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

David Breitgand Amir Epstein
Virtualization Technologies, System Technologies & Services
IBM Research - 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) in a cloud where the physical network (e.g., network interface or top of the rack switch link) may become a bottleneck. Since virtual machines do not simultaneously use maximum of their nominal bandwidth, the capacity of the physical container can be multiplexed. We assume that each virtual machine has a probabilistic guarantee on realizing its bandwidth Requirements–as derived from its Service Level Agreement with the cloud provider. Therefore, the problem of consolidating virtual machines on the minimum number of physical containers, while preserving these bandwidth allocation guarantees, can be modeled as a *Stochastic Bin Packing (SBP)* problem, where each virtual machine's bandwidth demand is treated as a random variable.**

We consider both offline and online versions of SBP. Under the assumption that the virtual machines' bandwidth consumption obeys normal distribution, we show a 2-approximation algorithm for the offline version and improve the previously reported results by presenting a $(2+\epsilon)$-competitive algorithm for the online version. We also observe that a dual polynomial-time approximation scheme (PTAS) for SBP can be obtained via reduction to the two-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 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 use 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]. With the cloud paradigm attracting an ever increasing number of customers and the proliferation of network traffic in and out of the cloud, it has become essential to share the network bandwidth without hurting the VMs' quality of service. While over-subscription of other physical resource types such as CPU and memory has been studied intensively in recent years [4]–[6], bandwidth over-subscription in virtualized cloud-like environments has attracted less attention thus far [7].

In this work, we focus on placement algorithms that allow the consolidation of VMs on a minimum number of hosts/racks. Although the bandwidth demand of these VMs varies over time, the placement should ensure that the physical bandwidth capacity shared by the VMs placed together on the same container will be sufficient to satisfy the total amount of bandwidth demand with a given probability. This allows us to control the risk of bandwidth congestion on the shared physical network device.

To satisfy the 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 the 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 the advantages of the cloud computing approach. It will result in resource waste at the provider side and higher costs to the customers.

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, the dynamic bandwidth consumption of a VM can be regarded as a random variable. When many such random variables are "packed" together in the same bin, the standard deviation of the random variable that represents the total bandwidth demand is smaller than the sum of standard deviations of the individual variables. This phenomenon is known as *smoothing*.

The 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.

This problem can be formulated as a Stochastic Bin Packing problem (SBP) where the items are random variables representing the dynamic bandwidth consumption of VMs [8], [9].

The goal is to consolidate VMs belonging to the same Service Level Agreement (SLA) class on the minimum number of bins (hosts or racks) such that the probability of satisfying the VM bandwidth demand is at least $1 - p_{sla}$. For simplicity, we will consider a single SLA class and refer to p_{sla}, the probability of resource congestion, called target *overflow probability*, per physical container, simply as p.

A. Our Contribution and Paper Organization

Recently, Wang et al. [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 the 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 observe that a dual PTAS for SBP can be obtained by reduction to the two-dimensional vector bin packing problem [10]. In the dual PTAS, the number of bins produced by the algorithm is optimal, but the capacity constraints of each bin are relaxed by a factor of $1+\epsilon$, for $\epsilon > 0$. Due to the lack of space we do not include the discussion on obtaining the dual PTAS for SBP in this paper. More details are available in [11].

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

The rest of this paper is organized as follows. Section II discusses the 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 directions for future work.

II. Related Work

There is a serious need to improve cost-efficiency by reducing capital investments in computing infrastructure and operational costs such as energy, floor space, and cooling. These needs are driving the research efforts in VM consolidation [12]–[14] and are motivating new features in products such as VMware and the IBM Server Planning Tool [15], [16]. 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 minimizing migration, optimizing performance, and others are considered. Variants of the classic packing algorithms such as First Fit (FF) and First Fit Decreasing (FFD) [17] are often used to achieve practical solutions.

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 exceeding 1 is at most p.

The study of stochastic bin packing from the perspective of approximation algorithms was initiated in [8], which presented an algorithm with an 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 of $1+\epsilon$.

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 as $R_A = \lim_{n\to\infty}\left(\sup_{I:OPT(I)\geq n}\frac{A(I)}{OPT(I)}\right)$.

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 goal 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.

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 [17].

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, $\sum_{i\in S_j} \mu_i + \beta\sqrt{\sum_{i\in S_j} \sigma_i^2} \leq \sum_{i\in S_j} \mu_i + \beta\sigma_i$.

As shown in [7], 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. Thus, even if we could optimally solve the classical bin packing problem, which is NP-hard, 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. The proof of the competitive ratio is only partially presented in this paper due to lack of space. The full proof can be found in [11].

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.

$$\sum_{i=1}^{n} x_{ij}\mu_i + \beta\sqrt{\sum_{i=1}^{n} x_{ij}\sigma_i^2} \leq 1 \quad 1 \leq j \leq m, \qquad (1)$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad 1 \leq i \leq n,$$

$$x_{ij} \geq 0 \qquad 1 \leq i \leq n, 1 \leq j \leq m.$$

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 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 $M_i' = M_i - \delta\mu_l + \delta'\mu_k$ 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 $V_i' > V_i$.

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: ONLINE 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 $0 < \epsilon \leq \min\{\beta^2, 1/(2\beta), 1/2\}$ and let $\epsilon' = \max\{1, 2\beta\}\epsilon$.

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

The proof appears in [11].

V. SIMULATION STUDY

In this section we present the results of our simulation study. 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 the performance of Algorithm 1 and FFD, which are offline algorithms and algorithms FF, GP and Algorithm 3, which are the online ones. For both offline and online algorithms we compare their performance with the lower bound for the optimal solution obtained by Algorithm 2. We evaluate performance using the real data center trace previously reported in [7].

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 performance of the online and offline algorithms on that trace. We evaluate the algorithms for three values of target overflow probability p: 0.001, 0.01, and 0.1.

As Table I shows, both Algorithm 3 (online) and Algorithm 1 (offline) obtain a number of bins that is very close to the theoretical lower bound obtained by Algorithm 2, with the offline algorithm performing $8\% - 10\%$ better than Algorithm 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 the Group Packing algorithm by a factor ranging between 3.3 and 3.65 and the First Fit by a factor ranging from 2 to 2.5. Note that 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 the Group Packing, FF and FFD algorithms for target overflow probability $p = 0.1$

The relatively poor performance of the Group Packing algorithm can be 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 used by the algorithm. We believe that on larger problem instances and on the instances with less variability in bandwidth consumption across different VMs, the 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. 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). We summed up the total momentary bandwidth consumption per bin, checking whether the bin capacity constraint was broken. We used these samples to create a Cumulative Distribution Function (CDF) of bin compliance with the overflow probabilities.

Algorithm 1 and Algorithm 3 performed significantly worse for the target overflow value $p = 0.01$ than 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 the 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, 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 a 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 rate of violation is still low enough to be applicable in a public cloud (which today does 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.

When the target overflow probability for the packing algorithms was increased to 0.1, 80% of bins produced by Algorithm 3 respected the target overflow probability and 99% of them had an 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 relative 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, which states that the distribution of a sum of the random variables asymptotically approaches normal distribution irrespective 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.

To extend our performance study to larger problem instances, we also compared the relative performance of the algorithms using synthetic data. We generated problem instances ranging from 2000 to 50000 items, making statistical properties of the synthetic traces similar to those reported for the real trace that was in our possession. This study showed that our proposed algorithms result in near optimal solutions and consistently outperform the Group Packing algorithm previously reported in [7]. For more details see [11].

VI. Conclusions and Future Work

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

We presented offline and online algorithms for SBP that 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 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] D. Breitgand and A. Epstein, Improving Consolidation of Virtual Machines with Risk-Aware Bandwidth Oversubscription in Compute Clouds. IBM Technical Report, H-0312, 2012.

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

[15] VMware Inc., "Resource Management with VMware DRS, Whitepaper," 2006.

[16] IBM, "Server Planning Tool," http://www-304.ibm.com/jct01004c/systems/support/tools/systemplanningtool/.

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.