Efficient VM Placement with Multiple Deterministic and Stochastic Resources in Data Centers

Hao Jin, Deng Pan, Jing Xu, and Niki Pissinou
Florida International University
Miami, FL

Abstract—**Virtual machines (VMs) may significantly improve the efficiency of data center infrastructure by sharing resources of physical servers. This benefit relies on an efficient VM placement scheme to minimize the number of required servers. Existing VM placement algorithms usually assume that VMs' demands for resources are deterministic and stable. However, for certain resources, such as network bandwidth, VMs' demands are bursty and time varying, and demonstrate stochastic nature. In this paper, we study efficient VM placement in data centers with multiple deterministic and stochastic resources. First, we formulate the Multidimensional Stochastic VM Placement (MSVP) problem, with the objective to minimize the number of required servers and at the same time satisfy a predefined resource availability guarantee. Then, we show that the problem is NP-hard, and propose a polynomial time algorithm called Max-Min Multidimensional Stochastic Bin Packing (M^{3}SBP). The basic idea is to maximize the minimum utilization ratio of all the resources of a server, while satisfying the demands of VMs for both deterministic and stochastic resources. Next, we conduct simulations to evaluate the performance of M^{3}SBP. The results demonstrate that M^{3}SBP guarantees the availability requirement for stochastic resources, and M^{3}SBP needs the smallest number of servers to provide the guarantee among the benchmark algorithms.**

I. Introduction

Virtual machines (VMs) are attractive to modern data centers because they may significantly promote the efficiency and flexibility [4], [10], [20]. However, such an incentive highly relies on a well-designed VM placement scheme [16], [17]. This is because an inefficient placement scheme may result in lower resource utilization and thus needs more physical servers, which will further lead to not only higher capital investments on equipment and facilities, but also increased operational expenditures on energy and labor.

VM placement needs to consider VMs' demands for various resources. CPU and memory are traditionally the two major considerations. More recently, due to the increasing concerns on data center energy [1] [9] and emerging bandwidth intensive applications [3] [11], VMs' power consumption [12] [14] and bandwidth requirement are also taken into account when computing the placement. It has started attracting attention in the research community to find optimal VM placement for VMs with multiple resource demands. Several models and heuristics are proposed in recent literature. Xu et al. [19] model VM placement as a multi-objective optimization problem. A fuzzy multi-objective evaluation aided genetic algorithm is proposed to search large solution space for large scale data centers. While [19] considers CPU, memory, power consumption and thermal dissipation as its placement criteria, Meng et al. [15] focus their attention on network bandwidth. Their goal is to find an optimized VM placement scheme to improve the network scalability for traffic-intensive data centers. An approximation algorithm aiming to reduce the average traffic latency is proposed. The algorithm takes a two-tier approach that first divides VMs and servers into clusters respectively, and then matches VMs and servers at the cluster and server levels consequently.

One common assumption of the aforementioned schemes is that, all resources are *deterministic* resources for which the demands are stable over time. The placement computing for this type of resource can be simply done by comparing the VM's resource demand with the server's available capacity. We refer VM placement algorithms handling only deterministic resources as deterministic algorithms.

However, recent studies [3], [5], [11], [13] indicate that VMs' demands for certain resources are highly bursty, and can be modeled as stochastic processes. In other words, the real demands of these *stochastic* resources are fluctuating, and it is difficult to obtain an accurate fixed-value measure. One such example is network bandwidth, and it is shown [11], [18] that bandwidth demands of VMs in data centers can be approximated by the normal distribution. As a result, besides supplying fixed-value deterministic resources requested by VMs, servers provide an availability guarantee for stochastic resources in the form of a violation probability threshold, specified in the service level agreements (SLAs). The threshold gives the worst-case likelihood that a server cannot satisfy the dynamical demands of a VM for stochastic resources.

It may seem straightforward for deterministic algorithms to handle stochastic resources by estimating an equivalent fixed-value demand. However, it has been shown in [18] that this estimation is not accurate, since the equivalent demand for a stochastic resource of individual VMs vary under different placement schemes. Such a naive approach may result in either waste of server resources or violation of servers' availability guarantee in the SLA. To tackle this challenge, Wang et al. [18] propose a method that first calculates a total equivalent demand for all VMs hosted by the same server, and then compares it with the server's capacity to determine whether a placement is valid.

In this paper, we study VM placement in data centers with multiple deterministic and stochastic resources. First, we model the studied issue as a Multidimensional Stochastic VM Placement (MSVP) problem, with the objective to minimize the number of required servers while satisfy the SLA availability guarantee. Then, we show that this problem is NP-hard, and propose a polynomial time algorithm named Max-Min Multidimensional Stochastic Bin Packing (M^{3}SBP). The basic idea is to maximize the minimum utilization ratio of all the resources of a server, while satisfying the VMs' demands for both deterministic and stochastic resources. Next, we conduct simulations to evaluate M^{3}SBP's performance. In the simulations, each VM has three deterministic resources including *CPU*, *Memory* and *Power Consumption*, and one stochastic resource *Bandwidth* that follows the normal distribution. The results show that M^{3}SBP guarantees the availability requirement for the stochastic resource while employing fewer servers than other benchmark algorithms. Moreover, the results demonstrate that, compared to modified deterministic algorithms that simply do over-provisioning for stochastic resources, M^{3}SBP obtains more efficient placement schemes.

II. Problem Formulation

In this section, we describe the Multidimensional Stochastic VM Placement (MSVP) problem and present the problem formulation.

We consider a scenario in which there are n VMs with m kinds of resources to be placed into a number of servers. Among the m resources, there are both deterministic and stochastic resources. Taking into account the homogeneous architectures of modern data centers [2], [8], the servers are assumed to have identical capacities. In order to host VM v_i, server s needs to meet all of its resource demands. For the deterministic resource, this can be simply done by assigning the same amount of resource as requested. However, the demand of stochastic resource is time-varying, and it is challenging to calculate an accurate resource allocation. Thus, for each stochastic resource, a violation probability threshold is defined to specify the maximum probability that the server's capacity of this resource is exceeded. In our scenario, we assume all stochastic resources share the same violation probability threshold α as in the SLA. Therefore, considering the multidimensional and stochastic characteristics of VM's demands, a placement scheme is considered valid only if it satisfies the following two conditions:

Condition 1) The capacities of the server's deterministic resources should not be exceeded by the total amount of demands of all hosted VMs;

Condition 2) The probability that the capacities of the server's stochastic resources are exceeded is no larger than the given violation probability threshold.

The objective of the MSVP problem is thus to find a VM placement scheme, such that the above two conditions are satisfied and the number of required servers is minimized.

Denote the demand of a deterministic resource p of VM v_i as $D_p(v_i)$ and server s's corresponding capacity as $C_p(s)$. *Condition 1* can be represented as follows,

$$\forall p \in P, \sum_{i\in U} D_p(v_i) \leq C_p(s) \tag{1}$$

where P is the set of all deterministic resources and U is the set of VMs hosted by s.

Assume the demand of stochastic resource q of VM v_i independently follows a normal distribution $N(\mu_q(v_i), \sigma_q^2(v_i))$. An equivalent total demand of all VMs within the same server for each stochastic resource can be calculated based on each VM's distribution and the server's violation probability threshold α as follows,

$$\forall q \in Q, \sum_{i\in U} \mu_q(v_i) + \Phi^{-1}(1-\alpha)\sqrt{\sum_{i\in U} \sigma_q^2(v_i)} \tag{2}$$

where Q is the set of stochastic resources of v_i, U is the set of VMs already placed in the current server, and $\Phi^{-1}(1-\alpha)$ is the quantile of $N(0,1)$ at probability α. Thus, Condition 2 can be quantified by the equivalent total demand as follows,

Quantified Condition 2: The capacities of each server's stochastic resources should not be exceeded by the equivalent total demand of all hosting VMs.

Denote server s's capacity of stochastic resource q as $C_q(s)$. Then, *Condition 2* can be formulated as follows,

$$\forall q \in Q, \sum_{i\in U} \mu_q(v_i) + \Phi^{-1}(1-\alpha)\sqrt{\sum_{i\in U} \sigma_q^2(v_i)} \leq C_q(s) \tag{3}$$

We define the TCR ratio for each resource of a server to be the ratio between the total demand of all VMs within the same server and the server's capacity for this resource. Denote $R_p(s)$ and $R_q(s)$ as the TCRs of deterministic resource p and stochastic resource q, respectively. Then we have,

$$\forall p \in P, R_p(s) = \frac{\sum_{i\in U} D_p(v_i)}{C_p(s)} \tag{4}$$

$$\forall q \in Q, R_q(s) = \frac{\sum_{i\in U} \mu_q(v_i) + \beta\sqrt{\sum_{i\in U} \sigma_q^2(v_i)}}{C_q(s)} \tag{5}$$

where $\beta = \Phi^{-1}(1-\alpha)$.

Therefore, by combining (1), (3), (4) and (5) , the MSVP problem can be formulated as follows,

$$\begin{aligned} &\text{minimize} && |S| \\ &\text{s.t.} && \forall p \in P, \forall s \in S, R_p(s) \leq 1 \\ &&& \forall q \in Q, \forall s \in S, R_q(s) \leq 1 \end{aligned} \tag{6}$$

where S is the set of servers to host the VMs and $|S|$ is the size of S.

We can see that the classic NP-hard multidimensional bin packing problem is a special case of the MSVP problem, with the standard deviation of the demand of each stochastic resource set to 0. Thus, MSVP is also an NP-hard problem.

III. Max-Min Multidimensional Stochastic Bin Packing (M^{3}SBP) Algorithm

In this section, we present the Max-Min Multidimensional Stochastic Bin Packing (M^{3}SBP) algorithm that finds an approximation result for the MSVP problem. This algorithm is inspired by First Fit Decreasing (FFD) [6], and Dominant Resource First (DRF) [7]. FFD solves the classical bin packing problem, by first sorting the items in the decreasing order of their sizes and then packing larger items with higher priorities.

DRF tackles the fair resource allocation problem, where bins with multiple resources are shared by different users. The user's dominant share is defined as the maximum share that the user has been allocated of any resource. DRF seeks to maximize the minimum dominant share across all users. Both FFD and DRF yield higher server utilizations and thus fewer servers than other naïve bin packing algorithms do.

The basic idea of M^{3}SBP is as follows. For each newly powered-on server (new server in short), M^{3}SBP finds a set of VMs which can maximize its minimum TCR, so that the minimum resource utilization of each server is maximized and hence the total number of servers needed to power on is minimized. M^{3}SBP runs in iterative rounds, and in each round one VM is selected and placed into the new server. If there still exist VMs without placement when the new server is full, another server will be powered on. The iteration repeats until all VMs find their placement.

A. Algorithm Description

Initially, all VMs are marked as unplaced and added into the unplaced-VM set V. M^{3}SBP employs the set V and the resource capacities of server s as inputs. The algorithm runs in iterations, and in each iteration, one VM will be placed into s and removed from V. M^{3}SBP ends when V is empty and then outputs the mapping between VMs and servers as the result. Specifically, each iteration of M^{3}SBP can be further divided into two steps: Candidate Finding and Placement. Pseudo-codes are shown in Algorithm 1.

Step 1 ***Candidate Finding****:* The goal of Step 1 is to find a set of candidate VMs that can be placed in the new server s. For each VM v_i in V, M^{3}SBP calculates the TCRs of all resources of s as if v_i was placed in s. By (4) and (5), the TCRs of deterministic and stochastic resources can be computed as follows,

$$\forall p \in P, R_p(v_i, s) = \frac{D_p(v_i) + \sum_{j\in U} D_p(v_j)}{C_p(s)} \tag{7}$$

$$\forall q \in Q, R_q(v_i, s) = \frac{(\mu_q(v_i) + \sum_{j\in U} \mu_q(v_j)) + \beta\sqrt{\sigma_q^2(v_i) + \sum_{j\in U} \sigma_q^2(v_j)}}{C_q(s)} \tag{8}$$

The minimum and the maximum TCRs, $R_{min}(v_i, s)$ and $R_{max}(v_i, s)$, are then derived. If $R_{max}(v_i, s)$ is no greater than 1, it indicates that v_i can be placed in s. Then, the algorithm records $R_{min}(v_i, s)$ and adds v_i into set V' that stores the candidate VMs. If V' is empty after all VMs are tested, M^{3}SBP powers on another new server s and repeat Step 1. Otherwise, the algorithm continues with Step 2.

Step 2 ***Placement****:* In this step, M^{3}SBP compares the minimum TCR values of all candidate VMs in V' and chooses the VM v_F, which has the maximum value, as the selected VM. Then, M^{3}SBP places v_F into server s and adds it into set U. Finally, M^{3}SBP removes v_F from V.

B. Illustration Example

In this subsection, we give a simple example to illustrate the algorithm. Assume that each server has identical capacities of memory, CPU, power and network bandwidth, denoted by C_m, C_c, C_p and C_b, respectively. In this example, their values are set to be 8 GB, 4 GHz, 1000 W and 1 Gbps. We assume that there are 3 VMs, v_1, v_2 and v_3, to be placed into the servers. Each VM has deterministic demands on memory, CPU and power consumption resources, denoted by $D_m(v_i)$, $D_c(v_i)$ and $D_p(v_i)$, respectively, and has a stochastic demand on the network bandwidth resource, which follows a normal distribution $N(\mu_b(v_i), \sigma_b^2(v_i))$. Table I summaries all demands of the 3 VMs. The violation threshold α is set as 0.01%.

Initially, all VMs are added to unplaced-VM set V and

Algorithm 1 Max-Min Multidimensional Stochastic Bin Packing (M^{3}SBP)

```
1:  procedure M$^3$SBP($V$, $s$)
2:      while $V \neq \emptyset$ do
          // Step 1: Candidate Finding
3:          for all $v_i \in V$ do
4:              CalculateTCR($v_i$, s);
5:              if $R_{max}(v_i, s) \leq 1$ then
6:                  Record $R_{min}(v_i, s)$;
7:                  Add $v_i$ into $V'$;
8:              end if
9:          end for
10:         if $V' = \emptyset$ then
11:             Power on a new server $s$;
12:             Repeat Step 1;
13:         end if

          // Step 2: Placement
14:         $R(v_F, s) \leftarrow max\{\forall v_i \in V', R_{min}(v_i, s)\}$
15:         Add $v_F$ into $U$;
16:         Remove $v_F$ from $V$;
17:     end while
18: end procedure

19: procedure CalculateTCR($v_i$, $s$)
      // Deterministic Resources
20:     $\forall p \in P, R_p(v_i, s) \leftarrow \frac{D_p(v_i)+\sum_{j\in U} D_p(v_j)}{C_p(s)}$;

      // Stochastic Resource
21:     $\forall q \in Q, R_q(v_i, s) \leftarrow$
22:         $\frac{(\mu_q(v_i)+\sum_{j\in U} \mu_q(v_j))+\beta\sqrt{\sigma_q^2(v_i)+\sum_{j\in U} \sigma_q^2(v_j)}}{C_q(s)}$;

      // Derive Max and Min TCRs
23:     $R_{min}(v_i, s) \leftarrow min\{\forall r \in \{P, Q\}, R_r(v_i, s)\}$;
24:     $R_{max}(v_i, s) \leftarrow max\{\forall r \in \{P, Q\}, R_r(v_i, s)\}$;
25: end procedure
```

	$D_m(v_i)$	$D_c(v_i)$	$N(\mu_b, \sigma_b^2)$	$D_p(v_i)$
v_1	5.0 GB	1.0 GHz	(200, 2^2) Mbps	225 W
v_2	2.0 GB	1.5 GHz	(500, 5^2) Mbps	300 W
v_3	1.0 GB	2.5 GHz	(300, 3^2) Mbps	150 W

Table I
SUMMARY OF RESOURCE DEMANDS OF 3 VMS

the first server s_1 is powered on. In the first step, *Candidate Finding*, M^{3}SBP calculates the TCR for each of the 3 VMs as if the VM was already placed in the server. By (7) and (8), we have $R_m(v_1, s) = 0.625, R_c(v_1, s) = 0.25, R_p(v_1, s) = 0.225, R_b(v_1, s) = 0.208$. Then the maximum and minimum TCR of v_1 can be derived as $R_{max}(v_1, s) = 0.625$ and $R_{min}(v_1, s) = 0.208$. Since $R_{max}(v_1, s) \leq 1$, the server has enough resource for v_1. Thus v_1 is added to the candidate set V' and $R_{min}(v_1, s)$ is recorded. Repeat the same calculation for v_2 and v_3, we have $R_{max}(v_2, s) = 0.519, R_{min}(v_2, s) = 0.25$ and $R_{max}(v_3, s) = 0.625, R_{min}(v_3, s) = 0.125$. Thus both v_2 and v_3 are added to V' and their R_{min} are recorded. Then, in the second step *Placement*, M^{3}SBP compares the $R_{min}(v_i, s)$ of VMs in V' and finds v_2 with the maximum value. As a result, M^{3}SBP places v_2 into s_1 by adding v_2 into s_1's set of hosting VMs, U. Repeat the above two steps for v_1 and v_3. The final result is that, v_1 and v_2 are placed in s_1, while v_3 is placed in s_2.

C. Complexity Analysis

The complexity of the M^{3}SBP algorithm can be calculated in two phases. First, we count the number of execution times of **while** loop. Denote the size of V during **while** loop's kth execution as l_k. Initially, V contains all VMs. Thus $l_0 = n$, where n is the total amount of VMs. During each iteration of the **while** loop, exact one VM finds its placement and the size of V decreases by one. Thus, after n times of execution, V will be empty, i.e. $l_n = 0$. Therefore, the execution time of the **while** loop is the same as the number of VMs, n. In addition, l_k can be calculated by using the following equation, $l_k = n - k$.

Second, we calculate the complexity inside the **while** loop. In *Candidate Finding* phase, it takes $O(\log m)$ time to find the minimum and maximum TCRs for each VM in V, where m is the total number of resources. Since there are l_k unplaced VMs in the kth iteration, it needs a total of $O(l_k \times \log m)$ time to finish the candidate searching. In the *Placement* phase, it takes $O(\log l'_k)$ time to find the maximum of the minimum TCRs, where l'_k denotes the size of candidate VM set V' of the kth while loop iteration. In the worst-case scenario, l'_k can be as large as l_k. Then the time complexity of the *Placement* phase becomes $O(\log l_k)$. Thus, the time complexity inside the **while** loop is

$$O(l_k \times \log m) + O(\log l_k) \tag{9}$$

It is showed that $l = n - k$, and thus (9) becomes

$$O((n - k) \times \log m) + O(\log(n - k)) \tag{10}$$

Since $O(n-k) > O(\log(n-k))$ and typically $n >> m$, (10) can be simplified to $O(n - k)$. By combining the complexity of inside and outside of the **while** loop together, the total time complexity of M^{3}SBP is $O(n(n - k)) = O(n^2)$.

IV. Performance Evaluation

In this section, we present the performance evaluation results of the M^{3}SBP algorithm. We have conducted multiple simulations to evaluate different aspects of the performance, including the number of used servers, the guarantee of violation probability threshold and the algorithm effectiveness. Simulation configurations are described in Section IV-A. Results and analysis are shown in Section IV-B, IV-C and IV-D.

A. Simulation Configuration

In the simulations, each VM is assumed to have four types of resource demands: CPU, memory, power consumption and bandwidth. The first three are deterministic, while the bandwidth demand is stochastic and follows the normal distribution. We employ the resource-**D**emand to server-**C**apacity **R**atio (DCR) to identify the demand intensity of each resource. For each VM, we define the resource, which has the largest DCR value among all resources, as the *intensive resource*, and define others as the *non-intensive resources*. A data center may have VMs with different resource intensities, such as memory-intensive and CPU-intensive VMs. This kind of data centers demands the most resources from the multidimensional placement algorithm. It is because that if all VMs have the same intensive resource, the placement problem is then reduced to the classical one-dimensional VM placement problem. To simulate the mixed-intensity situation, we configure 4 groups of VMs each with a different intensive resource, and each group contains $1/4$ of all VMs. For each VM, the intensive resource randomly selects its DCR value from a higher range between 30% to 40%, and other non-intensive resources randomly select their DCR values from a lower range between 5% to 10%. For the stochastic bandwidth demand, the selected DCR value represents the ratio between the mean of the bandwidth demand and the server capacity. The bandwidth's standard deviation is set to be 0.5% of the mean demand by default. This percentage may change later in different simulations. Servers are assumed to have the same capacity of, 24 GHz (8 cores $\times$ 3.0 GHz/core) of CPU, 48 GB of memory, 2000 W of power supply and 1 Gbps of bandwidth. Then we can calculate the demands by multiplying the DCRs with the server's capacities. The total number of VMs is set to be 2000 and the SLA violation probability is set to be 0.01%.

The following example illustrates how resource demands of a *Memory-intensive* VM are configured. The memory resource's DCR is randomly selected between 30% to 40%, assuming 34%. CPU's and power consumption's DCR are randomly selected between 5% to 10%, assuming 6% and 7.5%, respectively. The DCR of bandwidth mean is also randomly selected from the lower range, assuming 5.5%. The standard deviation is set be 0.5% of the mean demand which in this case is $5.5\% \times 0.5\% = 0.0275\%$ of server's bandwidth capacity. Then, from server's capacities, we can derive VM's demands as 48 GB $\times$ 34% = 16.32 GB of memory, 24 GHz $\times$ 6% = 1.44 GHz of CPU, 2000 W $\times$ 7.5% = 150 W of power and 1 Gps $\times$ 5.5% = 55 Mbps mean with 0.275 Mbps standard deviation of bandwidth.

B. Number of Servers

In this subsection, we present the simulation results on the number of used servers. We compare M^{3}SBP with other bin packing algorithms, including both deterministic

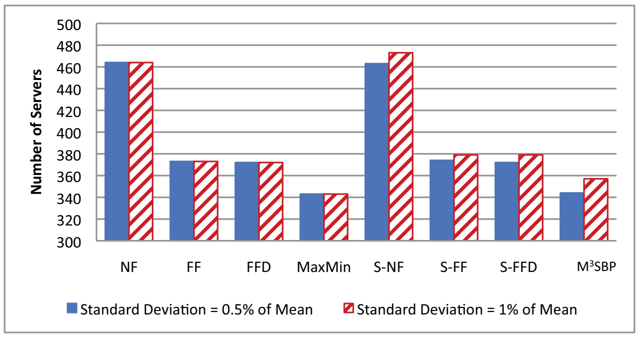

Figure 1. Number of servers used by different placement algorithms

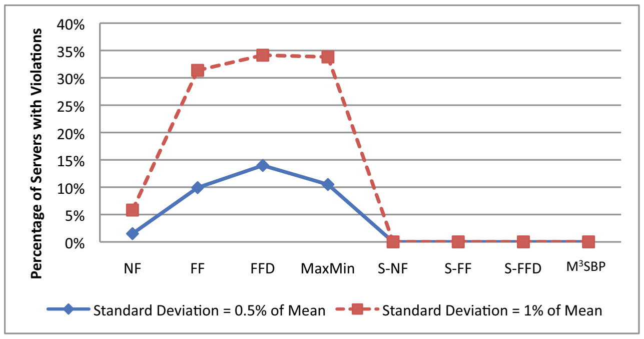

Figure 2. Percentage of servers that violate the target violation probability threshold

and stochastic algorithms. Deterministic algorithms, which compute placement only with fixed value demands, includes Next-Fit (NF) [6], First-Fit (FF) [6], FFD and Max-Min. NF places the VM into the current server if the demands of all resources are satisfied, or otherwise starts a new server. FF looks at all existing bins and places the VM into the lowest numbered server if it fits, or otherwise powers on a new server. FFD employs the same packing strategy as that of FF, except that, before the placement, FFD sums the DCRs of all resources of each VM and sorts the VMs in the decreasing order of their DCR summations. Max-Min has identical procedures as those of M^{3}SBP except that it treats the bandwidth as a deterministic resource and employ the mean bandwidth demand in the placement computing. Same as Max-Min, all other deterministic algorithms view bandwidth as a deterministic resource, and also employ the mean bandwidth demand when computing the placement.

Comparison stochastic algorithms include stochastic NF (S-NF), stochastic FF (S-FF) and stochastic FFD (S-FFD). These algorithms are modified based on the classic NF, FF and FFD algorithms, respectively. The modified algorithms calculate the equivalent TCR for stochastic resources by using the same equation (8) as in M^{3}SBP. We have conducted two sets of simulations. Both of them follow the same default configurations described in Section IV-A, except that each simulation uses a different standard deviation to mean ratio. The standard deviation represents the burst level of the VM's traffic. In the first set, the standard deviation is 0.5% of the mean, while in the second set the ratio is 1%.

Figure 1 illustrates the results. The solid and strip columns show results when the standard deviation is equal to 0.5% and 1% of the mean, respectively. Comparing the number of servers used by M^{3}SBP with that of all other stochastic algorithms, we can see that M^{3}SBP uses the fewest servers in both simulations. Then if we compare M^{3}SBP with the deterministic algorithms, we can find that M^{3}SBP still uses almost the least number of servers when VM's traffic burst level is low. When VM's traffic is more bursty, the number of servers used by all stochastic algorithms increase. This is reasonable since stochastic algorithms take the increasing burst into consideration, and allocate more server resources for VMs to prevent their network traffic from exceeding server's capacity. On the other hand, however, deterministic algorithms cannot detect the change of VM's burst level. This brings negative influence on server availability guarantees which is discussed in detail in Section IV-C.

C. *Server Availability Guarantee*

In this subsection, we evaluate how well the M^{3}SBP algorithm guarantees the availability requirement.

Stochastic algorithms compute placement based on the target violation probability threshold. After the VM placement, we verify whether the availability requirement is guaranteed by generating network traffic and comparing it with the server's bandwidth capacity. In each simulated second, we carry out the following procedures. First, we generate traffic for all VMs according to their stochastic parameters. Then, we calculate the total traffic amount for each server by adding up the traffic of all its hosted VMs. Lastly, we compare the total traffic amount with server's bandwidth capacity. If the former is larger, we say that in this second the bandwidth capacity is exceeded and the server's availability requirement is violated, and count the second as a violated second. At the end of the simulation, the real violation ratio is calculated by dividing the total number of simulated seconds by the total number of violated seconds. Then, if the real violation ratio is higher than the target violation probability threshold, we say that this server failed to guarantee the availability requirement. The target violation probability threshold is set to 0.01% and the simulation emulates 7 days of network traffic.

Following these procedures, we evaluate the placement results of the previous simulations. Figure 2 shows the percentage of servers which have violated the target violation probability threshold. We can see that all deterministic algorithms have violated servers. Moreover, when the traffic is more bursty, the number of violated servers of deterministic algorithms rises up dramatically. Both FF, FFD and Max-Min have more than 30% of servers violates the target violation probability threshold when standard deviation is equal to 1% of the mean. In contrast, none of the stochastic algorithms, including M^{3}SBP, has violated servers. This demonstrates that M^{3}SBP can guarantee the server's availability well.

D. *Effectiveness*

In this subsection, we evaluate the effectiveness of M^{3}SBP algorithm. Effectiveness of a placement algorithm is defined as follows: 1) if two algorithms consume the same number of

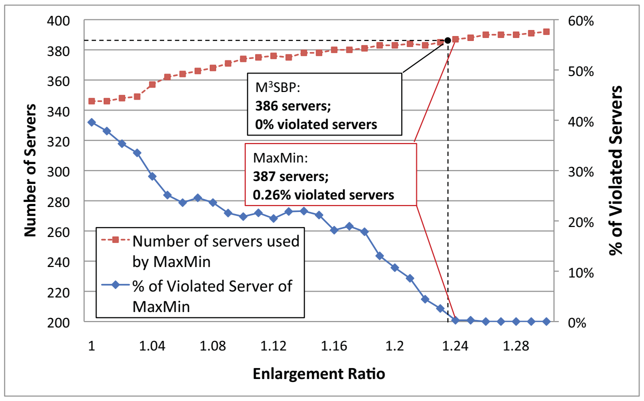

Figure 3. Number of servers and percentage of violated servers of Max-Min when gradually increasing the enlargement ratio from 1.00 to 1.30.

servers, the more effective one possesses less percentage of server violations; or 2) if two algorithms all have zero server violation (i.e. all guarantee the violation probability threshold), the more effective one uses a smaller number of servers.

By applying these definitions to the results in Figure 1 and 2, we find that M^{3}SBP is more effective than all other stochastic algorithms. This is because it is compliant with the second part of the effectiveness definition that M^{3}SBP requires the fewest number of servers while obtaining zero server violation.

However, it is not straightforward to compare the effectiveness between M^{3}SBP and deterministic algorithms, which possess a higher violation ratio but need fewer servers. One solution is to gradually enlarge the demand of stochastic resources used by deterministic algorithms in placement computing, so that the deterministic algorithms will use more servers and generate fewer violations. In our simulation, we gradually increase the demand of bandwidth resource in the deterministic Max-Min algorithm from the original value to 1.30 times of it. From Figure 3, we can see that, as expected, when the bandwidth demand increases, the number of used servers of Max-Min also increases and the percentage of violated server decreases. When the bandwidth demand is enlarged to 1.24 times of the original value, Max-Min uses more servers than M^{3}SBP does. However, there are still 0.26% of servers in Max-Min violates the target violation probability threshold. Thus, M^{3}SBP finds better placement in shorter amount of time than Max-Min. In other words, M^{3}SBP is more effective than Max-Min. Due to space limitations, the results of comparisons between M^{3}SBP and other deterministic algorithms are omitted. All comparisons lead to the same conclusion. Therefore, we can say that M^{3}SBP is more effective than all benchmark algorithms.

V. Conclusions

In this paper, we have studied VM placement in data centers when the VMs have multiple demands for various resources and some of them are stochastic resources. We formulate the Multidimensional Stochastic VM Placement (MSVP) problem. Because MSVP is NP-hard, we propose a fast algorithm named Max-Min Multidimensional Stochastic Bin Packing (M^{3}SBP), to calculate the VM placement for large scale data centers. Numerical simulations are conducted to evaluate M^{3}SBP's performance. In the simulations, each VM requests four types of resources, CPU, memory, power consumption and bandwidth. Among those resources, the first three are deterministic while bandwidth is stochastic, and we employ the normal distribution to model the bandwidth demand. The results show that M^{3}SBP uses fewer servers than other benchmark bin packing algorithms, while guaranteeing the server's availability requirement. In addition, the results also demonstrate that M^{3}SBP is more effective in finding the desired placement result than other benchmark algorithms.

References

[1] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter," *ACM ISCA*, Saint-Malo, France, Jun. 2010.

[2] M. Al-Fares, A. Loukissas and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM*, Seattle, WA, Aug. 2008.

[3] T. Benson, A. Akella and D. A. Maltz, "Network traffic characteristics of data centers in the wild," *ACM IMC*, Melbourne, Australia, Nov. 2010.

[4] N. Bobroff, A. Kochut and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," *IFIP/IEEE Integrated Network Management*, Munich, Germany, May 2007.

[5] M. Chen, H. Zhang, Y. Y. Su, X. Wang, G. Jiang and K. Yoshihira "Effective VM sizing in virtualized data centers," *IFIP/IEEE Integrated Network Management (IM)*, Dublin, Ireland, May 2011.

[6] E. G. Coffman, Jr., G. Galambos, S. Martello and D. Vigo "Bin packing approximation algorithm: combinatiorial analysis," *Handbook of Combinatorial Optimization*, D.-Z. Du and P. Pardalos, Eds. Kluwer Academic Publishers, 1998.

[7] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types," *USENIX NSDI*, Boston, MA, Mar. 2011.

[8] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, P. Lahiri, D. A. Maltz, P. Patel and S. Sengupta, "VL2: a scalable and flexible data center network," *ACM SIGCOMM*, Barcelona, Spain, Aug. 2009.

[9] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. Mckeown, "Elastictree: saving energy in data centernetworks," *USENIX NSDI*, San Jose, CA, Apr. 2010.

[10] C. Hyser, B. McKee, R. Gardner and B. J. Watson "Autonomic virtual machine placement in the data center," *HP Technical Report HPL-2007-189*, Feb. 2008.

[11] S. Kandula, S. Sengupta, A. Greenberg, P. Patel and R. Chaiken, "The nature of datacenter traffic: measurements and analysis," *ACM IMC*, Chicago, IL, Nov. 2009.

[12] A. Kansal, F. Zhao, J. Liu and N. Kothari, "Virtual machine power metering and provisioning," *ACM symposium on Cloud computing*, Indianapolis, IN, Jun. 2010.

[13] J. Kleinberg, Y. Rabani and E. Tardos "Allocating bandwidth for bursty connections," *SIAM Journal on Computing*, vol. 30, no. 1, pp. 191-217, 2000.

[14] B. Krishnan, H. Amur, A. Gavrilovska and K. Schwan, "VM power metering: feasibility and challenges," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 3, pp. 56-60, 2011.

[15] X. Meng, V. Pappas and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," *IEEE Infocom*, San Diego, CA, Mar. 2010.

[16] B. Sotomayor, R.S. Montero, I.M. Llorente and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14-22, 2009.

[17] H.Van and F. Tran, "Autonomic virtual resource management for service hosting platforms," *IEEE CLOUD*, Vancouver, Canada, May 2009.

[18] M. Wang, X. Meng and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," *IEEE Infocom*, Shanghai, China, Apr. 2011.

[19] J. Xu and J.A.B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," *IEEE GreenCom*, Hangzhou, China, Dec. 2010.

[20] J. Xu and J.A.B. Fortes, "A multi-objective approach to virtual machine management in datacenters," *ACM ICAC*, Karlsruhe, Germany, Jun. 2011.