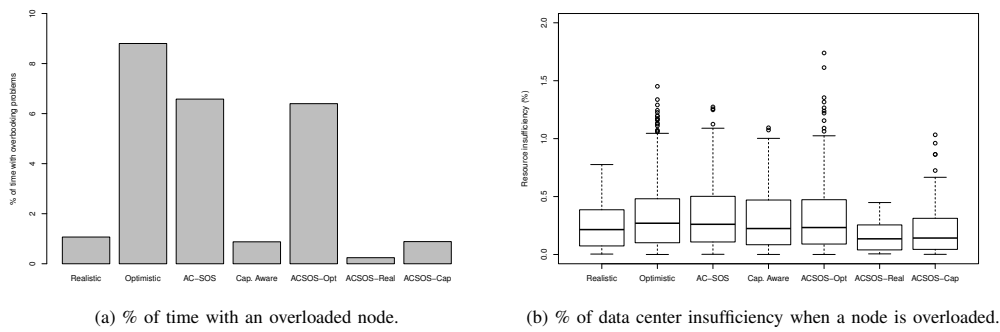


Fig. 8: Frequency and magnitude of overload problems for admission control algorithms.

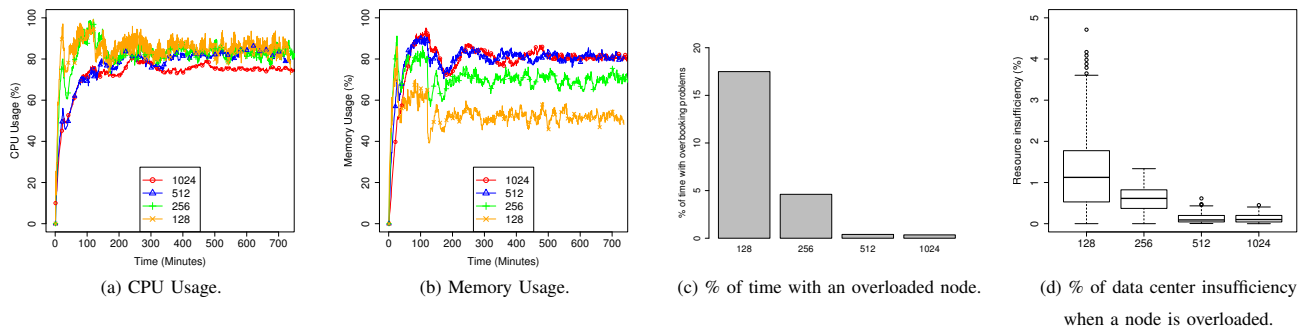


Fig. 9: Impact of different data center sizes (ACSOS Cap. Aware).

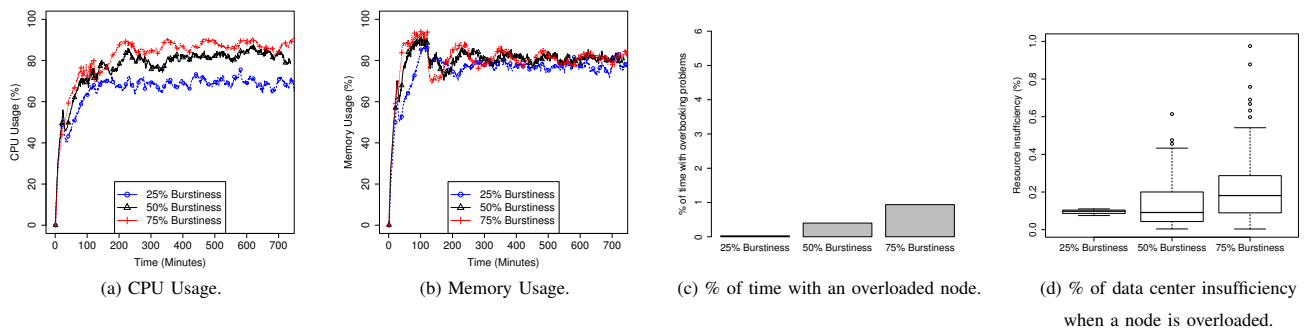


Fig. 10: Impact of different workloads mixture (ACSOS Cap. Aware).

3) *Risk Threshold impact*: Finally, we study the impact that varying it between 0.3 and 0.75 by steps increment of 0.05 (for changing the risk threshold has into the utilization achieved,

clarity only 0.4, 0.5, and 0.6 are depicted in the subsequent figures as the trend is the same). In order to just measure the impact that the established risk threshold has, we compare the different threshold only for Capacity Aware - a technique that is not using ACSOS methods. This may alter the results obtained as it could decrease the performance degradation problems by rejecting some services that would be accepted if only the risk values were taken into account.

Figure 11 depicts the results obtained. Figures 11 (c) and (d) show how both the probability and impact of having performance degradations increases with higher risk thresholds. On the other hand, regarding resource utilization (Figure 11 (a) and (b)) we see that the higher the threshold is, the higher the utilization is. However, this only applies to one of the dimensions (the most saturated one). Once the threshold is big enough (in this case 0.5), it is not worthy to keep increasing it. This leads to only slightly better utilization in one capacity dimension, but at expense of rather lower utilizations in the others (no increase in accumulated utilization) as well as increased overload problems. To sum up, the best solution for threshold setting is to have dynamic thresholds that autonomously change depending on the system behavior.

IV. RELATED WORK

Various approaches to cloud admission control have been suggested recently. Konstanteli et al. take a probabilistic approach to a combined scheduling and admission control problem formulated using mixed-integer non-linear programming [17]. In their work, the elastic service demand is modeled using cumulative distribution functions. Work on application level include a contribution by Leontiou et al. [18], who propose an adaptive feedback scheme with an application queue model to prevent overload of cloud services. Ashraf et al. propose a combination of noise-filtering and load predictions for session-based admission control in multi-tier servers [19].

Overbooking techniques as such have been applied in various fields as diverse as bandwidth allocation [20], airline yield management [21] and parallel computer scheduling [22]. Urgaonkar et al., propose techniques to overbook cluster resources in a controlled way, guaranteeing applications performance even despite overbooking [12]. Nevertheless, they assume that users provide information regarding the degree of overbooking that their applications may tolerate as well as their time periods, which may be known by the users in a cluster environments but is not available for cloud infrastructures. There are also examples of risk evaluation and SLA management applied to Grids, such as the one presented by Djemame et al. in [23]. A more recent study focusing on clouds analyze the risks of overbooking resources and proposes a threshold-based overbooking scheme [24]. The trade-off between overbooking and performance degradation is closely related to SLA management, studied e.g., by Breitgand et al. [25] who propose to extend standard availability SLAs to also include probability of successfully launching additional VMs (model based on CPU usage). They present an algorithmic framework that uses cloud effective demand to

estimate the total physical capacity required for performing the overbooking.

All kind of overbooking systems needs insight in future resource usage to avoid performance degradation. The literature on resource behavior prediction within highly distributed systems such as grids and clouds is very rich. A survey of several prediction techniques is presented in [26]. Examples of techniques include adaptive methods [27], state-space models [28], exponential smoothing [6], and use of control schemes for self-tuning for improved forecasts [29]. As predictions cannot be totally accurate in all situations, recovery mechanisms are needed when problematic situations occur. Beloglazov et al. [30] propose a Markov chain model and a control algorithm for the problem of host overload detection as a part of dynamic VM consolidation. Their system detects when hosts are overloaded and then perform the needed migrations to less loaded resources. This kind of migration approach complements overbooking techniques to avoid performance degradations upon mispredictions.

Once the decision about accepting and overbooking a new request is taken, the next step is deciding where to allocate it by studying the suitability of co-allocating the VM into the same physical node(s). On this topic, He et al. [31] present a multivariate probabilistic model for improving resource utilization for cloud providers. VM (anti)affinity rules are used to avoid repeating poor performance in the future. A similar approach is taken by Meng et al. [32], who propose a joint VM provisioning approach that, based on estimates of the aggregate VM capacity requirements, allocates and consolidates VMs. However, their work only takes into account CPU usage and perfect predictions about future workload behavior is assumed.

V. CONCLUSIONS

Resource utilization problems arising from the elastic nature of cloud applications can be solved by resource overbooking. Admission control mechanisms are needed to decide whether overbooking can be performed without impacting the performance of the already deployed VMs. This is a non-trivial decision with a strong prediction component. Our admission control algorithms evaluate the long term impact of the allocation decisions by means of prediction techniques combined with fuzzy methods for aggregating risk assessments. We demonstrate the advantage of using different risk degrees for different hardware capacity dimensions of VMs. Our experiments show that data center utilization is not only increased in overall but also harmonized across the capacity dimensions. On average, resource utilization is increased by a factor of two with overload problems occurring only a few percents of time with magnitudes less than one percent.

Future directions for this work include reusing our risk assessment tools for SLA negotiations. This would enable providers to specify different prices depending on the risk to be taken, or using different threshold risk values depending on the penalty to be paid in case of SLA violation, i.e., the greater the penalty the more pessimistic admission control should

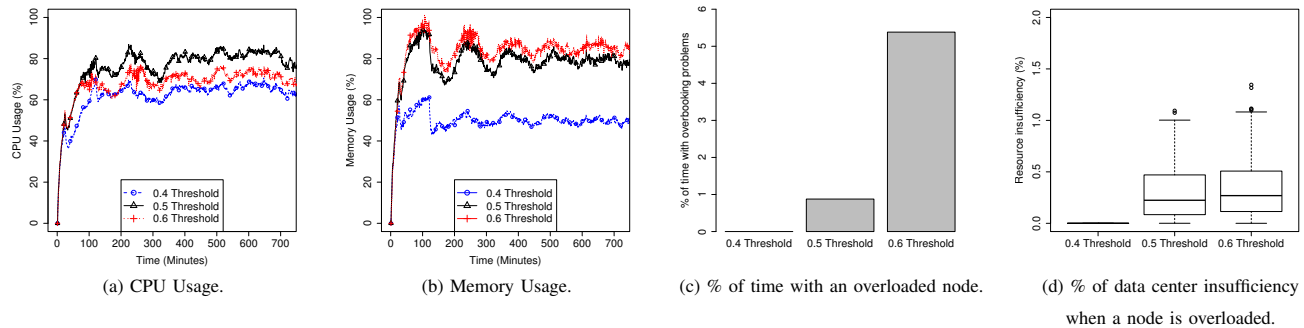


Fig. 11: Impact of different risk thresholds (CapacityAware algorithm).

be. Moreover, here the focus is on measuring the likelihood of overload problems rather than their possible impact. We plan to evaluate their real impact, instead of only measuring their frequency and size. Regarding our resource management framework, we plan to study affinity functions that aids the scheduling system in deciding which applications to co-allocate together, allowing it to further resource utilization increment and perform overbooking with even lower risk.

REFERENCES

- [1] The NIST Definition of Cloud Computing, Web page at <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>, Visited 2013-05-30.
- [2] D. Gmach, J. Rolia, and L. Cherkasova, "Selling t-shirts and time shares in the cloud," in *Intl. Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2012, pp. 539–546.
- [3] W. Cirne and F. Berman, "A comprehensive model of the supercomputer workload," in *Intl. Workshop on Workload Characterization*, 2001, pp. 140–148.
- [4] L. Tomás and J. Tordsson, "Improving Cloud Infrastructure Utilization through Overbooking," in *Cloud and Autonomic Computing Conference (CAC)*. ACM, 2013.
- [5] A. Sulistio, K. H. Kim, and R. Buyya, "Managing cancellations and no-shows of reservations with overbooking to increase resource revenue," in *Intl. Symposium on Cluster Computing and the Grid (CCGrid)*, 2008, pp. 267–276.
- [6] L. Tomás, A. Caminero, C. Carrión, and B. Caminero, "Exponential Smoothing for Network-aware Meta-scheduler in Advance in Grids," in *6th Intl. Workshop on Scheduling and Resource Management on Parallel and Distributed Systems (SRMPDS)*, 2010, pp. 323–330.
- [7] J. Heo, X. Zhu, P. Padala, and Z. Wang, "Memory overbooking and dynamic control of xen virtual machines in consolidated environments," in *11th IFIP/IEEE IM Conference*, 2009, pp. 630–637.
- [8] P. Vojtáš, "Fuzzy logic programming," *Fuzzy Sets and Systems*, vol. 124, no. 3, pp. 361–370, 2001.
- [9] J. Medina, M. Ojeda-Aciego, and P. Vojtáš, "Similarity-based Unification: a multi-adjoint approach," *Fuzzy Sets and Systems*, vol. 146, pp. 43–62, 2004.
- [10] P. Julián, G. Moreno, and J. Penabad, "Operational/Interpretive Unfolding of Multi-adjoint Logic Programs," *Journal of Universal Computer Science*, vol. 12, no. 11, pp. 1679–1699, 2006.
- [11] P. Morcillo, G. Moreno, J. Penabad, and C. Vázquez, "A Practical Management of Fuzzy Truth Degrees using FLOPER," in *4th Intl. Symposium on Rule Interchange and Applications*, 2010, pp. 119–126.
- [12] B. Urgaonkar, P. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in shared hosting platforms," in *OSDI*. ACM, 2002, pp. 239–254.
- [13] C. Vázquez, L. Tomás, G. Moreno, and J. Tordsson, "A fuzzy approach to cloud admission control for safe overbooking," in *Intl. Workshop on Fuzzy Logic and Applications (WILF)*. Springer Verlag, LNCS, 2013.
- [14] Nagios - The Industry Standard in IT infrastructure Monitoring, Web page at <http://www.nagios.org/>, Visited 2013-03-13.
- [15] libvirt: The virtualization API, Web page at <http://libvirt.org/>, Visited 2013-03-13.
- [16] LTTng Project. Linux Trace Toolkit - next generation, Web page at <http://ltng.org/ltng2.0>, Visited 2013-07-16.
- [17] K. Konstanteli, T. Cucinotta, K. Psychas, and T. Varvarigou, "Admission control for elastic cloud services," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 41–48.
- [18] N. Leontiou, D. Dechouniotis, and S. Denazis, "Adaptive admission control of distributed cloud services," in *Intl. Conference on Network and Service Management (CNSM)*. IEEE, 2010, pp. 318–321.
- [19] A. Ashraf, B. Byholm, and I. Porres, "A session-based adaptive admission control approach for virtualized application servers," in *Intl. Conference on Utility and Cloud Computing (UCC)*. IEEE, 2012, pp. 65–72.
- [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] 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.
- [22] G. Birkenheuer, A. Brinkmann, and H. Karl, "The gain of overbooking," in *Job Scheduling Strategies for Parallel Processing*, ser. LNCS, 2009, vol. 5798, pp. 80–100.
- [23] K. Djemame, J. Padgett, I. Gourlay, and D. Armstrong, "Brokering of risk-aware service level agreements in grids," *Concurr. Comput. : Pract. Exper.*, vol. 23, no. 13, pp. 1558–1582, 2011.
- [24] R. Ghosh and V. K. Naik, "Biting off safely more than you can chew: Predictive analytics for resource over-commit in iaas cloud," in *5th Intl. Conference on Cloud Computing*, 2012, pp. 25–32.
- [25] D. Breitgand, Z. Dubitzky, A. Epstein, A. Glikson, and I. Shapira, "Sla-aware resource over-commit in an iaas cloud," in *8th Intl. Conference on Network and Service Management (CNSM)*, 2012, pp. 73–81.
- [26] 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.
- [27] 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.
- [28] 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.
- [29] L. Tomás, A. C. Caminero, C. Carrión, and B. Caminero, "Network-aware meta-scheduling in advance with autonomous self-tuning system," *FGCS*, vol. 27, no. 5, pp. 486 – 497, 2011.
- [30] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE TPDS*, vol. 24, no. 7, pp. 1366–1379, 2013.
- [31] S. He, L. Guo, M. Ghanem, and Y. Guo, "Improving resource utilisation in the cloud environment using multivariate probabilistic models," in *5th Intl. Conference on Cloud Computing (CLOUD)*, 2012, pp. 574–581.
- [32] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing," in *Autonomic Computing (ICAC)*, 2010, pp. 11–20.