2018 IEEE International Conference on Cloud Engineering

An Online Virtual Machine Placement Algorithm in an Over-Committed Cloud

Siqi Ji, Ming Da Li, Niannian Ji, Baochun Li
University of Toronto

***Abstract*—Cloud computing provides users and enterprises shared pools of resources to store and process their data. Virtualization is one of the technologies used in cloud computing to separate the underlying hardware resources to create virtual machines (VMs). VM placement is the process of choosing the best PM for newly created VMs. Since different users have a variety of resource requirements, how to deploy VMs becomes a challenging problem in cloud management. Most of the existing works only consider maximizing the resource utilization of PMs without taking the overcommitment issue into account, which can cause PM overloading and degrade VM performance. In this paper, we propose an algorithm, called Min-DIFF, that can balance the usage of resources along multiple dimensions and reduce the risk of PM overloading in a threshold-based placement strategy. To better evaluate the performance of Min-DIFF, our simulations are driven by both real-world workload traces and datasets we generated. Extensive simulation results show that Min-DIFF achieves a significant improvement over the existing approaches in balancing the utilization of multi-dimensional resources and reducing the risk of PM overloading.**

I. Introduction

Cloud computing enables cheap and easy access to shared pools of computational resources, which provides users and enterprises with capabilities to store and process their data efficiently and reliably. Virtualization [1] is an important technology of cloud computing, it separates physical hardware resources to create virtual machines (VMs) with dedicated resources. Virtual machine (VM) acts like a real computer with an operating system. Through virtualization, it is possible to run multiple VMs on the same physical machine (PM) at the same time while increasing efficiency and utilization of hardware resources.

VM placement is the process of selecting the most appropriate PM for deploying VMs. Since different users have diverse resource requirements for VMs and resources on each PM are limited, improper VM placement can cause unbalanced resource utilization (overloaded in some resources but underutilized in others). Such resource fragmentation requires extra PMs and wastes more resources. Therefore, it is crucial to balance PM resources along multiple dimensions (i.e., CPU, memory, storage, network bandwidth, etc.) during VM placement and minimize the number of activated PMs.

Existing works [2]–[4] indicated that VMs tend to utilize less resources than reserved capacities, which causes substantial resource wastage. **Resource Overcommitment** is widely used for solving this wastage problem by allocating more resources to VMs than they actually have. For example, a PM has 64GB memory and it is sold as 128GB memory, we say that the overcommit ratio of this PM is 2. It could be problematic if we only consider the problem of minimizing the number of activated PMs and reducing resource fragmentation in the over-committed cloud. While resource overcommitment increases resource utilization and benefits the cloud service providers, the risk of provider-induced overload is also increased. This happens when users actually demand enough of resources such that collectively they exhaust all available physical resources, which leads to VM performance degradation and could drive off users. As such, it is of interest to consider resource overcommitment during VM placement, such that the risk of service degradation is reduced while resource utilization is enhanced.

There has been a significant amount of work on VM placement. Some existing works [5]–[8] only considered one resource dimension when they deploy VM placement, these approaches overlooked the multi-dimensional nature of the problem and did not balance resources along different dimensions. Other works solved the problem in an offline optimization manner [9]–[13]: resource requirements are known in advance. These approaches did not adequately reflect the true nature of VM requests, with unpredictable arrivals and departures. Other approaches like [14], [15] studied the VM placement problem with the goal of achieving balanced resource utilization along multiple dimensions and minimizing the number of physical machines activated, which has very close objectives with our work. However, they did not raise the PM overloading problem in an over-committed cloud as their concern. The overcommitment issue is typically considered in VM migration [2], [3] while there were very few works that took this problem into account in the process of VM placement.

In this work, we solve the VM placement problem with the objective of balancing the use of resources along multiple dimensions and reducing the risk of PM overloading while considering the over-committed issue. Our contributions are the following:

1) We consider multiple dimensions of resources in VM placement, heterogeneous settings regarding PM resource capacity configurations are generated in our simulations, which realistically resembles modern cloud datacenters.
2) We consider the model that VM deployment requests arrive and depart dynamically, in an online manner where resource requirements are not known beforehand.

0-7695-6371-6/18/31.00 ©2018 IEEE
DOI 10.1109/IC2E.2018.00033

This dynamic nature produces a realistic scenario.

3) We propose a threshold-based algorithm called Min-DIFF, which considers resource overcommitment in an effort to reduce the risk of service degradation. Besides, Min-DIFF obtains a more balanced use of resources along different dimensions than related works, which reduces resource fragmentation effectively.
4) Our simulations are driven by both real-world traces and datasets we generated, to evaluate the effectiveness of Min-DIFF under a wider spectrum of conditions. Our simulation results show that Min-DIFF has better performance in two aspects: First, Min-DIFF uses fewer PMs and achieves lower resource fragmentation than other approaches if we do not take the overcommitment issue into account. Second, Min-DIFF has a lower risk of PM overloading compared to other approaches in an over-committed cloud.

This paper is organized as follows. In Section II, we discuss the related works. In Section III, we motivate our work by using a simple example. In Section IV, we discuss how to deploy VM requests and illustrate details of Min-DIFF. In Section V, we present our simulation setup and show the simulation results by using different datasets. Finally, we conclude the paper in Section VI.

II. Related Works

The VM placement problem has been extensively studied in the field of Cloud Computing. VM placement can be formulated as a bin packing problem which is proved to be NP-hard [16]. Monil *et al.* [5] proposed a multi-pass Best Fit Decreasing VM placement algorithm that achieved a balance between energy consumption and quality of service. It did not consider multiple dimensions of resources, rather focusing on CPU utilization only. Wang *et al.* [6] formulated the VM placement problem into a Stochastic Bin Packing problem, which used random variables to characterize the uncertain future bandwidth usage for each VM. However, within their probabilistic characterization, only bandwidth was considered. Zhang *et al.* [7] formulated the problem into a constrained minimum k-cut problem, under the constraint of VM performance. Nevertheless, the solution was based on energy consumption as a singular criterion, instead of the specific PM resources. These approaches did not consider to balance the use of multiple resources, which could make one resource in high utilization but other resources under-utilized.

Other approaches solved the VM placement problem in an offline manner, they assumed that the VM requests are known beforehand. Beloglazov *et al.* [13] modified the Best Fit Decreasing algorithm for deploying VMs, it sorted all VM requests in decreasing order of their CPU requirements. Most related works like [8], [17]–[21] formulated the VM placement problem as an optimization problem which is not suitable for dynamic VM requests. Rampersaud *et al.* [9] designed an approximation algorithm that took multi-dimensional resources into account for maximizing profit derived from hosting VMs. However, in real scenarios, requests are coming at different time slots, and it is hard to know VM requests in advance.

There are a lot of related works that proposed approaches for the online VM placement [14], [15], [22]–[25] with the consideration of multi-dimensional resources, but few considered the overcommitment issue in VM placement. Max-BRU[14] considered multiple resource types, focused on maximizing resource utilization and meanwhile balanced the use of multiple types of resources. EAGLE [15] proposed a multi-dimensional space partition model to balance resource utilization along different dimensions while minimizing total energy consumed by running PMs. Overcommitment is mainly considered in VM migration [2], [3]. In this paper, we use a threshold-based idea to efficiently reduce the risk of PM overloading caused by overcommitment, which also reduces overhead in migration.

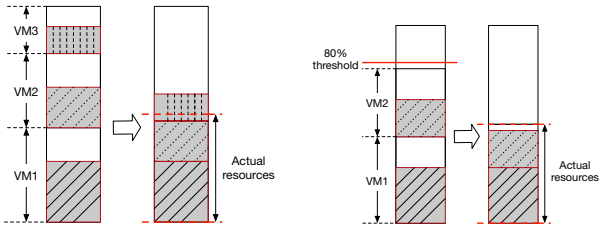
III. Motivation Example

Previous works on VM placement try to maximize the utilization of PMs and pack VMs as tightly as possible. However, resource overcommitment may cause PM overloading when total resources utilized by VMs do exceed the PM's actual capacities. We use an example to illustrate PM overloading, and we only consider memory in this example for simplification. As shown in Figure 1(a), considering there are three VM requests: VM1, VM2, and VM3, each requires memory of 32GB, 24GB, and 16GB, respectively. The memory capacity of the PM is 36GB, and it is sold as 72GB with the overcommit ratio of 2. If we pack all VMs in this PM and these VMs utilize 60% of their required resources, then this PM will be overloaded. Overloading can substantially degrade VM performance, and some VMs will not get their fair share of resources. A better approach is to set an 80% threshold of 72GB memory, total resources of VMs placed in the PM can not exceed this threshold. As we can see from Figure 1(b), only VM1 and VM2 are placed in this PM, VM3 will be placed in another PM. In this way, resources utilized by VMs will not exceed the PM's capacities, all VMs can work well and get good performance. Related works only consider the overcommitment issue in VM migration. Nevertheless, migrating VMs in overloaded PMs can cause extra overhead and increase bandwidth usage. We save overhead and network bandwidth by considering overcommitment in the VM initial placement. We will discuss the setting of the threshold in Section IV.

IV. Min-DIFF: A threshold-based online VM initial placement algorithm

In this section, we present our proposed threshold-based online VM placement algorithm Min-DIFF, which can reduce resource fragmentation efficiently with the consideration of the overcommitment issue.

Figure 2 gives a sketch of the threshold-based idea. Grey squares represent different VMs. We deploy VMs by using one of the two strategies shown in Figure 2. In Strategy 1, we select the most appropriate PM to place VMs under the threshold. If we can not find space under the threshold, then we use Strategy



(a) Packing VMs as tightly as possible. (b) Setting a 80% threshold for placement.

Fig. 1: A motivation example.

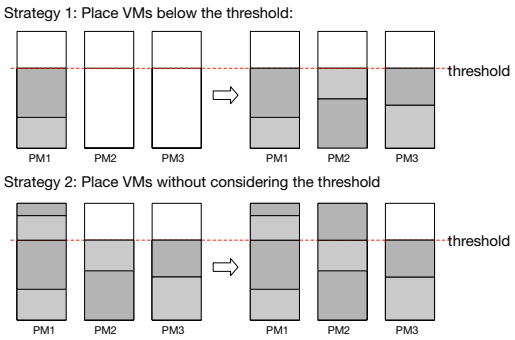


Fig. 2: A sketch of the threshold-based idea.

2 to place VMs without considering the threshold. Strategy 1 always has the highest priority. Table I presents variables we use in this paper and their definitions. A VM request i can be denoted as a tuple $\{a_i, du_i, \mathrm{VM}_i^d\}$.

A. Resource Threshold

Typically, to guarantee performance for most VMs and reduce the risk of PM overloading, some providers do not expect the utilization of over-committed PMs is higher than a specific percentage [26], and we call this percentage **a warning line** in this paper. For example, if a PM is sold as 72GB memory with the overcommit ratio of 2 and the warning line of memory is 80%, then the total memory of VMs in this PM should not exceed 57.6GB. The warning line is considered when we set the resource threshold in Min-DIFF.

On the other hand, to reduce resource fragmentation, we reserve enough space for large VMs above the threshold. Otherwise, if we can not find enough space below the threshold and need to use Strategy 2, a large VM can not be placed in the PM, which will cause large resource fragmentation. Therefore, based on the warning line w^d and the largest VM requirement L^d, the threshold Th_j^d is defined as:

$$\mathrm{Th}_j^d = \min\left\{\frac{PM_j^d - L^d}{\mathrm{PM}_j^d}, w^d\right\}. \tag{1}$$

B. Find the Best PM for Single VM Request

If there is one VM request at each time slot, we find the best PM for this request. We will illustrate how to choose the best PM in this section.

Variables	Meaning
D	The number of resource dimensions.
d	The index of resource dimensions, $d = 1, ..., \mathrm{D}$.
j	The index of PMs.
i	The index of VM requests.
U_j^d	Used resource along dimension d of the jth PM.
PM_j^d	Total resource along dimension d of the jth PM.
VM_i^d	Resource requirement along dimension d of the ith VM request.
a_i	Arrival time of the ith VM request.
du_i	The duration of the ith VM request.
w^d	The warning line of PMs along dimension d.
L^d	The largest VM resource requirement along dimension d.
Th_j^d	The threshold along dimension d of the jth PM.
RF_j	Resource fragmentation of the jth PM.
NR_j^d	The normalized residual resource along dimension d of the jth PM.
NU_j^d	The normalized used resource along dimension d of the jth PM.

TABLE I: Variables used in this paper.

In order to place more future VM requests, we try to balance resources along multiple dimensions left on each PM. Otherwise, if residual resources along one dimension become unavailable, resources along other dimensions are wasted. Such resource fragmentation will prevent future VM requests and waste resources.

Considering a datacenter provides a pool of resources such as CPU, memory, network bandwidth and storage, we deal with multiple resource types in VM placement. Inspired by previous works [17], [27], [28], we extend the resource wastage model to multiple dimensions, which is not specific to two dimensions. The following equation is used to measure the resource fragmentation of a PM:

$$RF_j = \frac{\sum\limits_{p,p\neq m}\left(NR_j^p - NR_j^m\right)}{\sum\limits_{d=1}^{D} NU_j^d}, \tag{2}$$

where RF_j represents the resource fragmentation of the jth PM. NR_j^m indicates the smallest normalized residual resource. NR_j^p denotes the normalized residual resource along dimension p, and p does not equal to m. Therefore the numerator calculates the sum of differences between the smallest normalized residual resource and the others. The denominator represents the sum of the normalized used resource along each dimension. When computing the resource fragmentation, the residual resource as well as the used resource on the PM is normalized by the PM's overall capacity. It is evident that the more used resource and more balanced residual resource along different dimensions, the resource fragmentation value is smaller.

To reduce resource fragmentation and pack VMs tightly, we propose an efficient algorithm based on the resource fragmentation Equation (2). Intuitively, an idea is to choose the PM that has the minimal resource fragmentation value after a VM is placed. However, this idea is problematic for

utilized PMs. For example, considering there are two utilized PMs and a VM needs to be placed. If we place the VM in PM_1, $RF_1 = 0.3$; if we place the VM in PM_2, $RF_2 = 0.2$, then PM_2 is selected. Nevertheless, this idea does not consider the resource fragmentation value before the VM is placed. if $RF_1 = 0.6$ and $RF_2 = 0.1$ before the VM is placed, deploying the VM in PM_2 actually makes the resource fragmentation value higher.

A proper approach is: for non-empty PMs, we deploy a VM in the PM that can maximize the resource fragmentation reduction. Before a VM is placed, the normalized used resource is:

$$NU_j^d\_bef = \frac{U_j^d}{PM_j^d}, \mathrm{d} = 1, ..., \mathrm{D}. \tag{3}$$

The normalized residual resource is:

$$NR_j^d\_bef = 1 - \mathrm{NU}_j^d\_bef, \mathrm{d} = 1, ..., \mathrm{D}. \tag{4}$$

The smallest value among $NR_j^d\_bef$ is $NR_j^m\_bef$, then the initial resource fragmentation is:

$$RF_j\_bef = \frac{\sum\limits_{p,p\neq m} \left(NR_j^p\_bef - NR_j^m\_bef\right)}{\sum\limits_{d=1}^{D} NU_j^d\_bef}. \tag{5}$$

Similarly, after a VM is placed, the normalized used resource is:

$$NU_j^d\_aft = \frac{U_j^d + VM_j^d}{PM_j^d}, \mathrm{d} = 1, ..., \mathrm{D}. \tag{6}$$

The normalized residual resource is:

$$NR_j^d\_aft = 1 - \mathrm{NU}_j^d\_aft, \mathrm{d} = 1, ..., \mathrm{D}. \tag{7}$$

We still find the smallest value among $NR_j^d\_aft$, which is denoted as $NR_j^m\_aft$. Therefore the resource fragmentation value after deploying a VM is:

$$RF_j\_aft = \frac{\sum\limits_{p,p\neq m} \left(NR_j^p\_aft - NR_j^m\_aft\right)}{\sum\limits_{d=1}^{D} NU_j^d\_aft}. \tag{8}$$

Then we calculate the difference of resource fragmentation before a VM is placed and resource fragmentation after a VM is placed, which is:

$$\delta_{RF_j} = RF_j\_bef - RF_j\_aft. \tag{9}$$

For non-empty PMs, we choose the PM that has the largest δ_{RF_j}. For empty PMs, we select the PM with the most balanced utilization along resource dimensions after deploying a VM. Thus, we calculate the sum of differences between the smallest normalized residual resource NR_j^m and the others NR_j^p, choose the PM that has the smallest RF_j_empty.

$$RF_j\_empty = \sum_{p,p\neq m} \left(NR_j^p - NR_j^m\right). \tag{10}$$

To pack VMs tightly, we first find the most appropriate PM among non-empty PMs, if there is no available utilized PM, we choose the best PM among empty PMs.

C. VM Selection for Multiple VM Requests

If there are multiple VM requests at each time slot, we do the placement as follows. When resources are available on a PM, we choose the set of VM requests at the current time slot whose resource requirements can be accommodated on that PM. If the PM is utilized, we compute δ_{RF_j} to the PM for each VM request in this set. The request with the largest δ_{RF_j} will be placed in that PM. If the PM is empty, we compute RF_j_empty and choose the VM with the smallest value to place in that PM. This process is repeated recursively until the PM can not accommodate any VM requests in the current time slot. Then we go to the next PM to place other VM requests.

D. Details of Min-DIFF

Min-DIFF is illustrated by Algorithm 1. First of all, we calculate the threshold for each PM based on Equation (1) (line 2-4). If there are multiple requests at the time slot, we use Strategy 1 to place current VMs below the threshold by calling the function PLACECURVMSBLWTH($current\_VMs$). If there is not enough space for all current VMs, we use Strategy 2 where function PLACECURVMS($current\_VMs$) is called. These two functions use the algorithm we present in Section IV-B. If there is only one VM request at the time slot, we choose the best PM for this VM and use the algorithm in Section IV-C. Function FINDBESTPM(v, PMs) uses Strategy 2, which is similar to Strategy 1: FINDBESTPMBLWTH(v, PMs).

V. PERFORMANCE EVALUATION

Now we are ready to evaluate the performance of Min-DIFF through simulations. In this section, we present the simulation setup and evaluation results.

A. Simulation Setup

We evaluate the performance of Min-DIFF by using four types of datasets. For the first dataset, we consider the resource requirement of VMs to be equal to the standard instances from general purpose applications provided by Amazon EC2. Table II presents the seven types of T2 instances we use in our simulations. We set D=3 and use our 3-dimensional VM placement scheme for this dataset.

For the second and third datasets, we generate the VM requests that follow the uniform distribution and the normal distribution as Hieu *et al.* [14]. Table III shows the resource requirements of each dimension of the VM requests, where $\mathcal{U}(a, \mathrm{b})$ denotes the uniform distribution and $\mathcal{N}(\mu, \sigma)$ denotes the normal distribution. We set D=4 and use our 4-dimensional VM placement scheme for the second and third datasets.

The last one is the real-world workload trace GWA-T-12 Bitbrains [29] which contains the performance metrics of VMs from a distributed datacenter from Bitbrains. Bitbrains is a service provider that hosts applications used in financial fields. We extract CPU cores and Memory requested of VMs from the traces and set the number of dimensions D=2 for VM placement.

Algorithm 1 Min-DIFF algorithm

```
function VMPLACEMENT(VMs, PMs)
    for m in PMs do:
        calculate threshold based on Equation (1)
    end for
    while VMs ≠ ∅ do
        current_VMs = requests at the current time slot
        remove current_VMs from VMs
        if length(current_VMs) > 1 then:
            flag, current_VMs
            =PLACECURVMSBLWTH(current_VMs)
            if flag is False then:
                PLACECURVMS(current_VMs)
            end if
        else if length(current_VMs) = 1 then:
            for v in current_VMs do
                BestPM = FINDBESTPMBLWTH(v, PMs)
                if BestPM is not None then:
                    Place VM v on BestPM
                    Remove VM v from current_VMs
                    continue
                end if
                BestPM = FINDBESTPM(v, PMs)
                if BestPM is not None then:
                    Place VM v on BestPM
                    Remove VM v from current_VMs
                end if
            end for
        end if
    end while
end function
```

VM Instances	CPU	Memory(GB)	Bandwidth(MBit/s)
t2.nano	1	0.5	30
t2.micro	1	1	70
t2.small	1	2	200
t2.medium	2	4	300
t2.large	2	8	500
t2.xlarge	4	16	800
t2.2xlarge	8	32	1024

TABLE II: Amazon EC2 VM instances used in the first dataset.

Typically, cloud environments are not homogeneous and they are constructed from different types of machines [30]. To better resemble the real-world cloud, we generate heterogeneous PMs based on the configurations of machines shown in Reiss *et al.* [30] for the first dataset and the real-world workload trace. For the second and third datasets, we generate five types of PMs, each with the resource capacity of 200, 250, 300, 350 and 400 along all dimensions.

We compare Min-DIFF with the following schemes for VM placement: First Fit algorithm, the balanced algorithm EAGLE in [15] and Max-BRU algorithm in [14]. In our simulations, to better compare the performance of different algorithms, we make the durations of all VM requests infinitely long, which means that once they are placed, they will not be deleted.

VM Instances	CPU capacity (GHz)	Memory (GB)	Bandwidth (Gbps)	Storage (GB)
$\mathcal{U}(a, \mathrm{b})$	$\mathcal{U}(20, 80)$	$\mathcal{U}(20, 80)$	$\mathcal{U}(20, 80)$	$\mathcal{U}(20, 80)$
$\mathcal{N}(\mu, \sigma)$	$\mathcal{N}(50, 12)$	$\mathcal{N}(50, 12)$	$\mathcal{N}(50, 12)$	$\mathcal{N}(50, 12)$

TABLE III: Resource requirements used in the second and third datasets.

B. Simulation results: threshold 100%

First, we compare Min-DIFF with First Fit, EAGLE and Max-BRU by setting the threshold as 100%, which means that we aim at packing VMs as tightly as possible when resources of PMs are not over-committed. We consider the following performance metrics:

- The number of utilized PMs: K.
- The average resource fragmentation of all utilized PMs:

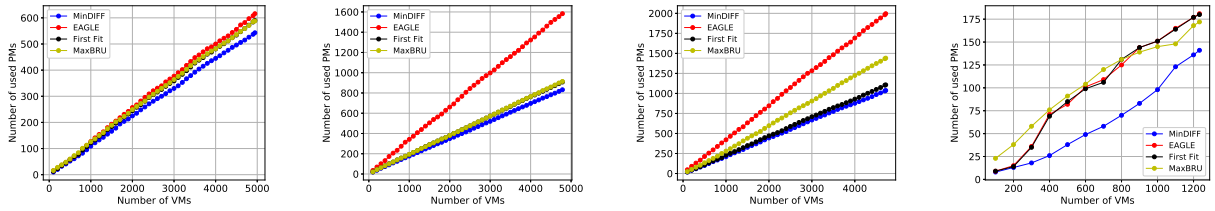
$$\overline{RF} = \frac{1}{K} \sum_{j=1}^{K} RF_j \tag{11}$$

1) Simulation results: Figure 3 shows the number of used PMs. As we can see from the figure, Min-DIFF uses fewer PMs than EAGLE, First Fit and Max-BRU. Figure 4 gives comparison results of the average resource fragmentation. Min-DIFF achieves the lowest resource fragmentation, which means that Min-DIFF has less resource wastage and obtains a more balanced resource utilization along different dimensions. EAGLE and Max-BRU do not perform well in the heterogeneous setting since they just open a new PM if they can not find available resources among the utilized PMs, Min-DIFF uses Equation (10) which works for deploying VMs in the empty set of PMs.

C. Simulation results: threshold is smaller than 100%

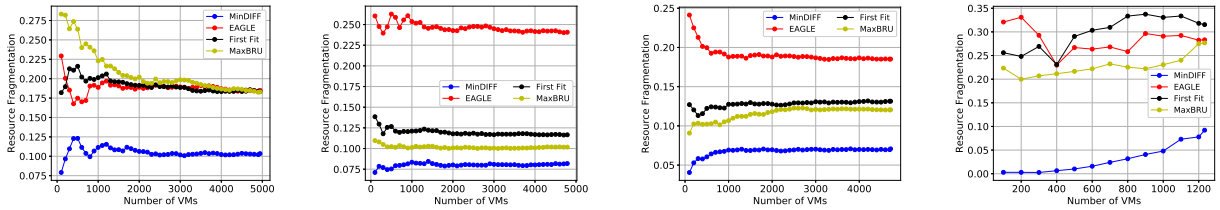
In this section, we will compare Min-DIFF with other approaches when the threshold is smaller than 100%, which means that providers do not prefer too high utilization of resources because of the over-committed issue, then we need to use the threshold-based idea to reduce the risk of PM overloading. We set D = 2 and use our two-dimensional placement algorithm for deploying VMs. The dataset used in this section is Amazon EC2.

Considering the warning line is 80% along each resource dimension, we do simulations when there are enough PMs for VM requests. Figure 5 shows comparison results of the number of activated PMs and the average resource fragmentation. In this scenario, there are enough PMs for all VM requests so that all VMs will be placed under the threshold. Since we place VMs below the threshold by using Strategy 1, Min-DIFF uses more PMs than other baselines. Although Min-DIFF uses more PMs, it can be seen from Figure 5(b), Min-DIFF effectively achieves the most balanced use of resources along different dimensions. Besides, other baselines do not consider PM overloading, which results in more PMs have the risk of



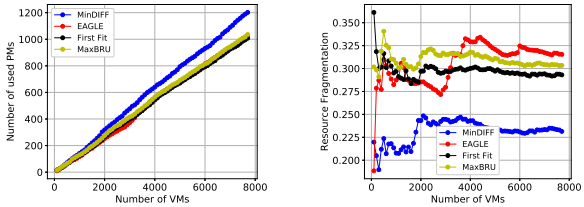
(a) (b) (c) (d)

Fig. 3: Number of used PMs for different datasets: (a) Amazon EC2. (b) Uniform distribution. (c) Normal distribution. (d) Real-world workload trace.



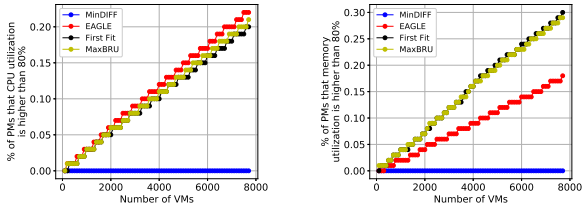
(a) (b) (c) (d)

Fig. 4: The average resource fragmentation of all used PMs for different datasets: (a) Amazon EC2. (b) Uniform distribution. (c) Normal distribution. (d) Real-world workload trace.



(a) (b)

Fig. 5: Comparison results: (a) Number of PMs. (b) The average resource fragmentation of all used PMs.



(a) (b)

Fig. 6: The percentage of PMs that resource utilization is higher than 80% (Over-committed resources are included): (a) CPU. (b) Memory.

overloading than Min-DIFF. As shown in Figure 6, Min-DIFF does not have PMs that utilizes resources higher than 80%, which can substantially reduce the risk of PM overloading. For other baselines, the number of PMs that obtain at least 80% utilization of resources increases as the number of requests increases. PM overloading can substantially degrade VM performance, Min-DIFF reduces such risk efficiently.

VI. Conclusion

In this paper, we propose an online algorithm called Min-DIFF which makes a tradeoff between minimizing the number of activated PMs and reducing the risk of PM overloading in an over-committed cloud. Besides, Min-DIFF achieves a more balanced use of resources along multiple resource dimensions, which significantly reduces resource fragmentation. To better resemble the real-world scenario, we consider VM requests that come at different time slots, heterogeneous settings of PMs are considered in our simulations. Simulation results demonstrate that our proposed algorithm Min-DIFF achieves better performance than other schemes in related works.

References

[1] R. P. Goldberg, "Survey of Virtual Machine Research," *Computer*, vol. 7, no. 6, pp. 34–45, 1974.

[2] X. Zhang, Z.-Y. Shae, S. Zheng, and H. Jamjoom, "Virtual Machine Migration in an Over-Committed Cloud," in *Proc. IEEE Network Operations and Management Symposium (NOMS)*, 2012.

[3] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Efficient Datacenter Resource Utilization Through Cloud Resource Overcommitment," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2015.

[4] L. Tomás and J. Tordsson, "Improving Cloud Infrastructure Utilization Through Overbooking," in *Proc. ACM Cloud and Autonomic Computing conference*, 2013.

[5] M. A. H. Monil and A. D. Malony, "QoS-Aware Virtual Machine Consolidation in Cloud Datacenter," in *Proc. IEEE International Conference on Cloud Engineering (IC2E)*, 2017.

[6] M. Wang, X. Meng, and L. Zhang, "Consolidating Virtual Machines with Dynamic Bandwidth Demand in Data Centers," in *Proc. IEEE INFOCOM*, 2011.

[7] X. Zhang, Y. Zhao, S. Guo, and Y. Li, "Performance-Aware Energy-efficient Virtual Machine Placement in Cloud Data Center," in *Proc. IEEE International Conference on Communications (ICC)*, 2017.

[8] H. Yanagisawa, T. Osogami, and R. Raymond, "Dependable Virtual Machine Allocation," in *Proc. IEEE INFOCOM*, 2013.

[9] S. Rampersaud and D. Grosu, "A Multi-Resource Sharing-Aware Approximation Algorithm for Virtual Machine Maximization," in *Proc. IEEE International Conference on Cloud Engineering (IC2E)*, 2015.

[10] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible, "Improving Performance and Availability of Services Hosted on Iaas Clouds with Structural Constraint-Aware Virtual Machine Placement," in *Proc. IEEE International Conference on Services Computing (SCC)*, 2011.

[11] F. Machida, M. Kawato, and Y. Maeno, "Redundant Virtual Machine Placement for Fault-Tolerant Consolidated Server Clusters," in *Proc. IEEE Network Operations and Management Symposium (NOMS)*, 2010.

[12] C. C. T. Mark, D. Niyato, and T. Chen-Khong, "Evolutionary Optimal Virtual Machine Placement and Demand Forecaster for Cloud Computing," in *Proc. IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 2011.

[13] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[14] N. T. Hieu, M. Di Francesco, and A. Y. Jääski, "A Virtual Machine Placement Algorithm for Balanced Resource Utilization in Cloud Data Centers," in *Proc. IEEE International Conference on Cloud Computing (CLOUD)*, 2014.

[15] X. Li, Z. Qian, S. Lu, and J. Wu, "Energy Efficient Virtual Machine Placement Algorithm with Balanced and Improved Resource Utilization in a Data Center," *Mathematical and Computer Modelling*, vol. 58, no. 5, pp. 1222–1235, 2013.

[16] X. Li, Z. Qian, R. Chi, B. Zhang, and S. Lu, "Balancing Resource Utilization for Continuous Virtual Machine Requests in Clouds," in *Proc. IEEE International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, 2012.

[17] J. Xu and J. A. Fortes, "Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments," in *Proc. IEEE/ACM International Conference on Green Computing and Communications & International Conference on Cyber, Physical and Social Computing*, 2010.

[18] A. C. Adamuthe, R. M. Pandharpatte, and G. T. Thampi, "Multiobjective Virtual Machine Placement in Cloud Environment," in *Proc. IEEE International Conference on Cloud & Ubiquitous Computing & Emerging Technologies (CUBE)*, 2013.

[19] F. L. Pires and B. Barán, "Multi-Objective Virtual Machine Placement with Service Level Agreement: A Memetic Algorithm Approach," in *Proc. IEEE/ACM International Conference on Utility and Cloud Computing*, 2013.

[20] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal Virtual Machine Placement Across Multiple Cloud Providers," in *Proc. IEEE Asia-Pacific Services Computing Conference*, 2009.

[21] M. Sun, W. Gu, X. Zhang, H. Shi, and W. Zhang, "A Matrix Transformation Algorithm for Virtual Machine Placement in Cloud," in *Proc. IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2013.

[22] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee, "Online Allocation of Virtual Machines in a Distributed Cloud," in *Proc. IEEE INFOCOM*, 2014.

[23] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, and S. Cheng, "Energy-Saving Virtual Machine Placement in Cloud Data Centers," in *Proc. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2013.

[24] M. Mishra and A. Sahoo, "On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach," in *Proc. IEEE International Conference on Cloud Computing (CLOUD)*, 2011.

[25] M. Alicherry and T. Lakshman, "Optimizing Data Access Latencies in Cloud Systems by Intelligent Virtual Machine Placement," in *Proc. IEEE INFOCOM*, 2013.

[26] Personal communication with HUAWEI company.

[27] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[28] F. Ma, F. Liu, and Z. Liu, "Multi-Objective Optimization for Initial Virtual Machine Placement in Cloud Data center," *Journal of Information &Computational Science*, vol. 9, no. 16, pp. 5029–5038, 2012.

[29] "Bitbrain Workload Traces," http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains.

[30] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis," in *Proc. ACM Symposium on Cloud Computing*, 2012.