SLA-aware Resource Over-Commit in an IaaS Cloud

David Breitgand Zvi Dubitzky Amir Epstein Alex Glikson Inbar Shapira
Cloud Operating System Technologies
IBM Haifa Research Lab, Haifa, Israel
Email: {davidbr, dubi, amire, glikson, inbar_shapira}@il.ibm.com

***Abstract*—Cloud paradigm facilitates cost-efficient elastic computing allowing scaling workloads on demand. As cloud size increases, the probability that all workloads simultaneously scale up to their maximum demand, diminishes. This observation allows multiplexing cloud resources among multiple workloads, greatly improving resource utilization. The ability to host virtualized workloads such that available physical capacity is smaller than the sum of maximal demands of the workloads, is referred to as over-commit or over-subscription.**

Naturally, over-commit implies risk of resource congestion. Therefore, there is a tradeoff between improving resource utilization by increasing an over-commit ratio and exposing the infrastructure provider and customers to the risk of resource congestion. In this work, we observe that while resource multiplexing naturally occurs in the cloud, the risks associated with exploiting it for higher levels of cloud utilization, are not transparent to the customers. We consider workloads comprising elastic groups of Virtual Machines (VMs). We suggest that cloud providers would extend a standard availability Service Level Agreement (SLA) to express the probability of successfully launching a VM (to expand a workload), complementing the current practice of offering a standard SLA on availability of VMs which are already successfully launched.

Using the proposed extended availability SLA, we introduce a notion of the cloud effective demand, which generalizes previously introduced notions of effective size of a single VM and effective bandwidth of stand-alone and multiplexed network connections. We propose an algorithmic framework that uses cloud effective demand to estimate the total physical capacity required for SLA compliance under over-commit. We evaluate our proposed methodology using simulations based on the data collected from a real private cloud production environment.

I. Introduction

One of the more important and attractive features offered by Cloud computing to the customers is ”just in time” scalability. In the Cloud model, resources are provided on demand, aiming to avoid over-provisioning and under-utilization. In Infrastructure as a Service (IaaS) compute clouds, which are in the focus of this study, the customers rent Virtual Machine (VM) instances paying for their usage according to a ”pay-as-you-go” model.

The economic incentives to release unneeded VMs as soon as possible, set by the public and community cloud charge back models, naturally cause high variability in resource demand. In private clouds, whose popularity increasingly grows, charge back schemes may not be implemented, but policies are used that require users disbanding unused VMs and re-acquiring them when needed.

Currently most IaaS providers offer VM instances from a catalogue suggesting a number of discrete hardware configurations. For example, Amazon EC2 offers ”small”, ”medium”, ”large”, ”extra large” and a few other VM configurations, where each configuration represents a different VM sizing and is charged using different instance hour rate. These VM configurations are referred to as *types*.

We refer to VM collections of possibly different types, belonging to the same user account or a service as *elastic workloads* that can be expanded and shrunk on demand either manually or by means of an elasticity controller [1]. The elasticity ranges of the users are, usually, limited by quotas stipulating the maximum number of VMs that can be run simultaneously by a given user.

Once a VM of a certain type is provisioned, the resources associated with the corresponding VM type (e.g., CPU, memory) will typically be allocated solely for that VM instance[1] We refer to this model as allocation by *nominal capacity*.

In this work, we consider the nominal capacity resource allocation model as presented to the customers for common types of VMs. Even though some hypervisors have built-in capabilities to share computational resources [3], [4], this is usually transparent to the upper layer cloud management systems and the cloud customers.

Because of elastic workload variability, statistical multiplexing naturally occurs in the cloud. Thus, the peaks of total resource demand expressed in terms of the number of VMs of different types that are simultaneously run by all workloads, are much lower than the sum of maximum theoretical demands that would have occurred in the cloud if all workloads would use their elasticity ranges simultaneously to a full extent. Since the cost efficiency of the cloud model critically depends on ability of the cloud providers to leverage statistical multiplexing, the clouds often operate in the over commit mode by design. However, the risks associated with this fact are not disclosed to the cloud customers and specific methods of establishing over-commit ratios represent a highly sensitive business level information.

Common cloud practices use availability SLAs to stipulate a percentage of VM availability computed over single billing period, where availability usually means that VM successfully responds to ping. This tacitly implies that VM would be allocated resources in accordance with the types' nominal specifications for the same percentage of time. An attention

[1]Other resource allocation models also exist. For example, in case of Amazon ”micro” instances [2], VMs are guaranteed minimal resource allocation, with possibility to expand this allocation on demand, conditioned on resources availability at the host.

978-3-901882-48-7

should be paid to the fact that availability SLAs refer to VM instances that have been successfully launched and, in general, do not specify any guarantees on the probability of successfully launching a VM, when needed for just in time elasticity. It is tacitly assumed that customers can create VMs, subject to quota, 100% of the time.

Depending on a non-disclosed method of capacity planning of a specific provider this might or might not be a truly enforced guarantee, but in this study we assume a benevolent and faithful IaaS provider that plans capacity for its cloud in such a way as to *fully* provision capacity for the maximum observed or projected demand (calculated using the nominal resource allocation model), after taking statistical multiplexing gain, naturally occurring in its data, into account. In other words, we assume that a cloud provider indeed procures the minimum physical capacity to guarantee effectively zero risk level of resource congestion, when launching a new VM.

To make this tacit assumption an explicit guarantee, we propose to extend the standard availability SLAs to stipulate the probability of successfully launching VMs during a billing period, such that the risks associated with cloud over-commit would be transparent to the customers similarly to the risks of unavailability. Since it is, in fact, very difficult to maintain 0 risk level deterministically, we suggest that the implicit guarantees on VM launch success would be relaxed and made no stricter than the standard availability guarantees. We elaborate on extended SLAs in Section III. We abbreviate extended availability SLA as *eSLA* in the rest of the paper.

In addition to an obvious benefit to the users (transparent and realistic risks), this proposition also benefits cloud providers by allowing them to further increase over-commit ratio, subject to eSLA conformance.

To implement our proposal computationally and storage efficiently, while maintaining sufficient accuracy, we propose a simple method of estimating total *effective nominal demand* of a cloud and use it for capacity sizing and placement reservation plan that is compliant with eSLA.

In our approach, a *momentary nominal demand* of a workload is calculated as the number of VM instances of different types being run by the workload at a specific time instance. Summing up the momentary nominal demand of all users, we obtain the momentary nominal demand of the cloud. We propose a simple method for calculating effective nominal demand from the time series of the momentary nominal demand.

While conceptually similar to an over-commit strategy that over-subscribes hosts' resources based on the actual utilization levels of VMs that received more attention in the literature recently [5]–[8], we treat an entire pool, datacenter or distributed cloud as a single overcommit domain, while respecting nominal capacity allocations of VM types. Our proposed method can handle multi-dimensional resource allocation simultaneously by reserving placement slots for VMs according to the nominal capacity specification. While such approach may result in lower over-commit ratios than those achievable with per host actual utilization over-commit, our method offers much more robust performance guarantees (via respecting nominal allocations), requires dramatically less amount of monitoring (actual demand over-commit methods require per VM monitoring of utilization), keeps VM migrations to the minimum, requires less computational power and is more transparent to the cloud customer.

One of the more difficult problems in managing over-committed resources is estimating *hidden resource demand* properly. The hidden demand is the portion of total demand that would be satisfied, if a cloud was dimensioned appropriately. While in actual utilization over-commit method estimating hidden demand is a difficult task [9], in the proposed nominal demand over-commit strategy, hidden demand can be estimated relatively easily by monitoring failed VM create API calls[2].

To the best of our knowledge this work is the first to propose eSLA compliant nominal demand over-commit.

A. Contributions

Our contribution in this paper are as follows.

- We introduce novel extended availability SLAs (eSLAs);
- We define a Risk-aware Cloud Capacity Planning problem in the context of eSLAs;
- We propose an algorithmic framework for solving this problem and substantiate it with a novel algorithm that generalizes previously introduced notions of effective size of a single VM and effective bandwidth of stand-alone and multiplexed network connections;
- We evaluate our methodology using simulations based on the data collected from a real private cloud production environment and demonstrate that the nominal demand over-commit strategy provides for considerable savings in physical capacity, while adhering to eSLAs. In the simulation study we focus on the CPU as the bottleneck resource.

The rest of this paper is organized as follows. In Section II we discuss related work. Section III discusses proposed eSLAs. In Section IV we define the model and the problem formally. In Section V we discuss an overall reference framework, where our proposed methodology fits. In Section VI we describe an instance of the proposed framework to efficiently use effective nominal demand for capacity planning and placement. We present our simulation study in Section VII and provide some conclusions and future work directions in Section VIII.

II. Related Work

The need to improve cost-efficiency by reducing capital investments into computing infrastructure and operational costs such as energy, floor space, and cooling drive the research efforts on VM consolidation [10]–[16] and motivate features

[2]It should be noted that most of the current IaaS cloud APIs support only one VM instance deployment/undeployment at a time. Therefore, when API calls requiring addition of VM instances fail, the cloud provider might have difficulties understanding whether subsequent API calls are the new ones or just the retries of the previously failed ones. Thus the provider might not know what is the actual number of instances required by the client. We expect that in the future this problem might be alleviated by advancement of the IaaS APIs that will include explicit *resize* call specifying the number of VM instances of each type that should be added/removed to/from a workload. In the meantime, we propose a simple heuristic that allows to estimate total nominal demand from the API call trace reasonably well.

of the products such as [17]–[19]. In most of the research work, VM consolidation is regarded as a classical bin packing problem where resource consumption is inferred from historical data or predicted using forecasting techniques. In addition to the primary optimization goal, which is minimizing the number of hosts, secondary goals such as migration minimization, performance optimization and other are considered. A variant on the classic packing algorithms such as First Fit (FF) and First Fit Decreasing (FFD) [20] are often used to achieve practical solutions.

In [16] a target over-commit ratio is part of the input. The resource allocation system operates in a reactive manner to mitigate possible overload by live migrating and preempting VMs. Our approach, in contrast, uses target overload risk as an input and strives to avoid overload by properly setting over-commit ratio using capacity planning.

A common drawback of many traditinoal approaches is that they consider each VM resource demand separately, which may lead to low resource utilization [21]. Recent work uses statistical multiplexing among the VMs to improve resource utilization, such as CPU [5], [6] and network bandwidth [7], [8].

All previous work exploit VM multiplexing to overcommit resources at the host level based on the actual resource utilization of VMs. In contrast, our approach leverages statistical multiplexing among the sets of VM types run by different users/services to over-commit pool of physical resources used to provide a compute cloud service. To the best of our knowledge, this is the first work, systematically exploring statistical multiplexing for overcommitting pool of resources according to VM nominal demand.

III. Extended Availability SLAs

A typical standard availability SLA stipulates that a VM will be available (i.e., will respond to ping) with probability p calculated over an evaluation period, which is usually a single billing period, such as a single month. However, when it comes to provisioning of new VMs, it is not clear what kind of guarantees are expected from the cloud provider in terms of success rate of provisioning requests, and we are not aware of cloud providers that explicitly formalize these guarantees in their SLA.

In general, handling of the failed VM create API calls may vary from the most conservative one, where failed provisioning requests are treated by the cloud providers as if the instance has been provisioned, but became unavailable immediately, to the most opportunistic behavior, in which the user may receive a notification that the request failed due to an unspecified error, and that they should try again later. In the former case, the standard availability SLA applies. In the latter case, the IaaS provider bears no responsibility at all.

A specific behavior is likely to be derived from the mechanism and policies used by the cloud provider to plan capacity in the cloud. For example, in order to support the conservative behavior, cloud provider might need to maintain capacity at the level of the maximal total demand expected in the cloud – effectively ensuring close to 100% success ratio of provisioning requests. Naturally, the cost of capacity unit in this case will be higher than the cost for a provider which over-commits capacity more aggressively at the price of higher risk of capacity congestion.

We propose to formalize the risk of VM create failure due to capacity congestion by introducing a specific clause stipulating a percentile of success in launching new VM instances, subject to elasticity quota that applies to a group to which a user account belongs. The probability is calculated over a single billing period, as defined by the standard SLA.

For simplicity, we assume single elasticity range $[r_{min}, r_{max}]$ for all the workloads in a given cloud[3]. We suggest that the probability p of launching VMs of *any* type, subject to the elasticity range, would be $p \leq p_a$, where p_a is VM availability percentile contracted by the standard availability SLA.

To illustrate, with eSLA of $p = 0.99$, calculated over a period of 1 month, cloud provider can deny provisioning requests for 7.2 hours, accumulated over the 1-month period, plus all the time that the workload is at its maximal capacity.

Cloud provider will publish the guaranteed eSLA percentage p, as well as a sampling period used for compliance calculation. This information can be used by a workload owner together with the statistics about failures of his/her provisioning requests, to check compliance with the published eSLA. The cloud provider is expected to calculate the observed percentage of VM create failures by using its internal logs and publish this value. The user can cross-validate this value using his own observations of the API calls. If the published failure percentage exceeds $1 - p$, the user will be eligible for eSLA violation fees.

IV. Model and Problem Formulation

In this section we give the formal definition of the model and formulate our problem.

We consider elastic workloads comprising multiple VM instances of different types, where the number of instances in the workload may change dynamically.

Let n be the total number of workloads. Let $Y_{j,i}$ be the random variable representing the number of VM instances of type i used by workload j. Then, $X_i = \sum_{j=1}^{n} Y_{j,i}$ is a random variable representing the total number of VM instances of type i. We do not make any assumptions on the form of the distributions of these variables, assuming that that $Y_{j,i}$ are arbitrarily distributed.

Definition 4.1: (Nominal Demand) We define nominal demand of the cloud as the array of random variables $\bar{X} = (X_1, X_2, ..., X_l)$, where l is the number of VM types.

Definition 4.2: Effective Nominal Demand

Effective nominal demand is a vector $\bar{D}$, such that $Pr(\bigwedge_i (X_i \leq D_i)) \geq p$.

Obviously, there are many vectors $\bar{D}$ that satisfy Definition 4.2. Clearly, if the cloud has sufficient physical capacity to place the number of VMs of each type as specified by $\bar{D}$, the cloud provider will comply to the target risk level. Naturally, we are interested in vector $\bar{D}$ that minimizes total

[3]This assumption is consistent with today's common practice to have a single standard availability SLA for all the VMs in a given cloud. However, additional SLA diversification is also possible in the future.

capacity. The notion of effective nominal demand is central to our approach. In the next section we show how to obtain an estimate of $\bar{D}$ that considerably reduces total capacity in comparison to full provisioning.

Definition 4.3: (Risk-aware Cloud Capacity Planning Problem) Given a collection of random variables $Y_{j,i}$, representing elastic workloads, probability of success p pertaining eSLA offered by the cloud, find minimal total capacity of physical hosts that is required to satisfy eSLAs of the cloud.

V. Risk Aware Capacity Planning Framework

In this section we put the Risk-aware Cloud Capacity Planning Problem in the context of a general Cloud Capacity management process.

Traditional capacity management has been seen as a CAPEX management process over disjoint silos of infrastructure (servers, network, storage). As the popularity of the cloud paradigm increases, capacity management becomes as much an SLA management process as it is a CAPEX management one. "This stems from the possibility of SLA breaches because of oversubscribed equipment or lack of resources to support the bursting of existing services or the instantiation of new services, which will inversely impact the user experience of a cloud service. The self-service nature of the cloud and the increase in virtualization and automation make capacity management more complex as providers can no longer over dimension their infrastructure to support peak usage, as peak usage will vary dramatically based on consumer requirements" [22].

A generic cloud capacity management framework comprises three main components: Data Collection, Forecasting, and Analysis. The output of analysis is an input for capacity procurement system.

Our methodology requires the following extensions to the existing practices of cloud capacity management.

- **Data Collection:** captures cloud API calls for creating and terminating VM instances. This information is used for creating time series of the cloud nominal demand, where each data point in the time series reflects the momentary nominal demand of the cloud at the time of sampling used to create the data point. More specifically, we assume that we sample the momentary nominal demand at fixed frequency. A data point at time t_k is obtained as $\bar{X}(t_k)$, where $X_i(t_k) = X_i(t_{k-1}) + C_i(t_{k-1}, t_k) - T_i(t_{k-1}, t_k)$, where $C_i(t_{k-1}, t_k)$ is the number of VM instances of type i created and $T_i(t_{k-1}, t_k)$ is the number of VM instances of type i terminated in the interval $[t_{k-1}, t_k]$, respectively. This time series is an input to the Estimator of Effective Nominal Demand. When minimal analysis interval is fixed, the monitoring system produces the more compact time series of $E[X_i]$ and $E[X_i^2]$ over the minimal analysis interval.

- **Forecast Subsystem:** In general, trending and forecasting demand in the cloud is a difficult problem, because of the nature of the cloud, where due to independence of user demands, patterns might be difficult to establish [22]. In this study we do not propose a new mechanism for trending and forecasting. Advanced techniques in this area [23] can be leveraged by our proposed methodology. In this work, the trend in data was close to flat that allows us effectively treating the current nominal demand as the future one.
- **Estimator of Effective Nominal Demand:** this is the main differentiator of our proposed method. It enhances the functionality of the generic Analysis component of the cloud management framework. The estimator gets momentary nominal demand time series as its input and produces an *effective nominal demand* of the cloud, reflecting a steady state of the cloud in terms of nominal demand with respect to a given probability p. We describe an efficient algorithm for calculating effective nominal demand in Section VI. The output of the estimator of effective nominal demand serves as input for the placement engine component.
- **Placement Engine:** the placement engine component is another contribution to the generic Analysis phase. It uses some placement algorithm to reserve placement 'slots' for different VM types for placing concrete VM instances on the hosts for concrete momentary demand. The placement engine obtains a specific hardware configuration that would be sufficient with probability p to satisfy the nominal demand of the cloud.

The structure of this framework allows to plug in different effective nominal capacity estimators and different placement algorithms. The latter is especially important, since due to the multiple optimization objectives occurring in different cloud management contexts, no single placement algorithm can satisfy all the needs of all cloud providers.

VI. Algorithm

In this section we present a specific instantiation of the algorithmic framework for Risk-aware Cloud Capacity Planning problem explained in Section IV. The algorithm's pseudo-code is shown in Figure 1.

Under our assumptions, each X_i that is part of the nominal demand of the cloud $\bar{X} = (X_1, X_2, ..., X_l)$, is a sum of a large number of random variables $Y_{j,i}$, reflecting relatively small contributions by a workload to the sum.

Although there might exist dependencies between some workloads in the cloud, for *sufficiently large* cloud with thousands of workloads, these dependencies can be ignored and X_i can be treated as independent variables.

Making this assumption, we observe that the distribution of each random variable X_i that is part of the nominal demand of the cloud asymptotically converges to normal distribution due to the Central Limit theorem [24]. Our evaluation study confirms this assumption.

Using this observation, we can estimate effective nominal demand in a computationally efficient manner as follows. Let, as before, assume that the probability of successfully launching a VM of type i as stipulated by eSLA of Section III is p. To conform to this probability, a cloud provider needs to have a number of physical hosts that is sufficient to feasibly place the effective nominal demand.

From the union bound we obtain that $Pr(\vee_i(X_i > D_i)) \leq \sum_{i=1}^{l} Pr(X_i > D_i)$. Letting $Pr(X_i > D_i) \leq p_i$, where $p_i = (1-p)\cdot w_i$ and w_i is such that $\forall i, w_i \in [0,1]$ and $\sum_{i=1}^{l} w_i = 1$, we obtain that $Pr(\wedge_i(X_i \leq D_i)) \geq p$.

In a direct application of the union bound, we set $w_i = 1/l$ for every i. Another possibility to calculate the weights w_i is by using relative popularity of VM types.

The effective nominal demand for VM of type i corresponds to the p_i-quantile of the cumulative distribution function (CDF) of X_i, i.e., to the minimum value x, such that $Pr(X_i \leq x) = p_i$.

Transforming X_i into a standardized normal variable $Z_i = \frac{X_i - \mu_i}{\sigma_i}$, we can write $Pr(\frac{X_i - \mu_i}{\sigma_i} \leq Z_{p_i}) = p_i$, where μ_i and σ_i are the mean and standard deviation of X_i respectively, and Z_{p_i} is the p_i-quantile of standard normal distribution. From this we can obtain an estimation of the effective nominal demand as shown in Equation 1.

$$D_i = \lceil \mu_i + Z_{p_i}\sigma_i \rceil, \tag{1}$$

where D_i is rounded up, since the nominal demand is integral. The distribution of X_i is truncated at 0 because, obviously, a workload cannot provision a negative number of VMs. An implication of this is that the first moment of the actual distribution will be larger than the one used in Equation 1, while the second moment will be smaller than the one predicted by Equation 1.

Since X_i follows approximate 0-truncated normal distribution, the standardized left-truncated $Z_i = \frac{X_i - \mu_i^{'}}{\sigma_i^{'}}$ follows approximate standard normal distribution, truncated at $-\frac{\mu_i^{'}}{\sigma_i^{'}}$, where $\mu_i^{'}$ and $\sigma_i^{'}$ are the mean and standard deviation of the truncated normal distribution, respectively.

Let $\Phi(.)$ denote CDF and $\phi(.)$ denote the probability density function (PDF) of the standard normal distribution.

From the definition of truncated normal distribution [25] we obtain that

$$\mu_i^{'} = \mu_i + \sigma_i \frac{\phi(-\frac{\mu_i}{\sigma_i})}{1 - \Phi(-\frac{\mu_i}{\sigma_i})} \tag{2}$$

and for $\sigma_i^{'}$ we obtain:

$$\sigma_i^{'} = \sigma_i \sqrt{1 - \frac{\phi(-\frac{\mu_i}{\sigma_i})}{1 - \Phi(-\frac{\mu_i}{\sigma_i})} \left(\frac{\phi(-\frac{\mu_i}{\sigma_i})}{1 - \Phi(-\frac{\mu_i}{\sigma_i})} + \frac{\mu_i}{\sigma_i} \right)} \tag{3}$$

Using $\mu_i^{'}$ and $\sigma_i^{'}$, we obtain that the effective nominal demand for VM type i as shown in Equation 4.

$$D_i = \lceil \mu_i^{'} + Z_{p_i}\sigma_i^{'} \rceil \tag{4}$$

A. Placement Reservation

After calculating an estimation of effective nominal demand $\bar{D}$ of the cloud with respect to the target success probability p, the next step is to find the minimal number of hosts that would be required to place $\bar{D}$.

This essentially requires to solve a bin packing problem. Since bin packing problem is NP-hard, an approximation algorithm, such as First Fit Decreasing (FFD) can be used to obtain the number of hosts required.

To obtain a very simple and computationally efficient risk-aware placement algorithm, we treat the values in vector $\bar{D}$ as the number of "slots" of different types that should be reserved in the cloud to conform to eSLA. Namely, we reserve the slots for VM in decreasing order of VM sizes according to the value of effective nominal demand for this size calculated using our normal approximation. A sample placement reservation resulting from running this heuristic is shown in Figure 1. When a new VM arrives, it is placed to any of the available slots corresponding to its type. With probability p such a slot should exist thanks to our capacity planning.

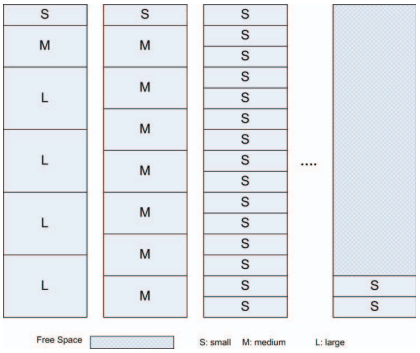

Fig. 1. An example of placement slots reservation

The usage of this reservation scheme is two-fold. First, it can be used as a reference point for capacity planning, since it produces a minimal number of bins. Second, it can be used as an actual placement algorithm.

An advantage of this placement algorithm is its extreme simplicity and efficiency, which might be important when coping with high VM arrival rates. In addition, this algorithm does not require migrating VMs from their current locations to accommodate the newly arrived ones as long as effective nominal demand estimation remains accurate, which minimizes performance impact on the workloads running inside the VMs. Since effective nominal demand may gradually change over time, periodic VM migrations to consolidate VMs on a minimum number of bins might be required, provided that the benefit accrued from capacity savings dominates performance impact due to migrations.

A disadvantage of this placement approach is that while the bulk of savings in capital costs and operational costs is already achieved by risk-aware overcommitting of resources as described, the approach ignores additional transient opportunities to save operational costs by temporarily migrating VMs to additionally improve utilization level of the hosts and putting the temporarily freed hosts into a less power consuming stand-by mode. Note, however, that additional operational cost gains should be balanced against performance impact on VMs due to migrations.

An additional disadvantage of the method is that it ignores complex placement constraints among VMs that might exist,

such as collocation and anti-collocation of VMs belonging to the same workload and among different workloads. It is, therefore, advisable to use the proposed placement algorithm in a resource management domain, where no hard constraints of this type exist. In many real life cases this is a viable approach.

To illustrate this point, consider different availability zones provided by physically separate host pools. When groups of VMs are launched, it might be required to place each VM in a separate availability zone (anti-collocation constraints, also known as high availability constraints). Effective nominal demand estimation, however, can be performed separately for each zone as described.

We defer investigating of the complex placement constraints and maximizing the net benefit of migration based opportunistic cost savings in the over-committed cloud to future work.

Algorithm 1: Risk-aware Cloud Capacity Planning Algorithm

Input: Historic momentary nominal demand time series for time window $[0, T]$: $\bar{X}_0^T$;
Target success probability as defined by eSLA: p.
Output: Number of homogenous bins (hosts) required to conform to eSLA

1 Calculate VM type weights w_i: $w_i \in [0, 1]$ and $\sum_{i=1}^{l} w_i = 1$ based on VM type popularity.
2 Calculate an estimation of effective nominal demand $\bar{D}$ as in Equation 4.
3 Run First Fit Descending (FFD) algorithm to reserve slots as follows:
4 Starting from the largest VM type slots to smallest, reserve the current VM type slot on the first bin, where the slot fits.
5 When the number of slots of the current type is exhausted, the algorithm proceeds to the next type.
6 If no more slots of the current VM type can be packed into either of the current bins, a new bin is opened.
7 Output the number of bins obtained.

VII. Experimental Evaluation

In this section we present a simulation study of our approach.

A. Methodology

We observed a medium size test and development private cloud that was recently introduced into production. The cloud is supposed to grow considerably in the future, following its popularity among developers and internal policies encouraging usage of virtual machines as opposed to acquiring hardware. The resource allocation model in this custom cloud requires allocating an integral number of virtual cores to VMs regardless of the actual utilization levels. There are a few VM sizes in the catalogue of the cloud, but only three types are used intensively, with the rest of VM types being used marginally. We, therefore, focus on these three VM types in our evaluation, labeling them ”small”, ”medium”, and ”large”, where ”small”, ”medium”, and ”large” require 1, 2, 4 vCPUs, respectively. The physical servers are homogenous, where each server has 24 vCPUs. The host have 128 GB of physical memory each. Thus, memory does not represent a bottleneck.

One of important questions faced by this cloud operations team is how much physical capacity should be provided to accommodate the users' demand with only minor resource congestion levels.

We observed the cloud for duration of continuous 20 days and collected statistics about the inter-arrival times and life

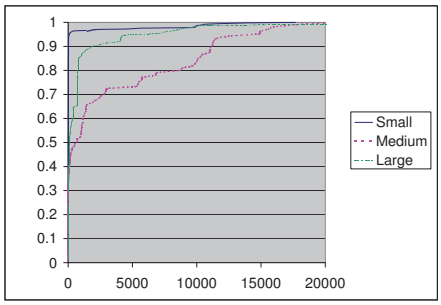

Fig. 2. CDF of VM lifetimes (min)

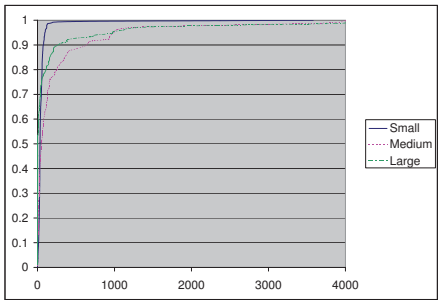

Fig. 3. CDF of VM inter-arrival times (min)

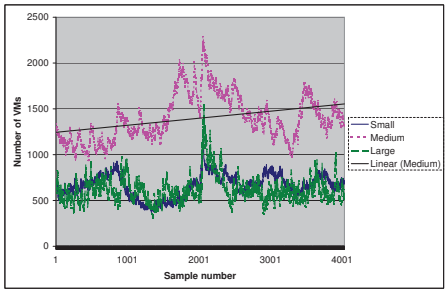

Fig. 4. An example of a simulated time series

times of VMs per type. The raw data cannot be presented at this point for business reasons. In Figure 3 and Figure 2 we present the CDF of inter-arrival and life-time of VMs per type, respectively. The inter-arrival times are scaled up uniformly from their raw values for all VM types, reflecting an expected future demand. The life-times are presented as is, reflecting the typical usage of launched VMs by developers and testers. In Figure 4, we show a sample simulated time series of the number of VMs per type. Each data point is obtained at resolution of 30 minutes. The trace represents a simulated momentary nominal demand time series of six months.

Only the last three months are used for analysis, reflecting a common practice of procuring physical capacity on a per quarter basis. As Figure 4 shows, there is a moderate linear upward trend for "medium" VM demand, while the overall trend for "small" and "large" VMs is, essentially, flat. Our raw traces were insufficient to establish whether different trends or seasonality exist in our data. Some recent studies in public clouds domain suggest, for longer traces, that an overall linear upward trend is usually present [26], while seasonality is difficult to establish [23]. Thus, our simulated data does not have seasonality.

One important phenomenon that we observed, were occasional flash crowds. The flash crowds, when creation of different VM types were strongly positively correlated, did not seem to follow a specific pattern and could be attributed to correlations among different projects. At the time of a flash crowd, the probability of successfully launching a VM dropped dramatically, which would have made a cloud provider – given that internal SLAs were established, or that it was a public test and development cloud – to violate its commitments and possibly pay penalties. Even in absence of explicitly stipulated penalties, a significant drop in VM launch success probability may result in inconvenience for the users, reducing their motivation to use the cloud service.

Presence of flash crowds in the raw trace considerably complicated our analysis, since they resulted in a large number of VM create API calls for which no life time of VMs existed in the data. These failures represent a hidden demand that would be satisfied if the cloud was dimensioned accordingly. To model rare flash crowds, we randomly selected short periods, where the inter-arrival times of all VM types considerably decreased, while the life times were assumed to continue following the empiric distribution obtained for actually run VMs. In other words, we assume that if VMs that failed creation were, in fact admitted to the cloud, the distribution of their lifetimes would be similar to that of VMs, that were indeed successfully created and run. This modeling approach results in occasional peaks as one can observe, for example, in Figure 4 between time 2000 and 2500.

To obtain statistically meaningful results, we repeat time series generation 20 times and show the average, standard deviation, maximal, and minimum performance of the proposed estimation method over the repeated experiments. The results are presented in the next subsection.

B. Results

In our experiments we measure the overcommit ratio actual congestion probability obtained using our method for target congestion probability $\alpha = 1 - p$, where p is probability of success specified by eSLA. We define the overcommit ratio as the ratio between the minimum number of bins required by full provisioning to satisfy the actually observed demand over the time series and the actual number of bins used by cloud provider.

Thus, when the actual number of bins is calculated using our proposed method, the overcommit ratio is the ratio between the minimum number of bins required to satisfy demand at all data points in the time series and the number of bins obtained by using Algorithm 1.

In order to calculate the minimum number of bins required by the full provisioning method (i.e., at all data points), it is required to run the placement algorithm, in our case the FFD algorithm, for each data point and to calculate the required momentary number of bins. The minimum number of bins required to satisfy demand with $\alpha = 0$ is the maximum number of bins obtained by the placement algorithm over all data points.

We evaluate the actual congestion probability as percentage of bins that exceed the effective number of bins suggested by our approximation method.

Table I summarizes the actual overcommit ratio and overflow probability obtained by Algorithm 1. The table shows the average, standard deviation, minimum and maximum overcommit ratio and overflow probability. We performed the experiments for three typical values of target congestion probability α: 0.01, 0.02 and 0.05. One random flash crowd was introduced.

The table shows that the obtained average overcommit ratio is at least 1.33. Moreover, the table shows that the overcommit ratio increases as the target congestion probability increases, as expected. This follows from the fact that the effective demand and ,thus, the effective number of bins decreases, when α increases. As one can see, the average actually obtained congestion probability for $\alpha = 0.01$ slightly exceeds the target one. However, for larger values of α the deviation of the obtained congestion probability from the target congestion probability decreases, and for $\alpha = 0.05$ the average congestion probability fells below the target one.

Table II summarizes essentially the same experiments with three random flash crowds simulated. As the table shows, while over commit ratio slightly decreases, and the actual congestion probability slightly increases, the results remain similar to those summarized in Table I, suggesting that our approach is robust.

α	Overcommit Ratio				Congestion Probability			
	AVG	STD	MIN	MAX	AVG	STD	MIN	MAX
0.01	1.33	0.097	1.15	1.59	0.018	0.0055	0.0083	0.031
0.02	1.36	0.073	1.23	1.54	0.023	0.0077	0.0065	0.037
0.05	1.46	0.109	1.25	1.7	0.036	0.011	0.02	0.052

TABLE I
SUMMARY OF THE SIMULATION STUDY (ONE RANDOM FLASH CROWD)

α	Overcommit Ratio				Congestion Probability			
	AVG	STD	MIN	MAX	AVG	STD	MIN	MAX
0.01	1.27	0.1	1.11	1.58	0.019	0.006	0.0077	0.031
0.02	1.31	0.08	1.17	1.44	0.026	0.0073	0.016	0.039
0.05	1.43	0.109	1.2	1.59	0.036	0.0085	0.014	0.046

TABLE II
SUMMARY OF THE SIMULATION STUDY (THREE RANDOM FLASH CROWDS)

VIII. CONCLUSIONS AND FUTURE WORK

Ability to scale cloud workloads on demand with known probability of success is an important feature, currently missing in common cloud practices. Although compute clouds often operate in over-commit mode, the risk of failing to provision a VM, associated with over-commit, are not explicitly known to the customers. We propose to make these risks explicitly known via a simple standard availability SLA extension. We propose a novel method to plan cloud capacity allowing significant additional gains in over-commit ratio, while conforming to SLAs.

We evaluated our approach through simulations based on the data from a private test and development cloud. While our methodology is general, more cloud use cases should be explored to get better insights on the proposed method advantages and disadvantages. One of the assumptions used by the proposed method is that in a sufficiently large cloud, workload demands can be treated as independent. This assumption should be further explored in other use cases.

We believe that using a single over commit strategy in all cloud use cases is disadvantageous. Choosing a specific strategy depends on the workloads characteristics, workload mix, cost of migrations, placement optimization objectives, performance guarantees, size of the cloud and a variety of other factors.

We identify three generic over-commit strategies: (a) nominal demand based, (b) actual demand based, and (c) mixed that combines the strong points of the first two.

Intuitively, if typical actual utilization levels of VMs are high, we expect to get similar over-commit ratios in both nominal and actual demand based over-commit strategies, with nominal demand strategy requiring considerably less VM migrations, smaller monitoring overhead and more robust performance. In addition, it seamlessly takes care of multiple resource dimensions by using VM type slots for placement reservations. If utilization levels for non-bottleneck resources are low, nominal demand based over-commit strategy may results in capacity waste.

In cases when utilization of VMs is typically low, the overhead of actual demand based over-commit strategy might be offset by the benefits accrued through stable consolidation of VMs on less physical hosts. It should be noted that while nominal demand over-commit strategy operates on anonymous VM slots, the actual demand based over-commit considers currently running VMs, which requires constant re-computation of the placement plans based on per VM statistics. It should be noticed that the number of VMs in the cloud can be very large and when new VMs arrive they might not have enough statistics on their utilization that additionally complicates actual demand based over-commit

A thorough comparative study of different over-commit strategies and their applicability in different cloud management contexts is one of our future research directions. Another research direction is exploring the methodology for mixed over-commit strategy in order to balance the advantages of each method. In addition, we will explore SLA diversification to strike a better balance between simplicity of a standard uniform SLA and flexibility and additional opportunities for increasing resource utilization levels with controlled risk facilitated by an SLA catalogue.

ACKNOWLEDGEMENTS

The research leading to these results has received partial funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n° 285258 (Fi-WARE). We thank our colleagues Bo Ma (IBM Tivoli, China), Ke Zhu (IBM Tivoli, China), and Andrew Trossman (IBM Tivoli, USA) for supporting and enabling this work.

REFERENCES

[1] A. Ali-Eldin, J. Tordsson, and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *NOMS*, 2012.
[2] Amazon Elastic Compute Cloud (EC2), http://aws.amazon.com/ec2.
[3] "VMware ESXi and ESX Info Center," http://www.vmware.com/products/vsphere/esxi-and-esx/index.html.
[4] "Kernel Based Virtual Machine," http://www.linux-kvm.org/page/Main_Page.
[5] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *The 7th IEEE/ACM International Conference on Autonomic Computing and Communications*, Washington, DC, USA, Jun 2010.
[6] M. Chen, H. Zhang, Y.-Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective vm sizing in virtualized data centers," in *IEEE/IFIP IM'11*, Dublin, Ireland, May 2011.
[7] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *INFOCOM*, 2011.
[8] D. Breitgand and A. Epstein, "Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds," in *INFOCOM*, 2012.
[9] C. Isci, J. E. Hanson, I. Whalley, M. Steinder, and J. O. Kephart, "Runtime demand estimation for effective dynamic resource management," in *NOMS*, 2010, pp. 381–388.
[10] A. Verma, P. Ahuja, and A. Neogi, "pmapper: power and migration cost aware application placement in virtualized systems," in *Middleware '08: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*. New York, NY, USA: Springer-Verlag New York, Inc., 2008, pp. 243–264.
[11] S. Mehta and A. Neogi, "Recon: A tool to recommend dynamic server consolidation in multi-cluster data centers," in *IEEE Network Operations and Management Symposium (NOMS 2008)*, Salvador, Bahia, Brasil, Apr 2008, pp. 363–370.
[12] J. E. Hanson, I. Whalley, M. Steinder, and J. O. Kephart, "Multi-aspect hardware management in enterprise server consolidation," in *NOMS*, 2010, pp. 543–550.
[13] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for Cloud computing," in *HotPower 08 Workshop on Power Aware computing and Systems*, San Diego, CA, USA, 2008.
[14] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *International Conference for the Computer Measurement Group (CMG)*, 2007.
[15] S. Chen, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and W. H. Sanders, "CPU gradients: Performance-aware energy conservation in multitier systems," in *Green Computing Conference*, 2010, pp. 15–29.
[16] S. A. Baset, L. Wang, and C. Tang, "Towards understanding of oversubscription in cloud," in *USENIX Hot-ICE'12*, San Jose, CA, Apr 2012.
[17] VMware Inc., "Resource Management with VMware DRS, Whitepaper," 2006.
[18] IBM, "Server Planning Tool," http://www-304.ibm.com/jct01004c/systems/support/tools/systemplanningtool/.
[19] ——, "WebSphere CloudBurst," http://www-01.ibm.com/software/webservers/cloudburst/.

[20] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
[21] B. Urgaonkar, P. J. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in a shared internet hosting platform," *ACM Trans. Internet Techn.*, vol. 9, no. 1, 2009.
[22] V. Josyula, M. Orr, and G. Page, *Cloud Computing: Automating the Virtualized Data Center*. Cisco Press, 2011, ch. Chapter 12, Cloud Capacity Management.
[23] Y. Jiang, C.-S. Perng, T. Li, and R. N. Chang, "ASAP: A Self-Adaptive Prediction System for Instant Cloud Resource Demand Provisioning," in *ICDM*, 2011, pp. 1104–1109.
[24] P. Billingsley, *Probability and Measure*, 3rd ed. John Wiley and sons, 1995.
[25] W. H. Greene, *Econometric Analysis*. Prentice Hall, 2003.
[26] C. Peng, M. Kim, Z. Zhang, and H. Lei, "VDN: Virtual Machine Image Distribution Network for Cloud Data Centers," in *INFOCOM'2012*, Orlando, FL, USA, Mar 2012.