H-0312 (H1201-006) January 10, 2012
Computer Science

IBM Research Report

Improving Consolidation of Virtual Machines with Risk-aware Bandwidth Oversubscription in Compute Clouds

David Breitgand, Amir Epstein
Haifa Research Laboratory
Mt. Carmel 31905
Haifa, Israel

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Improving Consolidation of Virtual Machines with Risk-aware Bandwidth Oversubscription in Compute Clouds

David Breitgand Amir Epstein
Virtualization Technologies, System Technologies & Services
IBM Haifa Research Lab, Haifa, Israel
Email: {davidbr, amire}@il.ibm.com

***Abstract*—Current trends in virtualization, green computing and Cloud computing require ever increasing efficiency in consolidating virtual machines without degrading quality of service. In this work, we consider consolidating virtual machines on the minimum number of physical containers (e.g., hosts or racks) where the physical network (e.g., network interface or top of the rack switch link) may become a bottleneck. Since virtual machines do not require maximum of their nominal bandwidth simultaneously, capacity of the physical container can be multiplexed. We assume that each virtual machine has a probabilistic guarantee – derived from its Service Level Agreement with the Cloud provider – on realizing its bandwidth requirements. Therefore, the problem of consolidating virtual machines on the minimum number of physical containers, while preserving these bandwidth allocation guarantees, can be modeled as *Stochastic Bin Packing (SBP)* problem, where each virtual machine's bandwidth demand is treated as a random variable.**

We consider both off-line and on-line versions of SBP. Under the assumption that virtual machines' bandwidth consumption obeys Normal distribution, we show a 2-approximation algorithm for the off-line version and improve the previously reported results by presenting $(2+\epsilon)$-competitive algorithm for the on-line one. We also observe that a dual PTAS for SBP can be obtained via reduction to the 2-dimensional vector bin packing problem.

Finally, we perform a thorough performance evaluation study using both synthetic and real data to evaluate the behavior of our proposed algorithms showing their practical applicability.

I. Introduction

Modern virtualization technology allows a broad spectrum of resource allocation optimizations, which were not available in a traditional data centers. The virtualized workloads comprised of Virtual Machines (VMs) can be migrated across different physical hosts spanning disparate racks and even data centers. One obvious usage of this new flexibility is VM consolidation on the minimum number of physical hosts/racks to reduce capital expenditures and save operational costs such as power and maintenance.

When VMs are consolidated on the same physical host or rack, they share the same physical network device such as network interface or top-of-the-rack switch port, which may become a bottleneck. In fact, a number of recent studies indicate that network bandwidth may constrain VM performance in Cloud environments due to network over-subscription [1]–[3]. As Cloud paradigm attracts an ever increasing number of customers, and network traffic in and out of the Cloud proliferates, sharing of the network bandwidth without hurting VMs quality of service gains in importance. While over-subscription of other physical resource types such as CPU and memory has been studied intensively for the last few years [4]–[6], bandwidth over-subscription in virtualized Cloud-like environments attracted less attention thus far [7].

In this work, we focus on placement algorithms that allow to consolidate VMs with bandwidth demand that varies over time on a minimum number of hosts/racks, such that the physical bandwidth capacity shared by the VMs placed together on the same container will be sufficient to satisfy VMs bandwidth demand with a given probability, thus controlling the risk of bandwidth congestion on the shared physical network device.

To satisfy demand of all the consolidated VMs, the total simultaneous bandwidth consumption of these workloads should not exceed the host's network interface or top-of-the-rack's switch port capacity, which is shared by the consolidated VMs. One way to ensure this is to place VMs on the same host or rack, such that the sum of maximum bandwidth demands (historically observed or predicted) will not exceed the bandwidth of the network device shared by the VMs. Obviously, this over-provisioning technique would not leverage advantages of the Cloud computing approach. It will result in resource waste at the provider side and higher costs to the customers.

Fortunately, thanks to statistical multiplexing, the actual aggregate bandwidth consumption of the consolidated VMs might be much smaller than the available physical network capacity, the latter can be multiplexed among VMs, removing the need for costly over-provisioning. Intuitively, if dynamic bandwidth consumption of VMs is regarded as a random variable and multiple such random variables are "packed" together in the same bin, the standard deviation of the random variable that represents the sum of dynamic bandwidth demands is smaller than the sum of standard deviations of the individual variables. This phenomenon is known as *smoothing*.

Cost-efficiency of a Cloud provider depends on its ability to over-subscribe capacity by leveraging the smoothing effect without degrading the quality of service. The internal policy of the provider for bandwidth over-subscription may be expressed, for example, by a set of values $p_{copper}, p_{bronze}, ..., p_{platinum}$ specifying that bandwidth demand (either estimated from historical data or predicted) of a VM in a corresponding SLA class will be satisfied with probability at least $1 - p_{sla}$.

This problem can be formulated as a Stochastic Bin Packing problem (SBP) where the items are random variables rep-

resenting dynamic bandwidth consumption of VMs [8], [9]. The problem is to consolidate VMs belonging to the same SLA class on the minimum number of bins (hosts or racks) such that the probability of satisfying VM bandwidth demand is at least $1 - p_{sla}$. For simplicity in what follows we will consider a single SLA class and refer to p_{sla}, probability of resource congestion, called target *overflow probability*, per physical container, simply as p.

A. Our Contribution and Paper Organization

Recently, [7] provided an online algorithm with a competitive ratio of $(1+\epsilon)(1+\sqrt{2})$ for any $\epsilon > 0$, for SBP where the items follow Normal distribution. We improve this result by providing an online algorithm with competitive ratio $(2 + \epsilon)$ for any $\epsilon > 0$.

We make use of the fact that the statistical multiplexing effect is more pronounced in bins where items with higher variability of bandwidth demand are packed together. We prove that a greedy algorithm that packs items in a non-increasing order of their variance to mean ratio (VMR) is a 2-approximation algorithm for SBP. The proof is based on the fact that this greedy algorithm obtains an optimal fractional solution to a relaxation of the natural mathematical program for SBP. To obtain our improved online algorithm, we partition the items into classes according to their VMR and solve SBP separately for each class using a greedy packing algorithm.

We also show how to obtain a dual PTAS for SBP by reduction to the 2-dimensional vector bin packing problem [10]. In the dual PTAS the number of bins produced by the algorithm is optimal, but capacity constraints of each bin is relaxed by a factor of $1 + \epsilon$, for $\epsilon > 0$.

We perform a thorough performance evaluation study of our algorithms using synthetic data and the data obtained from a production data center environment.

In our experiments, the proposed algorithms consistently outperformed the known alternatives on the synthetic and real data sets in terms of the number of bins used for packing. Since the real data trace deviates from the Normal distribution assumption, we discovered that the actual overflow probabilities achieved by our algorithms were larger than the target ones. However, as we show, the achieved overflow probability violation rates are sufficiently small to allow the practical application of the proposed algorithms.

The rest of this paper is organized as follows. Section II discusses background and related work. Section III defines the model. Section IV presents our algorithms and provides proofs of their worst case performance guarantees. Section V presents our simulation study of the proposed algorithms performance. Finally, Section VI provides some conclusions and future work.

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 [11]–[16] and motivate features 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 the number of hosts, secondary goals such as migration minimization, performance optimization and other are considered. Variants of the classic packing algorithms such as First Fit (FF) and First Fit Decreasing (FFD) [20] are often used to achieve practical solutions.

Bin packing is one of the basic problems in theoretical computer science and combinatorial optimization and has many real-world applications. In the classical bin packing problem we are given a finite list of elements, called items where each item has size in $(0, 1]$. The goal is to pack the items into a minimum number of unit-capacity bins such that the total size of the items in each bin does not exceed the bin capacity. The problem is known to be NP-hard [20], thus research has focused on the study and design of approximation algorithms, which find near optimal solutions with proven performance guarantees efficiently. A particular class of algorithms is on-line algorithms. An on-line algorithm receives the input incrementally, one piece at a time and the algorithm must generate output for each input part without knowing future input. This is in contrast to traditional off-line algorithms that have a complete knowledge of the entire input. In particular, an online algorithm for bin packing is given one item from the list of items at a time and must assign each item to a bin immediately upon arrival.

Various approximation algorithms are known for the problem. The simple First Fit Decreasing (FFD) algorithm, which sorts the items in non-increasing order of size and applies the first fit (FF) packing rule (each item is placed in the earliest bin into which it fits) has asymptotic approximation ratio of $\frac{11}{9}$ (see e.g., [21], [22]). These results imply that the absolute approximation ratio of FFD is $\frac{3}{2}$. This bound is tight since no algorithm can have absolute approximation ratio less than $\frac{3}{2}$ unless P=NP [20]. An improved version of FFD, called MFFD, has asymptotic approximation ratio of $\frac{71}{60}$ [23].

In [24] an asymptotic polynomial time approximation scheme (APTAS) for the problem is presented. This result was improved by [25] who provided an AFPTAS (asymptotic fully polynomial time approximation scheme).

The online bin packing problem was first studied in [26] who showed that First Fit has competitive ratio $\frac{17}{10}$. A revised version of First Fit has competitive ration of $\frac{5}{3}$ [27]. Another simple online algorithm for the problem is Next Fit (NF) which has competitive ratio of 2 [28]. In [29] the HARMONIC algorithm, which uses a bounded space and has competitive ratio that approaches 1.69103, is presented. The best upper bound currently known is 1.58899 [30] and the best lower bound currently known is 1.54014 [31].

In the stochastic bin packing problem we are given a list of items where each item is a random variable and we are given an overflow probability p. The goal is to pack the items into a minimum number of unit-capacity bins such that the probability that the total size of the items in each bin exceeds 1 is at most p.

The study of stochastic bin packing from the perspective of approximation algorithms was initiated in [8]. They showed

an algorithm with approximation ratio of $O(\sqrt{\frac{\log p^{-1}}{\log\log p^{-1}}})$ for the stochastic bin packing problem with Bernoulli variables. Later, [9] obtained polynomial time approximation schemes for Poisson and exponential distributions. They also obtained a quasi-polynomial time approximation scheme for Bernoulli variables, assuming p is a constant. Their algorithms relax bin capacity constraints and overflow probability constraint by a factor $1+\epsilon$.

Recently, [7] provided an online algorithm for stochastic bin packing with Normal distribution with a competitive ratio of $(1+\epsilon)(1+\sqrt{2})$ for any $\epsilon > 0$. This work studies SBP in the context of VM consolidation in a virtualized data center, where VMs have dynamic bandwidth requirements. The results presented in [7] serve as a basis for the algorithms that we develop in this work.

III. Model and Problem Definition

A. Preliminaries

In this paper we consider the approximation ratio performance guarantee for approximation algorithms and the asymptotic competitive ratio performance guarantee for online algorithms. For a given input I, let $A(I)$ be the cost of algorithm A for input I and let $OPT(I)$ denote the cost of the optimal solution for input I. The asymptotic approximation ratio (or asymptotic competitive ratio) of algorithm A, R_A is defined to be $R_A = \lim_{n\to\infty}\left(\sup_{I:OPT(I)\geq n}\frac{A(I)}{OPT(I)}\right)$.

The absolute approximation ratio (or absolute competitive ratio) of algorithm A is the infimum R such that for any input I, $A(I) \leq R \cdot OPT(I)$.

B. Problem Definition

We are given a set of items $S = \{X_1, \ldots, X_n\}$, where each item is a random variable, and an overflow probability p. The problem is to find the minimum number of unit capacity bins that are needed in order to pack all the items, such that for each bin, the probability that its capacity is exceeded is at most p.

In this paper we consider the case where the items independently follow Normal distribution $\mathcal{N}(\mu_i, \sigma_i)$, where the distribution is left truncated at 0 since, obviously, bandwidth cannot assume negative values. For simplicity we will refer to items X_i simply as i wherever no ambiguity arises.

When $\sigma_i = 0$, for all i, then $X_i = \mu_i$ and the problem reduces to the classical bin packing problem, which is NP-hard [20].

A packing of the set S of items to the bins is a partition of the set of items S into sets $S_1, \ldots, S_m$. We say that a packing is feasible if for every bin j, $\mathbf{Pr}[\sum_{i:X_i\in S_j} X_i > 1] \leq p$.

Since the items are independent and Normally distributed, the total size of the items in bin j is a random variable with mean $\sum_{i:X_i\in S_j}\mu_i$ and variance $\sum_{i:X_i\in S_j}\sigma_i^2$.

Lemma 3.1: A packing is feasible for a given overflow probability p if and only if for every bin j, $\sum_{i:X_i\in S_j}\mu_i + \beta\sqrt{\sum_{i:X_i\in S_j}\sigma_i^2} \leq 1$, where $\beta = \Phi^{-1}(1-p)$ and the quantile function Φ^{-1} is the inverse function of the CDF Φ of $\mathcal{N}(0,1)$.

For the proof see [7].

To ensure a feasible packing exists, we assume that each item alone can be packed into a bin. Thus, $\forall i \in (1,n)$: $\mu_i + \beta\sigma_i \leq 1$.

Definition 3.1: The effective load of bin j is $l_j = \sum_{i:X_i\in S_j}\mu_i + \beta\sqrt{\sum_{i:X_i\in S_j}\sigma_i^2}$.

We denote the variance to mean ratio (VMR) of item i by $d_i = \sigma_i^2/\mu_i$.

A simple solution approach to the SBP is to reduce the problem to the classical bin packing problem by regarding each item i as a deterministic item of size $\mu_i + \beta\sigma_i$. Every feasible solution to the classical bin packing is also feasible for the SBP, since for any bin j, $\sum_{i\in S_j}\mu_i + \beta\sqrt{\sum_{i\in S_j}\sigma_i^2} \leq \sum_{i\in S_j}\mu_i + \beta\sigma_i$.

An example showing that the optimal number of bins for the classical bin packing problem when using $\mu_i + \beta\sigma_i$ as the size of item i may be much larger than the optimal number of bins for SBP is given in [7]. Thus, even if we could solve the classical bin packing problem, which is NP-hard, optimally, this optimal number of bins may be much larger than the optimum for SBP.

IV. Stochastic Bin Packing Algorithms

In this section we first show a 2-approximation algorithm. Then we present a $(2+\epsilon)$-competitive online algorithm for SBP with Normal distribution. Finally, we show that a dual PTAS for SBP can be obtained by reducing the problem to 2-dimensional vector bin packing problem [10].

A. Approximation Algorithm

We first show an approximation algorithm for SBP with Normal distribution.

Algorithm 1: First Fit VMR-Decreasing

1. Order the items in non-increasing order of VMR.
2. Place the next item (in non-increasing order of VMR) in the first bin into which it can be feasibly packed according to Lemma 3.1 (i.e., the effective load of the bin after placing the item does not exceed the bin capacity).
3. If no such bin exists, open a new bin to pack this item.

Our proof of the approximation ratio of Algorithm 1 is based on the structure of a certain feasible solution to the mathematical program for SBP with Normal distribution. Let $m \leq OPT$ be the least number of bins for which the following Mathematical Program (MP), a relaxation of SBP, is feasible. The value of m can be found easily using binary search.

$$\begin{aligned}
\sum_{i=1}^{n} x_{ij}\mu_i + \beta\sqrt{\sum_{i=1}^{n} x_{ij}\sigma_i^2} \leq 1 \quad & 1 \leq j \leq m, \qquad (1)\\
\sum_{j=1}^{m} x_{ij} = 1 \quad & 1 \leq i \leq n,\\
x_{ij} \geq 0 \quad & 1 \leq i \leq n, 1 \leq j \leq m.
\end{aligned}$$

In this mathematical program the variable x_{ij} denotes the fraction of item i assigned to bin j.

Definition 4.1: A vector $(v_1, \ldots, v_m)$ is greater than $(\bar{v}_1, \ldots, \bar{v}_m)$ lexicographically if for some i, $v_i > \bar{v}_i$ and $v_k = \bar{v}_k$ for all $k < i$.

Lemma 4.1: There exists a feasible solution to the MP with the following property. For any pair of items k, l and a pair of bins $i < j$, if $x_{kj} > 0$ and $x_{li} > 0$, then $d_l \geq d_k$.

Proof:

For a feasible solution to the MP, we denote the mean vector of the bins by $(M_1, \ldots, M_m)$, where $M_j = \sum_{i=1}^{n} x_{ij}\mu_i$. We denote the variance vector of the bins by $(V_1, \ldots, V_m)$, where $V_j = \sum_{i=1}^{n} x_{ij}\sigma_i^2$. We denote the effective load vector of the bins by $(L_1, \ldots, L_m)$, where $L_j = M_j + \beta\sqrt{V_j}$. Consider a feasible solution to the MP with lexicographically maximal variance vector of the bins $(V_1, \ldots, V_m)$. Suppose by contradiction that there exist items k, l and pair of bins $i < j$ such that $x_{kj} > 0$ and $x_{li} > 0$ and $d_k > d_l$. We exchange fractions of the items as follows. Let $y = \min\{x_{kj}\mu_k, x_{li}\mu_l\}$ and let $\delta = \frac{y}{\mu_l}$. We decrease x_{li} by δ and increase x_{lj} by δ. Next we increase x_{ki} by δ' such that constraint (1) for bin i becomes tight again and decrease x_{kj} by the same amount δ'. Let x' denote the new assignment, let $(M'_1, \ldots, M'_m)$ denote the new mean vector of the bins, let $(V'_1, \ldots, V'_m)$ denote the new variance vector of the bins and let $(L'_1, \ldots, L'_m)$ denote the new effective load vector of the bins. We will reach a contradiction by showing that the new solution x' is a feasible one with variance vector of the bins lexicographically greater than that of the solution x.

We have that $M'_i = M_i - \delta\mu_l + \delta'\mu_k$ and $V'_i = V_i - \delta\sigma_l^2 + \delta'\sigma_k^2 = V_i - d_l\delta\mu_l + d_k\delta'\mu_k$. Since $d_k > d_l$, it follows that $\delta'\mu_k < \delta\mu_l$ and thus $M'_i < M_i$. Otherwise, $M'_i \geq M_i$ and $V'_i > V_i$ and thus constraint (1) would be violated. Now, since $M'_i < M_i$ and constraint (1) is tight, we have that $V'_i > V_i$.

Since $V_i + V_j = V'_i + V'_j$ and $V'_i > V_i$, it follows that $V'_j < V_j$. Let $f(z) = \sqrt{V_i + z} + \sqrt{V_j - z}$. Since $V_i \geq V_j$, the function f is strictly decreasing for $z > 0$. Thus, $\sqrt{V'_i} + \sqrt{V'_j} < \sqrt{V_i} + \sqrt{V_j}$ and therefore $L'_i + L'_j < L_i + L_j$. Since constraint (1) for bin i is tight and $L'_i + L'_j < L_i + L_j$, it follows that constraint (1) holds for bin j. Thus, solution x' is a feasible solution to the MP. Moreover, since $V'_i > V_i$ and $V'_j < V_j$, it follows that $(V'_1, \ldots, V'_m)$ is lexicographically greater than $(V_1, \ldots, V_m)$. This is a contradiction to the lexicographic maximality of the vector $(V_1, \ldots, V_m)$. ∎

Algorithm 2: COMPUTE FRACTIONAL OPTIMUM

1 Order the items in non-increasing order of VMR.
2 Place the next item in the bin with remaining capacity according to constraint (1) of the MP as follows. Assign to this bin the maximum fraction of this item not violating constraint (1). If the item cannot be completely placed in this bin, open a new bin to pack the remaining part of this item.

Lemma 4.1 yields the following observation.

Observation 4.1: Algorithm 2 produces a feasible fractional solution to the MP.

We note that in each step of Algorithm 2 there is at most one open bin with remaining capacity according to constraint (1).

For an instance I to the problem we denote by $B(I)$ the number of bins used by our algorithm, by OPT(I) the number of bins used by the optimal algorithm and by FRAC(I) the minimum number of bins for which the MP has a feasible solution. Clearly, $FRAC(I) \leq OPT(I)$.

Theorem 4.2: Algorithm 1 is a 2-approximation algorithm for SBP with Normal variables.

Proof: We prove the theorem for the NEXT FIT version of Algorithm 1 that considers the items in the same order as Algorithm 1 and keeps a single active bin at each step. If the next item cannot be packed into this bin then the bin is closed and never used again and a new active bin is opened for placing this item. It is easy to see that the number of bins used by Algorithm 1 that places an item in the first bin into which it fits is at most the number of bins used by the NEXT FIT version of Algorithm 1.

Consider an instance I of the problem. For $j = 1, \ldots, B$, let U_j be the set of items assigned to bin j by the algorithm and let $i(j)$ be the item that caused the algorithm to open a new bin $j + 1$, since it could cause bin j to overflow if assigned to bin j. Consider the set of bins $j = 2k - 1$, for $k = 1, \ldots, \lceil B/2 \rceil$. For each item $i(j)$, let $x_{i(j)}$ denote the maximum fraction of item $i(j)$ that could be packed into bin j containing the set of items U_j without causing an overflow to bin j according to constraint (1) of the MP.

Consider the instance of the problem I' with the set of items U', where U' is the set of items containing the items U_{2k-1} and items $i(2k-1)$ for $k = 1, \ldots, \lceil B/2 \rceil$ where the latter are modified from their original values as follows. We decrease the mean and variance of items $i(2k-1)$ to $x_{i(2k-1)}\mu_{i(2k-1)}$ and $x_{i(2k-1)}\sigma^2_{i(2k-1)}$, respectively.

By Observation 4.1, $FRAC(I') = \lceil B/2 \rceil$. Clearly, $FRAC(I') \leq FRAC(I)$ and $FRAC(I) \leq OPT(I)$. Therefore, $B(I) \leq 2OPT(I)$. ∎

B. Online Algorithm

Next, we present an online algorithm for the problem. We partition the items in the input instance into classes according to their VMRs as follows. Class 0 consists of all items i with $d_i \leq \epsilon^2$. Let $C = \lceil \frac{8}{\epsilon} \ln \frac{1}{\epsilon} \rceil \geq \log_{1+\epsilon} \frac{1}{\epsilon^4}$. For $k = 1, \ldots, C$, class k consists of all items i with $\epsilon^2(1+\epsilon)^{k-1} < d_i \leq \epsilon^2(1+\epsilon)^k$. Class $C+1$ consists of all items with $\frac{1}{\epsilon^2} \leq \epsilon^2(1+\epsilon)^C < d_i$.

Algorithm 3: ON-LINE ALGORITHM

1 Classify next item according to the VMR classes.
2 Place the next item in the first bin of its class into which it can be feasibly packed according to Lemma 3.1. If no such bin exists, open a new bin in this class to pack this item.

Let I' be the modified instance obtained from the original instance I by rounding up each μ_i and σ_i of each item i as follows. Let k be the class of item i. If $0 \leq k \leq C$, we set $\sigma_i = \epsilon^2(1+\epsilon)^k\mu_i$. Otherwise, we set $\mu_i = \epsilon^2\sigma_i^2$. Throughout

this section we refer to instance I' as the rounded instance of instance I.

Lemma 4.2: Let $0 < \epsilon \leq \min\{\beta^2, 1/(2\beta), 1/2\}$ and let $\epsilon' = \max\{1, 2\beta\}\epsilon$. Let I be an instance of the problem where all items are from the same class $0 \leq k \leq C+1$. It holds that $FRAC(I') \leq (1+\epsilon')FRAC(I) + 1$.

Proof:

Let μ_{min} and μ_{max} be the minimum and maximum total mean, respectively, of the items from class k that can be packed into a bin, such that constraint (1) of the MP for the bin is tight. Let σ_{min} and σ_{max} be the minimum and maximum total variance, respectively, of the items from class k that can be packed into a bin such that constraint (1) of the MP for the bin is tight. We denote the total mean of all the items by $M = \sum_{i=1}^{n} \mu_i$ and the total variance of all the items by $V = \sum_{i=1}^{n} \sigma_i^2$.

We will show that the following holds.

1) If $0 \leq k \leq C$ then $\mu_{max} \leq (1+\epsilon')\mu_{min}$.
2) If $k = C+1$ then $\sigma_{max} \leq (1+\epsilon')\sigma_{min}$.

It follows from (1) that for class $0 \leq k \leq C$, $M/\mu_{max} \leq FRAC(I)$ and $FRAC(I') \leq \lceil M/\mu_{min} \rceil$. Therefore,

$$\begin{aligned} FRAC(I') &\leq \lceil M/\mu_{min} \rceil \\ &\leq \lceil (1+\epsilon')M/\mu_{max} \rceil \\ &\leq (1+\epsilon')M/\mu_{max} + 1 \\ &\leq (1+\epsilon')FRAC(I) + 1 \end{aligned}$$

It follows from (2) that for class $k = C+1$, $V/\sigma_{max} \leq FRAC(I)$ and $FRAC(I') \leq \lceil V/\sigma_{min} \rceil$. Therefore,

$$\begin{aligned} FRAC(I') &\leq \lceil V/\sigma_{min} \rceil \\ &\leq \lceil (1+\epsilon')V/\sigma_{max} \rceil \\ &\leq (1+\epsilon')V/\sigma_{max} + 1 \\ &\leq (1+\epsilon')FRAC(I) + 1 \end{aligned}$$

Now it remains to show that for class $0 \leq k \leq C$, $\mu_{max} \leq (1+\epsilon')\mu_{min}$ and for class $k = C+1$, $\sigma_{max} \leq (1+\epsilon')\sigma_{min}$.

We first consider class $0 \leq k \leq C$. We distinguish between two cases.

Case 1: $k = 0$. Since, VMR of class 0 lies in $[0, \epsilon^2]$, μ assumes the maximum value $\mu_{max} = 1$ when VMR=0 and μ_{min} is a solution to the equation $\mu + \beta\sqrt{\epsilon^2\mu} = 1$, which satisfies $\mu_{min} \geq 1 - \beta\epsilon$. Thus, $\mu_{max}/\mu_{min} \leq 1 + 2\beta\epsilon$.

Case 2: $1 \leq k \leq C$. Since VMR of class k lies in $(\epsilon^2(1+\epsilon)^{k-1}, \epsilon^2(1+\epsilon)^k]$, μ_{min} and μ_{max} are solutions to the equations $\mu + \beta\sqrt{\epsilon^2(1+\epsilon)^k\mu} = 1$ and $\mu + \beta\sqrt{\epsilon^2(1+\epsilon)^{k-1}\mu} = 1$, respectively. Thus, it easily follows that $\mu_{max}/\mu_{min} \leq 1 + \epsilon$.

We now consider class $C+1$. Clearly, $\sigma_{max} \leq 1/\beta^2$. Since VMR of class $C+1$ is at least $\frac{1}{\epsilon^2}$, then σ_{min} is a solution to the equation $\epsilon^2\sigma^2 + \beta\sqrt{\sigma^2} = 1$, which satisfies $\sigma_{max}/\sigma_{min} \leq 1+\epsilon$ for $\epsilon \leq \beta^2$. ∎

We prove theorem 4.3 for the NEXT FIT version of Algorithm 3 that considers the items in the same order and keeps a single active bin at each step. If the next item cannot be packed into this bin then the bin is closed and never used again and a new active bin is opened for placing this item. It is easy to see that the number of bins used by the FIRST FIT version of the algorithm that places an item in the first bin into which it fit is at most the number of bins used by NEXT FIT.

Let I_k, I'_k denote the subinstances of instances I, I', respectively, that contain only the items from class k. Recall that instance I' is the rounded instance of instance I. Let μ_i, μ'_i be the mean of item i and let σ_i, σ'_i be the standard deviation of item i in instances I_k and I'_k, respectively. We denote the total mean of the items packed in bin j by M_j, M'_j and the total variance of the items packed in bin j by V_j, V'_j for instances I_k and I'_k, respectively.

The proof of Theorem 4.3 requires the following lemma.

Lemma 4.3: For any k it holds that $B(I_k) \leq B(I'_k)$. Moreover, if $B(I_k) = B(I'_k)$ then $M_m \leq M'_m$ and $V_m \leq V'_m$, where $m = B(I_k)$.

Proof: We proceed by induction on $|I_k|$. Clearly, $B(I_k) = B(I'_k) = 1$ if $|I_k| = 1$. Suppose $|I_k| = i > 1$. By the induction hypothesis, the claim holds for the instances $I_k \setminus \{i\}$ and $I'_k \setminus \{i\}$.

Consider step $i = |I_k|$, where Algorithm 3 packs item i in instances I_k and I'_k.

We distinguish between two cases:

Case 1: $B(I_k \setminus \{i\}) < B(I'_k \setminus \{i\})$. If $B(I_k) = B(I'_k)$, it follows that $B(I'_k) = B(I'_k \setminus \{i\})$. Let $j = B(I'_k)$. Since, Algorithm 3 opened a new bin for item i in instance I_k and used already opened bin for item i in instance I'_k and the fact that $\mu_i = \mu'_i$ and $\sigma_i \leq \sigma'_i$, it follows that at the end of step i, $M_j \leq M'_j$ and $V_j \leq V'_j$. Otherwise, clearly $B(I_k) < B(I'_k)$.

Case 2: $B(I_k \setminus \{i\}) = B(I'_k \setminus \{i\})$. Let $j = B(I'_k)$. If $B(I'_k) = B(I'_k \setminus \{i\})$, then by the induction hypothesis and the fact that $\mu_i \leq \mu'_i$ and $\sigma_i \leq \sigma'_i$, it follows that the algorithm can pack item i of instance I_k in bin j. Moreover, at the end of iteration i, $M_j \leq M'_j$ and $V_j \leq V'_j$. Otherwise, $B(I'_k) > B(I'_k \setminus \{i\})$, clearly the claim holds. ∎

Let ϵ' be as defined in Lemma 4.2.

Theorem 4.3: Algorithm 3 is a $2(1+\epsilon')$-competitive online algorithm for stochastic bin packing with Normal variables.

Proof:

We first show a lower bound for $FRAC(I)$. Consider a feasible solution y created by Algorithm 2 for the instance I with $FRAC(I)$ bins and a feasible solution y_k created by Algorithm 2 for the instances I_k with $FRAC(I_k)$ bins. Since for each class k, the number of bins containing any items of class k in the solution y is at least $FRAC(I_k)$ and the number of bins containing items from at least two classes in the solution y is at most $C+1$ (recall that Algorithm 2 packs the items such that the items of each class are packed in group of consecutive bins), we get

$$FRAC(I) + C + 1 \geq \sum_{k=0}^{C+1} FRAC(I_k) \tag{2}$$

Now we show the upper bound for $B(I)$.

$$\begin{aligned} B(I) &= \sum_{k=0}^{C+1} B(I_k) \leq \sum_{k=0}^{C+1} B(I'_k) \leq \sum_{k=0}^{C+1} 2FRAC(I'_k) \\ &\leq \sum_{k=0}^{C+1} 2((1+\epsilon')FRAC(I_k)+1) \\ &= 2(1+\epsilon')\sum_{k=0}^{C+1} FRAC(I_k) + 2(C+2). \end{aligned}$$

The first inequality follows from Lemma 4.3. The second inequality follows from the proof of Theorem 4.2 and the fact that all the items in instance I'_k have the same VMR. The last inequality follows from Lemma 4.2. We now apply inequality (2) and get

$$\begin{aligned} B(I) &\leq 2(1+\epsilon')(FRAC(I)+C+1)+2(C+2)) \\ &= 2(1+\epsilon')FRAC(I)+(2+2\epsilon')(C+1)+2(C+2) \\ &\leq 2(1+\epsilon')OPT(I)+(4+2\epsilon')(C+2). \end{aligned}$$

The last inequality follows from the fact that $FRAC(I) \leq OPT(I)$. ∎

C. PTAS

We obtain a dual PTAS for SBP problem with Normal distribution by a simple reduction to the 2-dimensional Vector Bin Packing problem. The PTAS presented by Chekuri and Khanna [10] for the Vector Scheduling problem can be viewed as a dual PTAS for the 2-dimensional vector bin packing problem, where a dual PTAS for bin packing is a polynomial time algorithm that uses the optimal number of bins, but relaxes the bin capacity constraints by a factor $1+\epsilon$. We then use the dual PTAS for the 2-dimensional Vector Bin Packing problem to obtain a dual PTAS for SBP with Normal variables

We now define the vector bin packing problem. For a vector x, the quantity $\|x\|_\infty$ denotes the standard ℓ_∞ norm.

Definition 4.2: (Vector Bin Packing (VBP)) Given a set of n rational vectors $p_1, \ldots, p_n$ from $[0,1]^d$, find a partition of the set into sets $S_1, \ldots, S_m$, such that $\|\bar{S}_i\|_\infty \leq 1$ for $1 \leq j \leq m$. The objective is to minimize the size of the partition.

The linearity of the mean and variance of the items that are packed into a bin can be used to translate an instance of SBP with normal variables to an instance of two-dimensional vector bin packing as follows. Each item i with mean μ_i and standard deviation σ_i is replaced by a 2-dimensional item $p_i = (p_{i1}, p_{i2}) = (\mu_i, \beta\sigma_i)$. We normalized the second coordinate to ensure that the range of both coordinates is $[0,1]$. We add a new constraint as follows.

$$\textstyle\sum_{i\in S_j} p_{i1} + \sqrt{\sum_{i\in S_j} p_{i2}^2} \leq 1 \quad 1 \leq j \leq m, \qquad (3)$$

where S_j is the set of items packed in bin j. Thus, the effective load of a bin does not exceed 1 if and only if constraint (3) holds. We observe that the dual PTAS for the vector bin packing problem can be simply extended to deal with the additional constraint (3) that ensures that the effective load of a bin does not exceed 1. This observation follows from the fact that constraint (3) can be handled by the PTAS as follows. The algorithm guesses the capacity configuration of the bins, where a capacity configuration of a bin is a pair (c_1, c_2) such that $0 \leq c_i \leq \lceil 1/\epsilon \rceil$. A capacity configuration of a bin approximately describes how a bin is filled. A packing of vectors is said to *respect* a capacity configuration (c_1, c_2) if the load of the packing in each dimension k is at most ϵc_k. We can prune capacity configurations that do not respect constraint (3) where we relax the bin capacity to $1+2\epsilon$. This pruning maintains approximate bin configurations of valid bin configurations, since the approximate configuration (c_1, c_2) of any valid bin filling represented by a 2-dimensional vector r, where r_i is the height of the packing in dimension i satisfies $\epsilon c_1 \leq r_1 + \epsilon$ and $\epsilon c_2 \leq r_2 + \epsilon$. Therefore, if $r_1 + \sqrt{r_2^2} \leq 1$ then $\epsilon c_1 + \sqrt{(\epsilon c_2)^2} \leq 1 + 2\epsilon$. The PTAS for SBP proceeds as follows. We guess the number of bins m in an optimal solution to the problem (can be found easily using binary search) and then we solve the corresponding 2-dimensional vector bin packing using its dual PTAS. Since constraint (3) holds for bin capacity $1+2\epsilon$, the solution is feasible for SBP with bins of size $1+2\epsilon$.

V. Simulation Study

In this section we present the results of our simulation study. We used two sets of data to compare the performance of our proposed algorithms with that of the previously reported ones. The first data set comprises synthetic traces that we generate to simulate a typical use case. The second data set is a real production data center trace that we used to compute the mean and standard deviation of bandwidth consumption for 6000 VMs over a few hours period.

For the sake of a thorough evaluation, we implemented the following bin packing algorithms:

- Algorithms 1–3;
- First Fit (FF) with deterministic item sizes $\mu_i + \beta\sigma_i$;
- First Fit Descending (FFD) according to deterministic item sizes $\mu_i + \beta\sigma_i$;
- Group Packing (GP) reported in [7].

We compare performance of Algorithm 1 and FFD, which are off-line algorithms and algorithms FF, GP and Algorithm 3, which are on-line algorithms. For both off-line and on-line algorithms we compare their performance with the lower bound for the optimal solution obtained by Algorithm 2.

We start with presenting our results for synthetic data. We generate problem instances ranging from 2000 to 50000 items. To make statistical properties of the synthetic trace similar to those reported for the real traces (e.g., in [7]), 80% of the generated items have their standard deviation smaller than twice their mean. For the rest of the items, standard deviation ranges from twice an item mean to four times an item mean. Figure 1 depicts performance of Algorithm 1 and FFD for problem instances of different size. In these experiments the overflow probability $p = 0.01$. This value of p yields $\beta = 2.32623$.

As one can see, FFD uses 70% more bins than Algorithm 1 and the number of bins used by Algorithm 1 is at most 0.3% more than the lower bound.

Figures 2, 3 present the results of performance evaluation for the online algorithms for small and large problem instances for $\epsilon = 0.1$. We split the graphs into two parts to avoid visual

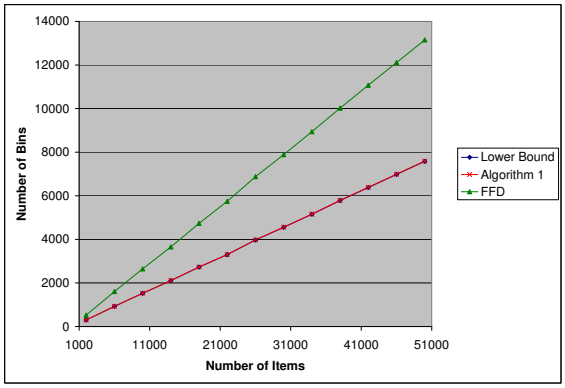

Fig. 1. Comparing Off-Line Algorithms on Synthetic Data

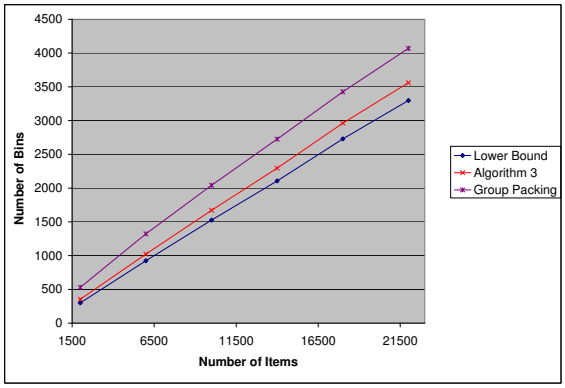

Fig. 2. Comparing On-Line Algorithms on Synthetic Data

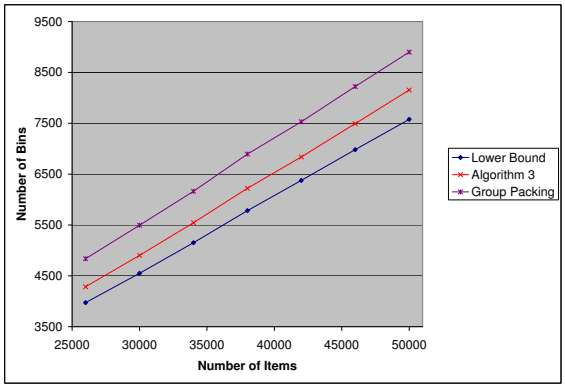

Fig. 3. Comparing On-Line Algorithms on Synthetic Data

distortion due to the scale used. For the same reason we do not show performance of FF on these graphs. FF performed significantly worse than the rest of the algorithms as we explain below. The data set used to benchmark the online algorithms is the same synthetic data set as the one used to compare the offline algorithms, but the data set is being processed one VM at a time.

As Figure 2 shows, for small instances ranging from 2000 to 22000 items, Algorithm 3 used from 18% to 8% more bins, respectively, than Algorithm 2, which obtains a theoretical lower bound. Group Packing algorithm used from 50% to 14% more bins than Algorithm 3.

Figure 3 presents the results for the large problem instances ranging from 26000 to 50000 items. Algorithm 3 consistently used at most 8% more bins than the theoretical lower bound. On the large problem instances. Group Packing algorithm used from 12.8% to 9% more bins than Algorithm 3.

FF used from 62% to 49% more bins than Algorithm 3 on the same problem instances ranging from 2000 to 50000 items respectively.

As expected, for smaller instances Algorithm 3 significantly outperforms the Group Packing algorithm, but for the larger instances the gap between the two stabilizes around 9% in favor of Algorithm 3. The reason for that is that both algorithms approach their competitive ratios asymptotically. There is a constant factor that is paid by both algorithms due to classifying items into classes and spending at least one bin per every non-empty class. In the Group Packing algorithm the number of classes is proportional to $\frac{\ln(1/\mu_{min})}{\epsilon\mu_{min}}$, where μ_{min} is the minimal item mean. Therefore, the number of classes in the Group Packing algorithm potentially can be much larger than in Algorithm 3, where it is proportional to $\frac{1}{\epsilon}\ln\frac{1}{\epsilon}$ and, therefore, depends only on ϵ, but not on the properties of the problem instance.

p	Algorithm 3	Algorithm 1	Group Packing	FFD	FF	Algorithm 2
0.1	164	146	595	332	334	144
0.01	215	195	785	519	522	192
0.001	263	243	881	656	662	237

TABLE I
COMPARING ALGORITHMS ON REAL DATA

Table I summarizes the online and offline algorithms performance on the real trace that was at our disposal. We evaluate the algorithms for three values of target overflow probability p: 0.001, 0.01, and 0.1, which correspond to the percentiles widely used in today Cloud practices to specify up-time SLAs for VMs. Even though bandwidth SLAs usually are not provided in the public clouds today, we believe that in the future, they will be provided and the percentiles of success in those SLAs will be similar to the percentiles used today for the more simple VM up-time SLAs; hence the choice of values for the overflow probability to assess the practical applicability of the proposed algorithms.

As Table I shows, both Algorithm 3 (online) and Algorithm 1 (offline) obtain the number of bins that is very close to the theoretical lower bound obtained by Algorithm 2, with the offline algorithm performing $8\% - 10\%$ better then Algo-

rithm 3, the online one, as naturally expected. The number of bins used by Algorithm 3 grows by 30% and 20% as the target overflow probability reduces by an order of magnitude and two order of magnitudes, respectively. Algorithm 3 consistently outperforms Group Packing algorithm by a factor ranging between 3.3 and 3.65 and First Fit by a factor ranging from 2 to 2.5. Moreover, as one can see in Table I, the number of bins packed by Algorithm 3 for target overflow probability $p = 0.001$ is significantly smaller than the number of bins packed by Group Packing, FF and FFD algorithms for target overflow probability $p = 0.1$. The relatively poor performance of the Group Packing algorithm is explained by the small size of the problem instance (6000 items) and the fact that VMs with relatively small bandwidth consumptions appear in the trace. This increases the number of item classes as explained above. We believe that on larger problem instances and on the instances with less variability in bandwidth consumption across different VMs, Group Packing algorithm will behave significantly better (as suggested by our experiments on the synthetic data).

Now we turn to validating the packing obtained by the algorithms and the question we want to answer is whether we indeed obtained the packing that respects the target values of the overflow probability.

To validate the actual overflow probability we performed the following procedure. For each packing that was obtained by the algorithms and for each bin in the packing, we sampled the trace to obtain 2000 samples of the momentary values of each item size (i.e., bandwidth consumption) and summed up the total momentary bandwidth consumption per bin checking whether the bin capacity constraint is broken.

Figure 4 shows the Cumulative Distribution Function (CDF) of bin compliance with the overflow probabilities. As one can see, Algorithm 1 and Algorithm 3 performed significantly worse for the target overflow value $p = 0.01$ then the Group Packing, FF, and FFD algorithms. More specifically, only 18% of the bins packed by Algorithm 3 respected the target overflow probability 0.01 while more than 80% of bins packed by the Group Packing algorithm and more than 90% of the bins packed by FF and FFD obeyed the target overflow probability. At the same time, we see that for all bins packed by our algorithms, the actual overflow probability achieved was 5%. The explanation for this is as follows. Our algorithms create much better packing than their counterparts. This packing is produced under the assumption that VM bandwidth follows Normal distribution. By inspecting the trace we found out that (a) the distribution of bandwidth consumption of a single VM may deviate from the Normal distribution and (b) many large items appear in the trace. Therefore, the actual average load per bin in our approach is larger than the theoretically expected effective load and because we use less bins than other algorithms, this increases the overflow probability.

Although the probability violation that we obtain is larger than the target overflow probability, this is still a sufficiently low rate of violations to be applicable in a public Cloud (which today do not support any bandwidth SLAs at all). Given that this rate is achieved while reducing the number of bins by 50% to 70% (compared to other algorithms as explained above), the Cloud provider can benefit significantly from employing the proposed algorithms.

Figure 5 summarizes the same validation experiment, but the target overflow probability for the packing algorithms was increased to 0.1. As one can see, 80% of bins produced by Algorithm 3 respected the target overflow probability and 99% of them had overflow probability at most 0.12.

In general, when we consider large physical hosts or racks that are capable of co-hosting tens and hundreds of VMs and the size of each VM is small relatively to the total capacity of a host or rack, the importance of deviation of specific VMs from the Normality assumption diminishes. This is due to the well known Central Limit theorem that states that the distribution of a sum of the random variables asymptotically approaches Normal distribution irrespectively of the distributions of the individual variables. Therefore, as larger hosts/racks are used for consolidating VMs, the actual overflow probability approaches the target one.

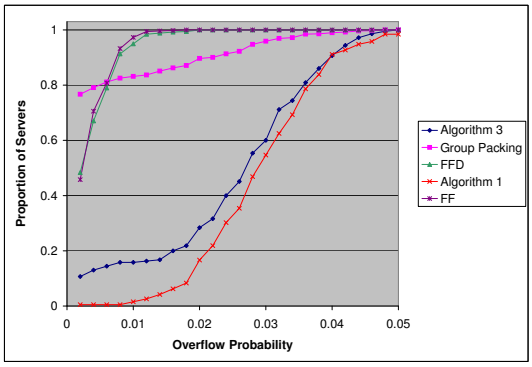

Fig. 4. Overflow Probability for 6K instance size and target overflow probability $p = 0.01$

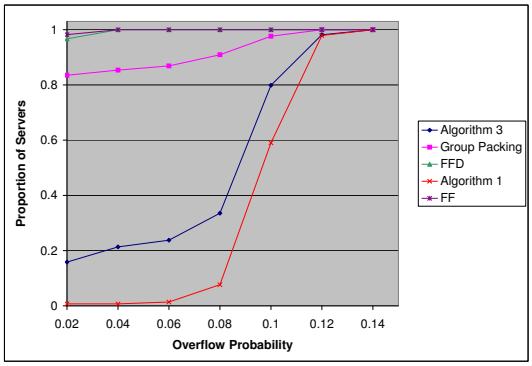

Fig. 5. Overflow Probability for 6K instance size and target overflow probability $p = 0.1$

Figure 6 compares the algorithms using a large synthetic

problem instance comprising 30000 items. As one can see, in all algorithms, the number of bins decreases as the target overflow probability increases. This behavior is explained by the fact that as the target overflow probability increases, the weight of the standard deviation component (i.e., β) of the effective load reduces and, thus, the effective load decreases.

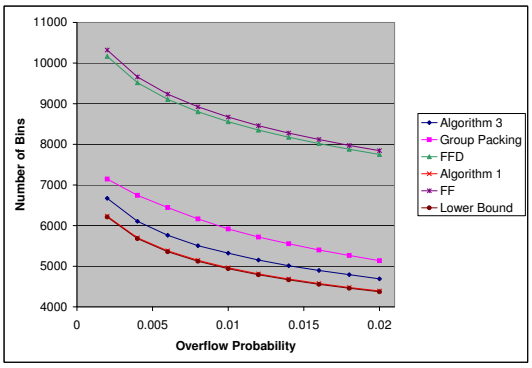

Fig. 6. Comparing the algorithms for different overflow probabilities

VI. Conclusions and Future Work

We presented off-line and on-line algorithms for SBP. Under the assumption that virtual machines' bandwidth consumption obeys Normal distribution, we showed a 2-approximation algorithm for the off-line version and a $(2+\epsilon)$-competitive algorithm for the on-line one. We also observed that a dual PTAS for SBP can be obtained via reduction to the 2-dimensional vector bin packing problem.

Using synthetic and real data sets we thoroughly studied performance of our proposed algorithms and their applicability to real life scenarios.

Our proposed algorithms considerably reduce the number of bins compared to the best known algorithms for SBP with Normally distributed random variables.

Today public clouds do not provide explicit SLAs guarantees on VM bandwidth. However, traditional hosting environments provide bandwidth guarantees as a common practice. Thus, we may expect that the Cloud SLA practices will also be extended to bandwidth in the future.

Our algorithms can be used as building blocks to support such future SLAs in a cost-efficient manner

Our approach is general, so it can be applied to resources other than bandwidth. We defer this to future work.

Other topics that we plan to explore in the future, include multidimensional SBP to consider resources other than bandwidth, and other statistical distributions. Also, we plan to validate our approach with more data coming from the real production environments.

Acknowledgements

We thank M. Wang (Cornell Univ.), X. Meng (IBM), and L. Zhang (IBM) for kindly sharing their traces with us.

References

[1] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM'10*, 2010.

[2] M. Girola, A. M. Tarenzio, M. Lewis, and M. Friedman, "SG24-7928-00, IBM Data Center Networking: Planning for virtualization and cloud computing," http://www.redbooks.ibm.com/redpieces/pdfs/sg247928.pdf, IBM, Feb 2011.

[3] J. Schad, J. Dittrich, and J.-A. Quiane-Ruiz, "Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance," *Proceedings of the VLDB Endowment*, vol. 3, no. 1, 2010.

[4] 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.

[5] 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.

[6] B. Urgaonkar, P. J. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in a shared internet hosting platform," *ACM Transactions on Internet Technology*, vol. 9, no. 1, 2009.

[7] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *INFOCOM*, 2011.

[8] J. M. Kleinberg, Y. Rabani, and E. Tardos, "Allocating bandwidth for bursty connections," *SIAM J. Comput.*, vol. 30, no. 1, pp. 191–217, 2000.

[9] A. Goel and P. Indyk, "Stochastic load balancing and related problems," in *FOCS*, 1999, pp. 579–586.

[10] C. Chekuri and S. Khanna, "On multidimensional packing problems," *SIAM J. Comput.*, vol. 33, no. 4, pp. 837–851, 2004.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *International Conference for the Computer Measurement Group (CMG)*, 2007.

[16] 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.

[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] M. Yue, "A simple proof of the inequality FFD (L) $\leq$ 11/9 OPT(L)+1, $\forall$ L, for the FFD bin-packing algorithm," *Acta Mathematicae Applicatae Sinica (English Series)*, vol. 7, pp. 321–331, 1991.

[22] D. Simchi-Levi, "New worst-case results for the bin-packing problem," *Naval Res. Logist.*, vol. 41, pp. 579–585, 1994.

[23] M. Yue and L. Zhang, "A simple proof of the inequality MFFD(L) $\leq 71/60OPT(L)+1$, L for the MFFD bin-packing algorithm," *Acta Mathematicae Applicatae Sinica (English Series)*, vol. 11, pp. 318–330, 1995.

[24] W. F. de la Vega and G. S. Lueker, "Bin Packing can be Solved within $1+\epsilon$ linear time," *Combinatorica*, vol. 1, no. 4, pp. 349–355, Dec 1981.

[25] N. Karmarkar and R. M. Karp, "An efficient approximation scheme for the one-dimensional bin-packing problem," in *FOCS*, 1982, pp. 312–320.

[26] J. D. Ullman, "The performance of a memory allocation algorithm," *Technical Report 100*, 1971.

[27] A. C.-C. Yao, "New algorithms for bin packing," *J. ACM*, vol. 27, no. 2, pp. 207–227, 1980.
[28] D. S. Johnson, "Fast algorithms for bin packing," *J. Comput. Syst. Sci.*, vol. 8, no. 3, pp. 272–314, 1974.
[29] C. C. Lee and D. T. Lee, "A simple on-line bin-packing algorithm," *J. ACM*, vol. 32, no. 3, pp. 562–572, 1985.
[30] S. S. Seiden, "On the online bin packing problem," *J. ACM*, vol. 49, no. 5, pp. 640–671, 2002.
[31] A. van Vliet, "An improved lower bound for on-line bin packing algorithms," *Inf. Process. Lett.*, vol. 43, no. 5, pp. 277–284, 1992.