Biting off Safely More than You Can Chew: Predictive Analytics for Resource Over-commit in IaaS Cloud

Rahul Ghosh* and Vijay K. Naik†
*Duke University, USA, †IBM T. J. Watson Research Center, USA
Email: rg51@ee.duke.edu*, vkn@us.ibm.com†

***Abstract*—Cloud service providers are constantly looking for ways to increase revenue and reduce costs either by reducing capacity requirements or by supporting more users without adding capacity. Over-commit of physical resources, without adding more capacity, is one such approach. Workloads that tend to be 'peaky' are especially attractive targets for over-commit since only occasionally such workloads use all the system resources that they are entitled to. Online identification of candidate workloads and quantification of risks are two key issues associated with over-committing resources. In this paper, to estimate the risks associated with over-commit, we describe a mechanism based on the statistical analysis of the aggregate resource usage behavior of a group of workloads. Using CPU usage data collected from an internal private Cloud, we show that our proposed approach is effective and practical.**

I. Introduction

As the technology and business models for Cloud services are getting mature, service providers (e.g., Amazon [1], Google [2], IBM [3]–[5], Microsoft [6]) are experiencing intense competitions to increase revenue and reduce costs. Providers of Infrastructure-as-a-Service Cloud (e.g., Amazon EC2 [1], IBM SmartCloud Enterprise+ [4]) deliver on-demand Virtual Machine (VM) instances to their users. These VM instances are provisioned with request specific physical resources e.g., CPU, RAM and disk capacity. In many cases, Cloud users do not have the proper insights about the capacity requirement for their workloads. As a result, the requested amount of physical resources is far beyond the actual capacity needed by the workloads during run-time. To cope up with such wasteful over-provisioning of resources by the Cloud users, service providers typically resort to two approaches: (i) for a fixed number of users and known workload characteristics, total system capacity is minimized, or, (ii) for a fixed capacity system, number of users is maximized. Over-commit of physical resources belongs to the second category, wherein, without adding more capacity, number of running workloads is maximized.

Are the real workloads over-commit friendly? To justify the case for resource over-commit, we analyzed the resource usage data collected from an internal private Cloud. Figure 1 shows the histogram of maximum CPU utilization of 2193 VMs over a period of one month. We observe that: (i) 84% of the running VMs have a peak CPU utilization less than 20% and (ii) only 14 VMs (i.e., less than 0.7% of total VMs) have a maximum CPU utilization close to 100%. Thus, a large fraction of workloads rarely use the resources that they are entitled for. Clearly, these workloads are attractive candidates for CPU over-commit.

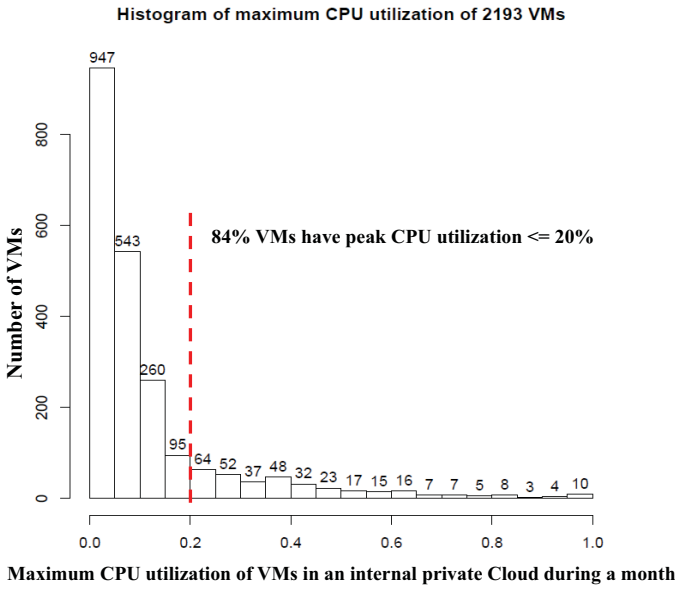

Figure 1: Majority of workloads are 'peaky' – using CPU resources only occasionally.

Risks of over-commit. On an over-committed physical machine (PM), aggregate capacity requested by the provisioned VMs exceeds the actual physical capacity of the PM. Thus, an implicit assumption behind a *safe* over-commit is that, aggregate VM resource usage at any time, does not exceed the total PM capacity. However, if the aggregate resource usage exceeds the available physical capacity of the shared resources, then there may be two impacts on the system: (i) service response time of some or all requests may significantly degrade (performance issue), and (ii) some VMs may crash and, in the worst case, the system itself may crash (availability issue). Such incidents lead to the violation of service level agreements (SLA) promised by a IaaS Cloud provider to its users, affecting the business credibility. Since it is difficult to predict the exact the resource usage of a workload mix [7], gains in revenue by any resource over-commit approach comes at a price of probabilistic risk of SLA violations. Hence, a resource over-commit approach must be coupled with a

†Corresponding author

predictive mechanism to take into account the underlying stochastic behavior of the aggregate VM resource usage.

This paper describes a statistical analysis based technique for estimating the risks associated with over-committing shared resources to an arbitrary combination of workloads. Our analysis and predictions are based on historical aggregate resource usage data, collected from an internal private Cloud. Specifically, our contributions are:

(1) We quantify the risk of demand exceeding the available shared resource capacity by a group of workloads. For a specified confidence on the resource usage data, we use the statistical measure *one-sided tolerance limit* [8] to estimate the net aggregate risk.

(2) For an arbitrary workload mix, we propose a method for analyzing and predicting the probability of aggregate resource usage exceeding a specified utilization threshold. We provide an upper bound on the probability of SLA violation because of resource over-commit. Using the data collected from an internal private Cloud, we validate the proposed approach for CPU over-commit.

(3) An online analytic method is developed for identifying workloads where resources can be over-committed. Placement of a target workload is decided by comparing the risks associated with different candidate workload groups.

Rest of the paper is organized as follows. In Section II, we define the key terms for over-commit analysis and formulate the problems of interest. Proposed one-sided tolerance limit based approach is described in Section III. Using a running example of group of candidate workloads, we validate the proposed approach in Section IV. Use case scenario for the developed analytics for the placement of a target group of workloads is presented Section V. Related research is highlighted in Section VI. Finally, we conclude this work and outline the future avenues of research in Section VII.

II. Definitions and Problem Formulation

Key Terms. We define the following terms in the context of over-commit analysis.

(1) Virtual CPU (vCPU). VMs are provisioned with the requested number of virtual CPUs (vCPUs) along with RAM and disk capacity. A vCPU refers to a *virtual processor* as seen by a VM [9]. Such virtual processor [10] is a representation of a physical processor to the OS of a logical partition that uses shared processors. In our analysis, a vCPU is roughly half of a thread of a hyper-threaded real physical CPU. Thus, if the total number of physical CPUs on a PM is C_{total}, maximum number of vCPUs (vC_{total}) that can be supported without over-commit is $2C_{total}$.

(2) CPU utilization of a VM (U_{vm}). We define the CPU utilization of a VM (U_{vm}) for a time interval T as:

$$U_{vm} = \frac{CPU\_time_{_T}}{T \cdot vC} \tag{1}$$

where, CPU_time_T = Total CPU time spent by the VM across all the vCPUs during the time interval T, and vC = Total number of vCPUs allocated to a VM.

(3) Aggregate CPU utilization for a group of VMs (U_{group}). For a group of $n_{_v}$ VMs, we define aggregate CPU utilization (U_{group}) during time interval T as :

$$U_{group} = \frac{\sum_{i=1}^{n_{_v}} CPU\_time_{_T}^{(i)}}{T \cdot \min\{\sum_{i=1}^{n_{_v}} vC_{_i}, vC_{total}\}} \tag{2}$$

where, $CPU\_time_{_T}^{(i)}$ = total CPU time spent by the i-th VM across all the vCPUs during the time interval T, $vC_{_i}$ = number of vCPUs allocated to the i-th VM, vC_{total} = maximum number of vCPUs that can be supported on a PM without over-commit.

(4) Sample and observation. For each VM or for a group of VMs, utilization value at a specific time instance is called a sample. An observation consists of a set of samples collected over a time interval. Figure 2 shows a conceptual view of samples and observations.

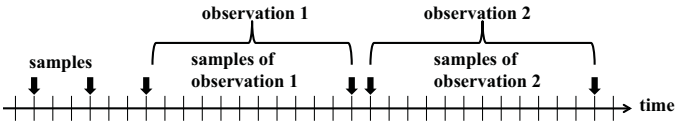

Figure 2: Conceptual view of samples and observations.

(5) Infrastructure SLA and its violation. In IaaS Cloud, when users sign up for a Cloud based service, they can provision one or more VM instances with specific CPU, RAM and disk capacity. Infrastructure SLA is a way to assure IaaS Cloud users that they are entitled for a guaranteed service for which they are charged. During the service period, an SLA violation occurs if the user is unable to receive any of the promised computing resources (CPU, RAM or disk) provisioned with her VM.

Problem statement. In an over-committed IaaS Cloud, since the total VM capacity requested by the Cloud users is more than the PM capacity on which the VMs are running, there is a risk of SLA violation. To minimize the risk of such SLA violations, a threshold utilization is set and the Cloud management is alarmed if the aggregate utilization of

all the VMs running on the same PM exceeds the threshold utilization. In this paper, we address the following two problems to mitigate the risks of over-commit: (1) given a group of VMs, with known workload characteristics, predict the probability that aggregate resource utilization by the group of VMs will exceed a pre-defined set threshold over a time horizon, and (2) quantify and estimate the risk associated with provisioning a certain PM capacity to a group of VMs. In subsequent Sections, we describe our proposed approach to solve these problems and validate our approach against the data collected from an internal private Cloud.

III. Proposed Approach

A. Motivation.

We first identify a few key characteristics required for a predictive resource over-commit analysis.

(1) Notion of safety margin. To estimate the risk of SLA violation on an over-committed PM, any random utilization sample needs to be analyzed rather than the average utilization behavior. This is because, violation of SLA, even once, can be very critical to a service provider. To capture these sudden transient peaks, we introduce the notion of perceived *safety margin*. Such a safety margin is the characteristic of workload mix and contains a fraction of samples below a utilization level. From a given observation, risk of running a group of VMs on the same PM can be estimated by comparing the gap between the safety margin and the set threshold. In Figure 3, the safety margin R_1 is

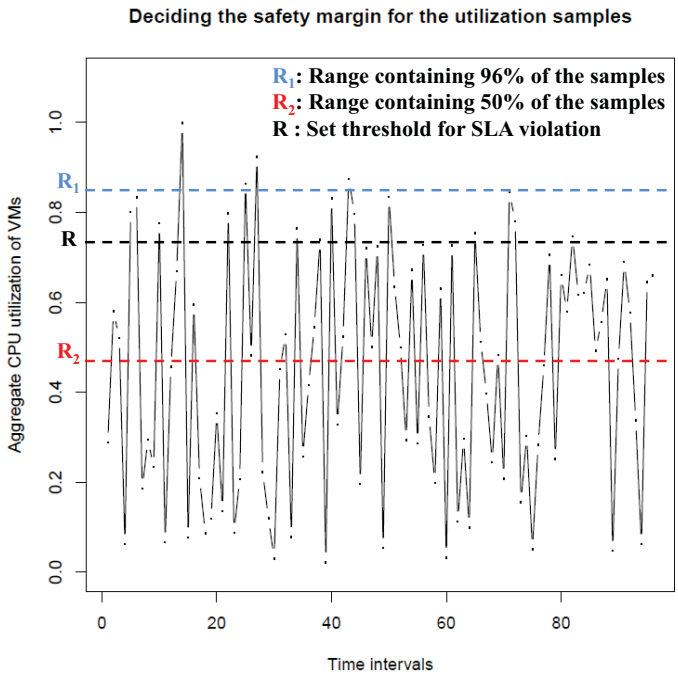

Figure 3: Safety margin to estimate the risk of over-commit.

above the pre-defined set threshold R and contains 96% of the samples, while R_2 is below R but contains only 50% of the samples. Clearly, setting a meaningful safety margin is of interest to estimate the risks of over-commit.

(2) Confidence on the data. For online analysis from a given observation, we need to take into account the uncertainty associated with an observation. This is because, the actual (or future) utilization behavior is unknown and it varies across different observations. Hence, a *confidence* is necessary on the predicted safety margin to capture the uncertainty in the analysis based on the observation.

Given the two characteristics of the required analytics, we observe that *tolerance interval* [8], a measure of statistical quality control, is an effective tool to develop a predictive analytic method in estimating over-commit risk.

B. Tolerance interval for resource over-commit analysis.

To meet quality control, engineering or manufacturing products are sometimes required to satisfy certain tolerance specifications. In many practical cases, to save time and cost, a suitable *tolerance interval* is constructed based on a small number of product samples, to assess the fraction of the entire batch of products that is within the specifications. We explain this concept of tolerance interval with an example. Assume that a large number of items submitted for quality inspection will be accepted if *at least* 95% of the items are within the specification interval (L, U), where, L and U are the lower and upper specification limits respectively. To evaluate the quality of the items, only a small sample items are inspected and a tolerance interval is computed with a given confidence, such that at least 95% of all items are within the tolerance interval. If the computed tolerance interval falls in the interval (L, U), then it can be concluded that at least 95% of the items can be accepted with a given confidence. In this example, if the items are required to satisfy only the upper specification limit, a *one-sided upper tolerance limit* is computed to include at least 95% of the items with a given confidence and then compared with the value of U. Notice, the similarity between: (i) constructing a one-sided upper tolerance limit for the product samples and (ii) determining a safety margin for the utilization samples of a group of VMs, over-committed on a PM. By comparing the externally set threshold (i.e., upper specification limit) on the aggregate VM utilization with the safety margin (i.e., one-sided upper tolerance limit), risk of over-commit can be quantified.

Analytic expressions. Let $X_1, ..., X_n$ be a set of utilization samples during an observation, drawn from a *normal* population with *mean* μ and *variance* σ^2. Let $\overline{X}$ and S denote the sample mean and sample standard deviation respectively. Given a number of samples n, a coverage factor

β ($0 < \beta < 1$) and a confidence γ ($0 < \gamma < 1$), the one-sided upper tolerance limit is given by [8]:

$$U(\beta, \gamma) = \overline{X} + cS \tag{3}$$

where, the tolerance factor c needs to be determined so that:

$$P\Big[P[X \leq \overline{X} + cS|\overline{X}, S] \geq \beta\Big] = \gamma \tag{4}$$

Mathematically, c can be computed as [8]:

$$c = \frac{t_{n-1,\gamma}(z_\beta \sqrt{n})}{\sqrt{n}} \tag{5}$$

where, z_β denotes the 100βth percentile of the standard normal distribution, $t_{n-1,\gamma}(z_\beta \sqrt{n})$ denotes the 100γth percentile of a non-central t distribution with $(n-1)$degrees of freedom and non-centrality parameter $z_\beta \sqrt{n}$. Thus, at least $100\beta\%$ of the samples from the normal population are less than or equal to the one-sided upper tolerance limit $U(\beta, \gamma)$ with confidence γ. Observe, the coverage factor β determines the fraction of samples that can be contained within the computed tolerance limit, whereas, the confidence γ captures the uncertainty associated with a given observation. Next, we describe how one-sided tolerance limit can be used to answer the two problems mentioned in Section II.

C. Computing the bound on SLA violations.

Given a group of VMs, we predict the probability that the aggregate resource utilization will exceed a pre-defined set threshold R over a time horizon. The prediction steps are:

(1) Create a profile of n time samples (estimated or measured) of the aggregate CPU utilization for the group of VMs. This profile is used as an observation in our analysis.

(2) Compute sample mean $\overline{X}$ and sample standard deviation S from the profile created in step (1).

(3) Set the threshold to the one-sided upper tolerance limit:

$$\overline{X} + cS = R \tag{6}$$

Rearranging, we obtain the value of c as:

$$c = \frac{R - \overline{X}}{S} \tag{7}$$

(4) Comparing Equations (5) and (7), we can write:

$$\frac{R - \overline{X}}{S} = \frac{t_{n-1,\gamma}(z_\beta \sqrt{n})}{\sqrt{n}} \tag{8}$$

For a given value γ (i.e., confidence on the observation), we solve the Equation (8) for the value of β.

(5) The value of $(1-\beta)$ provides *upper bound* on the probability of exceeding the set threshold R.

Why $(1-\beta)$ is an upper bound on threshold exceeding probability? Using Equation (6), we can simplify the Equation (4) as:

$$P\Big[P[X \leq R|\overline{X}, S] \geq \beta\Big] = \gamma \tag{9}$$

Let us define A as:

$$A = P[X \leq R|\overline{X}, S] \tag{10}$$

Thus, Equation (9) can be further simplified as:

$$P[A \geq \beta] = \gamma \tag{11}$$

Rearranging, we can write:

$$P[(1 - A) \leq (1 - \beta)] = \gamma \tag{12}$$

Equation (10) can be interpreted as follows: for a given observation (characterized by $\overline{X}, S$), A is the probability that any unknown utilization (denoted by the random variable X) will be less than R. Hence, from the Equation (11), we can say: with confidence γ, lower bound on the value of A is β. Conversely, for a given observation, $(1-A)$ is the probability that any unknown utilization will be exceeding R i.e., our measure of interest. Thus, upper bound on the value of $(1-A)$ is given by $(1-\beta)$, with confidence γ.

D. Quantifying the risk of over-commit.

Given a group of VMs, we quantify the risk of provisioning a certain PM capacity. The steps are described below:

(1) For a given coverage factor (β) and confidence (γ), compute the one-sided upper tolerance limit $U(\beta, \gamma)$ for the group of VMs. $U(\beta, \gamma)$ is a predicted upper bound on the utilization behavior for a percentage of samples with a given confidence.

(2) For a given coverage factor (β), confidence (γ) and a set utilization threshold R, we quantify the risk ($M(\beta, \gamma)$) of running a group of VMs as:

$$M(\beta, \gamma) = \frac{U(\beta, \gamma) - R}{R} \tag{13}$$

Thus, we define risk as the relative difference between the one-sided upper tolerance limit and the set threshold. Observe, in Equation (13), $M(\beta, \gamma)$ can be positive or negative. Higher the estimated value of $M(\beta, \gamma)$, higher is the risk associated with over-commit. When $M(\beta, \gamma)$ is negative, it quantifies the over-commit opportunities that may exist. We discuss in Section V, how this property can be leveraged for the placement of a new request.

IV. Results, Validation and Discussions

In this Section, we show how the proposed approach can be applied for the analysis of CPU over-commit for a group of VMs running on a PM. We collected the anonymized

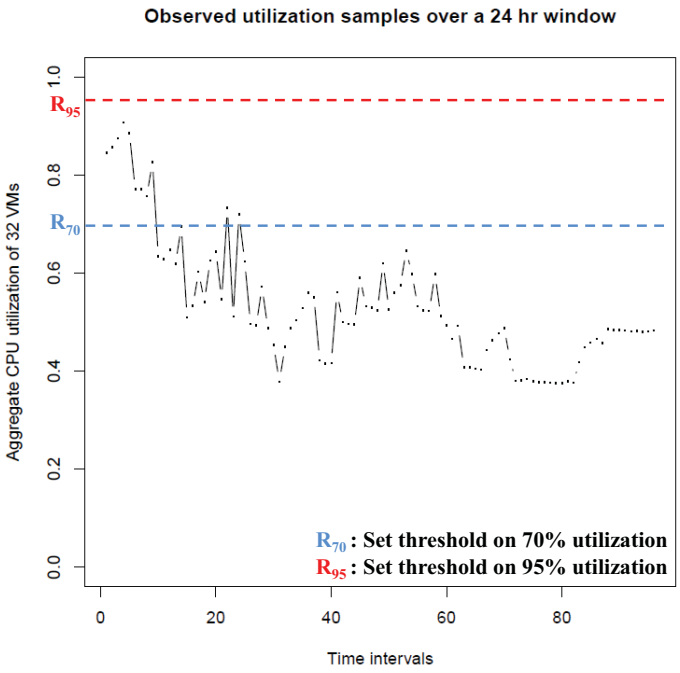

Figure 4: Monitored utilization samples of aggregated CPU utilization over 24 hr window.

CPU usage data from an internal private Cloud for large number of provisioned VMs. The average CPU utilization of a VM was measured for a 15 minute time interval from the hypervisor. Thus the collected utilization samples are 15 minutes apart. From the collected data, we randomly created large (around $100,000$) groups of $8, 16, 32$ and 64 VMs. However, due to space limitation, we present the results using a random example of group of 32 VMs and then, summarize how the analysis can be applied for large collection of VM groups.

Proposed approach using a running example. For the sample of group 32 VMs presented in this analysis, total number of vCPUs requested is 99. We assume that these VMs are to be provisioned on a PM with 16 physical CPU cores that can support a maximum of 32 vCPUs. Thus, the vCPU capacity is over-committed by a factor of $99/32$ or 3.09. For this group of VMs, we predict the probability of exceeding a set threshold utilization and estimate the risk over a 24 hr time window. Specifically, the analysis is presented for two values of threshold utilization: 70% and 95%. Since the collected data contains utilization samples at every 15 minutes interval, over a 24 hr window, total 96 samples are recorded. Figure 4 shows the aggregate CPU utilization in each time interval and the set thresholds at 70% and 95% utilization. Observe, for the given group of VMs, 70% threshold is exceeded by a small samples although none exceeds the 95% threshold.

From the observed data, we compute the sample mean ($\overline{X}$), sample standard deviation (S) and the predicted upper bound ($1-\beta$) on threshold exceeding probability for different confidence values (γ). Table I shows the predicted

Table I: Predicted bounds with 70% threshold R_{70}.

Confidence (γ)	**Lower bound** (β) **on the probability of being within the threshold** R_{70}	**Upper bound** ($1-\beta$) **on the probability of exceeding the threshold** R_{70}
90%	87.01%	12.99%
95%	85.92%	14.08%
99%	83.73%	16.27%

upper bounds of exceeding the 70% utilization threshold for confidence values of $90\%, 95\%$ and 99%. To validate the predicted results shown in Table I, we compute the empirical probability of exceeding the threshold from the observed data. As shown in Figure 4, out of 96 utilization samples, 11 samples exceed the 70% utilization threshold R_{70}. Hence, empirical probability of exceeding the threshold is $11/96$ or 11.46%. We then compare this empirical probability with the predicted upper bounds (i.e., $12.99\%, 14.08\%, 16.27\%$) in Table I, computed for different confidence values. The proposed approach is effective since the predicted upper bounds are higher than the value of empirical probability. Table II shows the predicted upper bounds of exceeding the

Table II: Predicted bounds with 95% threshold R_{95}.

Confidence (γ)	**Lower bound** (β) **on the probability of being within the threshold** R_{95}	**Upper bound** ($1-\beta$) **on the probability of exceeding the threshold** R_{95}
90%	99.83%	0.17%
95%	99.77%	0.23%
99%	99.61%	0.39%

95% utilization threshold for confidence values of $90\%, 95\%$ and 99%. For the group of VMs, as shown in Figure 4, out of 96 utilization samples, no samples exceed the 95% utilization threshold R_{95}. Hence, empirical probability of exceeding the threshold is zero. Observe, the predicted upper bounds (i.e., $0.17\%, 0.23\%$ and 0.39%) in this case are smaller than the upper bounds computed in Table I. Since no sample goes beyond the set threshold for the observed window, computed upper bounds are close to zero.

In Table I and II, we observe that the predicted upper bounds increase when the confidence on the observed data is increased. This can be explained mathematically. Equation (11) shown in Section III, can be re-written as:

$$P\Big[A \in [\beta, 1]\Big] = \gamma \tag{14}$$

Table III: Estimated risks for the group of 32 VMs.

$(100\gamma, 100\beta)\%$	**One-sided upper tolerance limit**	**Risk for 70% threshold**	**Risk for 95% threshold**
$(90, 90)$	72.13%	3.04%	-99.24%
$(90, 95)$	77.15%	10.22%	-99.19%
$(90, 99)$	86.65%	23.79%	-99.09%
$(95, 90)$	72.87%	4.10%	-99.23%
$(95, 95)$	78.01%	11.44%	-99.18%
$(95, 99)$	87.74%	25.35%	-99.08%
$(99, 90)$	74.35%	6.21%	-99.22%
$(99, 95)$	79.72%	13.88%	-99.16%
$(99, 99)$	89.92%	28.46%	-99.05%

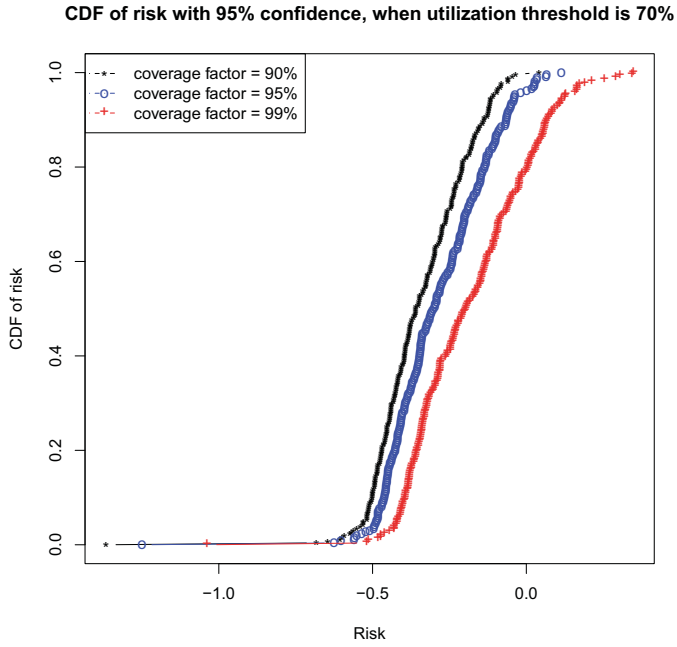

Figure 5: CDF of risk for a large collection of VM groups.

This is because the quantity A is a probability as shown in Equation (10). If the value of confidence γ increases, the probability on the left-hand side of the Equation (14) can increase only if the length of the interval $[\beta, 1]$ increases. Further, A is a probability and hence, the interval will be larger only if the value of β is reduced or the value of $(1 - \beta)$ i.e., the upper bound is increased.

Using the Equation (13), we estimate the risks of running the group of 32 VMs for the specified 70% and 95% utilization thresholds. Table III shows the values of risks for different combinations of coverage factor and confidence values. For the given group of VMs, we notice from the observed data that no sample goes beyond the 95% threshold. Hence, the estimated risk is negative for 95% threshold. Such negative values also suggest that the given group of VMs is *more friendly* for CPU over-commit with 95% threshold compared to 70% threshold. From Table III, we also observe that the risk increases if the value of coverage factor (or confidence) is increased keeping the confidence (or coverage factor) constant. This is because, one-sided upper tolerance limit increases if the value of coverage factor or confidence is increased.

Proposed approach on a collection of groups. In a data center hosting IaaS Cloud, there can be thousands of PMs, each of which running a group of VMs. While the previous results show how the developed analytics can be applied for over-commitment on a given PM, it is also important to understand the aggregate risk of over-commit in a data center. We describe how the proposed approach can be used to perform an aggregate analysis on a large collection of VMs. Figure 5 shows the CDF of risk over a large collection

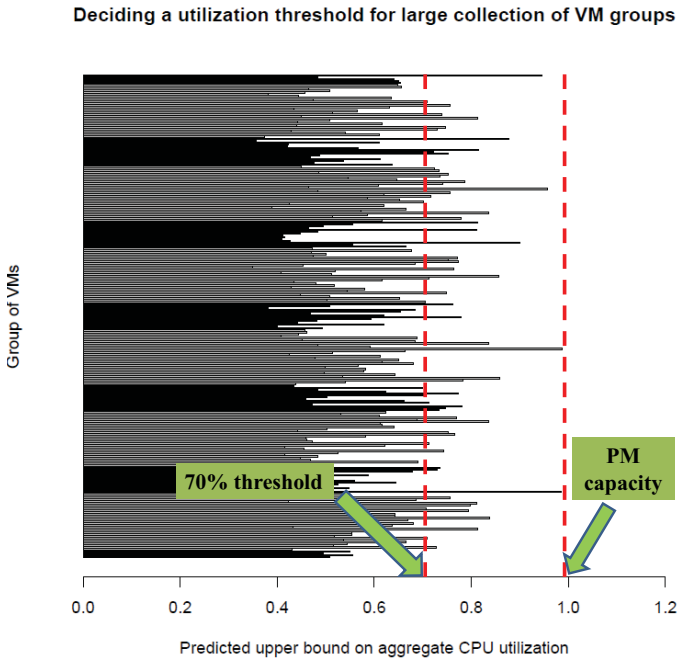

Figure 6: Deciding a meaningful utilization threshold for over-commit analysis.

of VM groups when the utilization threshold is set at 70%. For a given confidence, we observe that CDF of risk is steeper for a larger value of coverage factor. While the risk is computed by comparing the one-sided upper tolerance limit with the set threshold, it is also important to set the threshold meaningfully. If the set threshold is too low (or high) compared to the one-sided upper tolerance limit, the predicted risk of over-commit will be too high (or low). To set a meaningful threshold on aggregate utilization, we

leverage the one-sided upper tolerance limit [11], which provides an upper bound on aggregate utilization. In Figure 6, we show the predicted upper bound on aggregate CPU utilization for a large collection of VM groups with 99% coverage factor and 99% confidence. Each horizontal bar in the Figure 6 represents a group of 32 VMs. We also show the 70% and 100% (PM capacity) utilization thresholds. When the threshold is set at 70%, horizontal bars (or the corresponding VM groups) which exceed the set threshold are at a higher risk of SLA violation due to over-commit. Hence, to reduce the aggregate risk of over-commit in data center, threshold should be set to a value so that the predicted upper bound of aggregate utilization exceeds the thresholds only for a small fraction of VM groups.

V. Use Case for the Analytics

Using the proposed approach, the service provider can set an *acceptable risk* for over-commit, with a goal to minimize the SLA violations and to maximize the revenue. Developed analytics can then be used for the placement of a target workload by comparing the acceptable risk with the risk of running the target workload together with a group of VMs. Algorithm 1 describes the identification of the VM group from a set of candidate groups, for the placement of a target workload. Input parameters to this algorithm are: (i) an ordered list ($candidate\_list[.]$) of candidate VM groups, (ii) the target workload (W), (iii) acceptable risk for over-commit (M_a), (iv) set utilization threshold (R), (v) coverage factor (β), (vi) confidence value (γ), and (vii) database ($util\_db[.]$) of historical resource utilization by workloads. A service provider can populate $candidate\_list[.]$ by ordering the VM groups based on several factors e.g., upper bound on aggregate utilization, risk behavior and so on. In this paper, we do not focus on the details of those issues as they will be addressed elsewhere. The output of the algorithm is G_{select}, which denotes the selected group of VMs for placement of the target workload. Initially, G_{select} is set to NULL. Algorithm 1 iterates over the $candidate\_list[.]$ to check if a potential group of VMs $G_{potential}$ in the list, can be selected as G_{select}. In each iteration, a profile (F) of observed aggregate utilization by $G_{potential}$ and target workload W, is created by calling the function $createProfile(.)$. For a given coverage factor β, given confidence γ and set threshold R, the profile F is then used by the function $computeUpperTolerance(.)$ to compute one-sided upper tolerance limit $U(\beta,\gamma)$. Function $estimateRisk(.)$ computes the estimated risk ($M_{(\beta,\gamma)}$) of running the target workload W, together with the VM group $G_{potential}$. Finally, the iteration stops if: (i) the estimated risk of a potential VM group is less than or equal to the acceptable risk or, (ii) $candidate\_list[.]$ is exhausted. Observe, when none of the candidate VM groups is suitable for over-commit with the target workload, G_{select} is NULL.

Algorithm 1 Identification of the VM group from a set of candidate groups, for the placement of a target workload.

Input: (i) $candidate\_list[.]$: candidate VM groups for placement, (ii) W: target workload, (iii) M_a: acceptable risk, (iv) R: set threshold, (v) β: coverage factor, (vi) γ confidence value and (vii) $util\_db[.]$ database of historical resource utilization by workloads.
Output: selected group of VMs where the target workload can be placed.

declare G_{select}: *VM group*;
declare $G_{potential}$: *VM group*;
declare F: *utilization profile*;
declare $U_{(\beta,\gamma)}$: *one-sided upper tolerance limit*;
declare $M_{(\beta,\gamma)}$: *risk*;
declare *stop*: *boolean*;
declare i: *integer*;
$G_{select} \leftarrow$ NULL;
$i \leftarrow 0$;
stop $\leftarrow$ false;
do:
  $G_{potential} \leftarrow candidate\_list[i]$;
  $F \leftarrow createProfile\Big(W, G_{potential}, util\_db[.]\Big)$;
  $U_{(\beta,\gamma)} \leftarrow computeUpperTolerance(F, \beta, \gamma)$;
  $M_{(\beta,\gamma)} \leftarrow estimateRisk(U_{(\beta,\gamma)}, R)$;
  if $M_{(\beta,\gamma)} \leq M_a$:
    $G_{select} \leftarrow G_{potential}$;
    stop $\leftarrow$ true;
  else:
    $i \leftarrow i + 1$;
while (*stop* == false) or ($i < len(candidate\_list[.])$)
output G_{select};

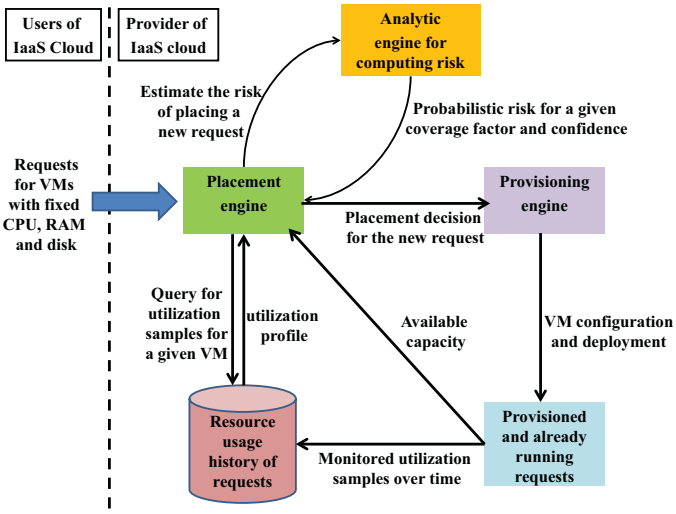

Figure 7: Workflow describing the high level interactions among different components for placement and provisioning in an over-committed IaaS Cloud.

Figure 7 describes the interactions among different components for placement and provisioning in an over-committed IaaS Cloud. The predictive analytics developed in this paper can be used to build an *analytic engine*. In

response to a Cloud user's VM provisioning request, *placement engine* together with the analytic engine will check the feasibility of over-committing a PM to provision the request. Once the request is provisioned by the *provisioning engine*, Cloud should be instrumented to monitor and collect the utilization samples of the provisioned VMs. These historical resource usage behaviors of the VMs are used to generate utilization profiles during the placement decision of a new request.

VI. Related Research

In [12], Gordon *et al.* presented a system *Ginkgo* to handle memory over-commit while minimizing SLA violations. In [13], Williams *et al.* proposed a system *Overdriver* when run-time memory allocated to the VMs is insufficient (overload) due to a over-commit. Analytics developed in this paper can complement the experimental systems described in [12] and [13]. In [14], Tsakalozo *et al.* described a two step approach to solve the VM placement problem in an over-committed IaaS Cloud. However, their proposed approach considered only the average CPU utilization and thus failed to capture the variability in the resource utilization behavior. In another work, [15], Nathuji *et al.* developed *Q-Clouds* to create a MIMO based online feedback model for managing CPU resources while reducing the performance interference effects due to a shared environment. There are few research efforts that developed statistical approaches for over-commit analysis. For video-on-demand servers, Vin *et al.* [16] proposed a statistical admission control technique that over-commit resources. Most closely related to our work is by Urgaonkar *et al.* [17], where the authors demonstrated the feasibility and benefits of over-commit while provisioning CPU and network resources in a Linux cluster. Authors of [17] used a probabilistic model for resource prediction which is different from the one-sided tolerance limit approach described in this paper. The approach proposed in [17] failed to capture the uncertainty associated with an observation, which is taken into account by confidence (γ) in our one-sided tolerance limit approach. While the authors in [17] evaluated their approach using synthetic benchmarks (e.g., SPECweb99) on Linux cluster, we show the effectiveness of our approach through real resource usage behavior of VMs from an internal private Cloud. Finally, in [17], there is no discussion on risk estimation for running VMs on an over-committed PM, which is addressed in our paper.

VII. Conclusions and Future Research

This paper presents a statistical approach to quantify and estimate the risk of over-committing a group of VMs beyond the PM capacity. Using one-sided tolerance limit approach, we predict the upper bound on the probability of exceeding a set utilization threshold while running a group of VMs. We describe how the developed analytics can be an integral part of the system during placement of a new request. More validation w.r.t. training and test data will be done in future. Although, this paper presents results for CPU over-commit analysis, the same statistical approach is applicable to other types of resources, e.g., memory, disk and network. In future, we plan to extend this work to simultaneous over-commit of multiple types of resources. We will predict and analyze degradations in system performance and availability using a combination of statistical, queuing theoretic and Markov models [18]. We will also investigate the solution approach when the utilization samples follow non-normal distributions.

Acknowledgments. The authors would like to thank Murthy Devarakonda (IBM Research) for introducing the problem and Norbert Vogl (IBM Research) for providing help during data collection from Cloud. We also thank Avishek Chakraborty (Texas A&M University) for insightful discussions on the concept of tolerance interval. This work was completed during Rahul's internship at IBM Research.

References

[1] "Amazon EC2," 2006, http://aws.amazon.com/ec2.
[2] "Google Apps for Business," 2011, http://lp.google-mkto.com/NorthAmAppsMM1CloudKit_NorthAm-CloudKit.html.
[3] "IBM SmartCloud Enterprise," 2011, http://www-935.ibm.com/services/us/en/cloud-enterprise/index.html.
[4] "IBM cloud computing," 2012, from Wikipedia: http://en.wikipedia.org/wiki/IBM_cloud_computing.
[5] "IBM Smart Business Desktop Cloud," 2011, http://www-07.ibm.com/businesscenter/au/services/cloud_computing/smart_business_desktop_cloud.html.
[6] "Microsoft Cloud," 2010, http://www.microsoft.com/en-us/cloud/default.aspx.
[7] R. Buyya, J. Broberg, and A. Goscinski (Eds.), *Cloud Computing Principles and Paradigms*. Wiley, 2011.
[8] K. Krishnamoorthy, *Handbook of Statistical Distributions with Applications*. Chapman and Hall/CRC, 2006.
[9] VMware vSphere 4.1, "VMware vSphere: The CPU Scheduler in VMware ESX 4.1," *VMware white paper*, 2010.
[10] "Virtual processors," 2009, http://publib.boulder.ibm.com/infocenter/powersys/v3r1m5/index.jsp?topic=/iphat/iphatvirtualproc.htm.
[11] T. Chandrupatla, *Quality and Reliability in Engineering*. Cambridge University Press, 2009.
[12] A. Gordon, M. Hines, D. da Silva, M. Ben-Yehuda, M. Silva, and G. Lizarraga, "Ginkgo: Automated, application-driven memory over-commitment for cloud computing," in *ASPLOS RESoLVE workshop*, 2011.
[13] D. Williams, H. Jamjoom, Y. Liu, and H. Weatherspoon, "Overdriver: Handling memory overload in an oversubscribed cloud," in *VEE*, 2011.
[14] K. Tsakalozos, M. Roussopoulos, and A. Delis, "VM placement in non-homogeneous iaas-clouds," in *ICSOC*, 2011.
[15] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: managing performance interference effects for QoS-aware clouds," in *EuroSys*, 2010.
[16] H. M. Vin, P. Goyal, A. Goyal, and A. Goyal, "A statistical admission control algorithm for multimedia servers," in *ACM Multimedia*, 1994.
[17] B. Urgaonkar, P. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in shared hosting platforms," in *OSDI*, 2002.
[18] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. Wiley, 2001.