SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

July 2020, Vol. 63 172101:1–172101:23
https://doi.org/10.1007/s11432-019-9948-6

Online traffic-aware linked VM placement in cloud data centers

Liwei LIN[1], David S. L. WEI[2], Ruhui MA[1], Jian LI[1*] & Haibing GUAN[1]

[1]*Shanghai Key Laboratory of Scalable Computing and Systems, Shanghai Jiao Tong University, Shanghai* 200240*, China;*
[2]*Department of Computer & Information Sciences, Fordham University, New York* NY 10458*, USA*

Received 27 February 2019/Revised 19 May 2019/Accepted 16 June 2019/Published online 18 May 2020

Abstract In cloud computing, virtual machine (VM) placement plays a crucial role in data center (DC) management, as different ways of VM placement may require different system resources. As Cisco research reveals that virtualization of DC increases traffic within the DC and causes network bandwidth to become scarce resource, recent researches have been focusing on traffic-aware VM placement. However, previous traffic-aware VM placement schemes treat the VM placement as a static process in that they do not take into account the impact of the current placement decision on the subsequent placement. In this paper, we thus propose a novel online traffic-aware VM placement scheme. Our scheme views VM placement as a context-sensitive dynamic process in that the decision of every step of the placement is made aiming at helping the subsequent steps of placement to reduce the required network bandwidth in the long run. In our scheme, we consider not only inter-VM traffic but also the bandwidth constraint of a physical machine (PM) when making a VM placement decision. To realize our objective, we put those VMs with close end time in the same or close proximity PMs so that when the VMs are terminated, one can make enough room for the future arrivals so as to not only minimize the number of active PMs but also reduce networking costs. We conduct extensive simulations to verify the superiority of our scheme in terms of networking costs and energy consumption. Simulation results show that our scheme outperforms improved-best-fit-decreasing (IBFD) scheme, a revised best-fit version that takes inter-VM traffic into account, by 30%–40% on network cost under various scenarios. Our scheme also promises 10%–25% power savings compared with IBFD.

Keywords linked VMs, traffic-aware, online VM placement, cloud data center, energy efficient

Citation Lin L W, Wei D S L, Ma R H, et al. Online traffic-aware linked VM placement in cloud data centers. Sci China Inf Sci, 2020, 63(7): 172101, https://doi.org/10.1007/s11432-019-9948-6

1 Introduction

Virtualization is the key technology that enables cloud users to share the physical resources in cloud data centers (DCs) in a cost-effective way and can provide high availability service [1]. In a virtualized DC, virtual machines (VMs) are created according to users' demands, and users run their applications on their VMs that are indeed running on physical machines (PMs). To optimize resource allocation, how those users' VMs are placed in those PMs plays crucial role, as different ways of VM consolidation and placement may require different computational resources (CPU time, memory space) and network resources (switches, network bandwidth). However, traditional VM consolidation schemes have focused on minizing the usage of CPU, memory [2–5], and the VM consolidation problem has been formulated as bin packing problem or its variant before it is solved [6].

Though virtualization provides a flexible way to support various large scale applications or systems, such as multi-tier web application [7], MapReduce [8, 9], and network function virtualization (NFV) [10,

* Corresponding author (email: li-jian@sjtu.edu.cn)

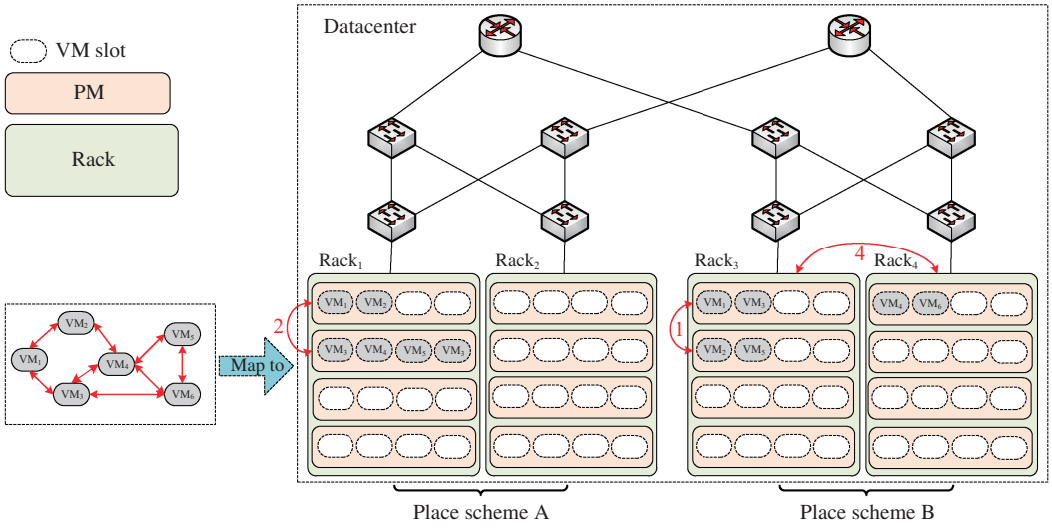

Figure 1 (Color online) The placement of linked VMs in data center.

11], such a large scale task may need to run on more than one VM. For example, a typical multi-tier web application consists of three tiers: presentation layer (web tier), business logic layer (app tier), and data access layer (DB tier) [7]. Different layers usually run on different VMs and have different memory access patterns. These VMs cooperate with each other to support the web application. Therefore, we consider that a user's request is a set of linked VMs in which some VMs cooperate with each other via message passing.

Considering a real situation in a cloud DC in which users' requests (represented by a set of linked VMs) arrive online, and the placement decisions need to be made in real-time without any prior knowledge of other users' future requests, and the fact that the off-line version of VM consolidation has been shown to be NP-hard [12, 13], the problem of online VM consolidation is extremely challenging in reality and may need to be solved in a heuristic way. Besides, researches in [14–16] show that a DC's internal traffic, including inter-VM traffic, contribute up to about 80% of the whole traffic of the DC. Thus, in a DC network, network bandwidth is quite limited at both the PM level and the rack level and thus need to be used frugally. Thence, traffic-aware VM placement becomes a critical issue in DC management. Consolidating VMs with the objective of minimizing network resource, e.g., network bandwidth, has been receiving attention recently [12, 17–19].

As shown in Figure 1, the VM placement problem that we will tackle is different from those solved by existing algorithms in that the VMs to be placed in our algorithm are linked VMs. There are several racks in the DC, and a rack consists of several PMs, and a PM can accommodate several VMs. The distance between two PMs is counted as the number of switches along their shortest path. The set of linked VMs of a user[1)] will be mapped to the PMs in the racks in a way of minimizing the network bandwidth required by the set of VMs for completing its application. In Figure 1, a user's request, which is a set of six linked VMs, is to be served. The red arrows represent the communication needs for the pair of VMs. For simplicity, we temporarily assume that the amount of traffic between VMs are equal[2)]. The figure shows two different placement schemes that consume different network costs and energy. In scheme A, 6 VMs are placed in Rack_1, and the needed network traffic is 2 which is the traffic between the two PMs that accommodate the VMs. In scheme B, 4 VMs are placed in Rack_3 and two VMs are placed in Rack_4. The needed network traffic between Rack_3 and Rack_4 is 4, and the needed network traffic between the two PMs in Rack_3 is 1. It is not hard to see that scheme A is with lower network cost and is more efficient than scheme B in terms of required bandwidth. Also, scheme A consumes less energy than scheme B, since scheme A uses only 2 active PMs, but scheme B uses 3 active PMs.

1) Only the VMs belonging to the same user may need the communications among VMs.
2) In our simulations, we indeed assume that the inter VM traffic follows normal distribution.

Also, in Section 5, we show that end time is another important factor that needs to be taken into account when devising an online VM placement scheme. In a virtualized DC, the creation and destruction of VMs is an important operation of VM management. A VM is created and destructed based on its start time and end time. Therefore, in VM management, information such as the end time of each VM is recorded such that the VM will be destructed at the right time. For example, in cloud systems, one needs to define the VM's end time [20][3)], and some other studies also state that it is required to preset VMs' end time [21–23]. By considering the end time, our VM consolidation scheme not only guarantees the minimized usage of the number of PMs, but also minimizes the required network bandwidth.

We summarize the contributions and novel ideas of our work as follows:

(1) We take the VMs' end time into consideration when making the placement decision for the arriving VMs. The VMs with the closer end time, if placed in the slots of close proximity, will free these slots together with higher probability and likely benefits future arriving VMs.

(2) Unlike previous schemes, our traffic-aware VM consolidation scheme considers the VMs to be placed as a set of linked VMs in which the weight associated with a link represents the required traffic between the two VMs. We also prove that this problem is NP-hard. Our greedy algorithm, with best efforts, places the pair of VMs with heavy communication needs in the same PM so that the network bandwidth is absorbed, thereby reducing the usage of network bandwidth.

(3) As link overflow has negative impact on the system's performance, our placement scheme also takes the risk of PM link overflow into consideration.

(4) Our scheme produces less free slot fragmentation in active PMs and thus activate as few as possible sleeping PMs, thereby reducing energy consumption.

2 Related work

Ref. [24] performs an overview of VM placement problems in DC as well as their current solutions. So far, the VM placement problem has been studied in different aspects.

In the previous researches on virtualized DC, the VMs to be placed are assumed to be independent from each other. Their resource requirements are mainly CPU and memory. So these studies mainly focused on the CPU and memory resource allocation [25], and the communication needs among VMs were not considered.

Due to multiple physical constraints and different optimization objectives in DC, researchers have focused on different targets in devising their VM placement strategies. They thus study this problem from various aspects, namely energy efficiency [26,27], scalability [12,28], availability [29,30], reliability [31,32], survivability [19], and network traffic [12, 17–19]. In general, VM placement can be classified into two categories, namely energy consumption concern strategy and QoS concern strategy [33]. Most of the placement strategies are either minimizing the utilities of physical resources or saving energy consumption by shutting down the unused physical devices [17]. Although VMs can be re-mapped to PMs via VM migration [34], migration can still increase the network burden.

Moreover, placing a set of linked VMs is more complex than placing a set of independent VMs. Placing such VMs to PMs has to take the communications between VMs into consideration, and such problems have been proven to be NP-hard [12, 13]. Therefore, inter-VM traffic is another important factor in the placement of linked VMs, although there have been a few studies on traffic-aware VM placement in DC [12, 17–19]. The VM placement scheme, called VMPlanner, in [17] optimizes both virtual machine placement and traffic flow routing. The main function of the study is to turn off as many unneeded network elements as possible for power saving. However, this study is not a pure online VM consolidation scheme. Authors [12] traced and analyzed the traffic patterns in DC to observe the potential network scalability benefit, and proposed an approximation algorithm, named TVMPP, to solve the VM placement for scalability benefit. However, the states in DC are dynamic in that old VMs will be destroyed and

3) Get the list of events generated on any VM. https://portal.nutanix.com/#/page/docs/details?targetId=API_Ref-Acr_v4_6:vms_api_getVMEvents_auto_r.html.

the new VMs are to be created. This indicates that the states are changing and TVMPP cannot fit such scenario well. Ref. [18] formulated the VM consolidation problem into a stochastic bin packing problem and proposed an approximation algorithm as a solution. The solution is aware of the bandwidth demand, but does not take the network topology into account. Indeed, in reality, the constraints are more complex. The solution model should be improved to suit a realistic environment. Ref. [35] processed the VM placement by taking the traffic between VMs into consideration to reduce the network cost in DC. But the two communication models in [35] are not general cases in practice. The solution proposed by [19] is mainly to improve the VMs' survivability in DC. However, the authors do not take the performance of the applications of the VMs' owner into consideration. In fact, in many scenarios, there is no need to have many backup VMs as what Ref. [19] proposed. In summary, the studies of VM placement mentioned above can be categorized as the off-line VM placement scheme in that the placement decision, whether it is traffic-aware or not, is made only for the good of the current arriving VMs without considering those future arrivals. Fortunately, the research of online VM placement has been conducted by some researchers [5]. However, this study focuses on minizing the usage of computing resources, CPU, memory, disk, and has ignored the network bandwidth as an important factor in performance measurement. As we will show in Section 5 that the end time of each VM may also affect the online placement decision and in turn affect the system and application performance, in this paper, we propose to study traffic-aware online linked VM placement by taking the end time of each VM into account.

3 Models and definitions

In this section, we introduce the notations and models that are needed in the description and discussion of our VM consolidation algorithms.

3.1 The environment of linked VM placement

3.1.1 *Physical DC environment*

We consider that a DC consists of n_r racks, and a rack can accommodate n_{rp} physical machines (PMs). Each PM has n_{ps} slots and each of which accommodates a VM. PMs are connected with each other through several switches or routers. The distance between two PMs is the number of links along their shortest path. We name the distance of two PMs as Hop, as shown in Table 1. To reduce the energy consumption, the empty PMs (the PMs that do not accommodate any VM) will be in sleeping state. When there are no sufficient active PMs for the VM placement, some sleeping PMs will be activated.

3.1.2 *VM and linked VMs*

In reality, a DC services many users, and each user leases several VMs for a period of time. From the view point of DC manager, the VMs' lifespan in DC is a dynamic process in that the old VMs are destroyed at their end time, and some new VMs are created when new users' requests arrive.

Every user requires $n_{u_i}(n_{u_i} > 0)$ VMs cooperating with each other to run his/her applications. In order to finish the tasks, these cooperative VMs need to communicate with each other, and thus induce network cost in data center. Thus, the input to the consolidation algorithm is a set of linked VMs for each user. An occupied slot is freed if the VM that it accommodates reaches its end time.

3.2 Linked VMs of a user and some equations

We assume that some VMs of a user may need to communicate with each other and the set of linked VMs of a user is denoted by a graph $G(\mathrm{VM}) = (V, E)$, where V represents the VM set, and E represents the set of communication links among the VMs. $|V|$ is the number of VMs in $G(\mathrm{VM})$, and $|E|$ is the number of links in $G(\mathrm{VM})$. The notations needed for the description of our algorithms are summarized in Table 1.

Table 1 Notations

v	A VM
$e(v_i, v_j)$	$e(v_i, v_j) = 1$ if there is a direct communication link between v_i and v_j, otherwise $e(v_i, v_j) = 0$
$\mathrm{tr}(v_i, v_j)$	Direct communication traffic between v_i and v_j, otherwise $\mathrm{tr}(v_i, v_j) = 0$
$e(v_i, *)$	All direct communication links of v_i
$\mathrm{tr}(v_i, *)$	All direct communication traffic of v_i
VS	A VS represents a set of VMs. e.g., VS_a represents the VMs to be placed in or residing in PM P_a
$v_i^{t_e}$	v_i's end time
$\mathrm{VS}_a^{\mathrm{out}}$	The whole out traffic of VS_a
B_c	The bandwidth constraint of a PM
NC	The network cost
$\mathrm{Hop}(v_i, v_j)$	Number of physical links along the shortest path between PM P_{v_i} and PM P_{v_j} after v_i and v_j are placed
$\lvert P(fs)\rvert$	Number of free slots in PM P
aPMs	The set of active PM in DC

We then define some important notations for the convenience of algorithm description. Let the arriving users follow Poisson distribution $P(x = k) = \frac{\mathrm{e}^{-\lambda}\lambda^k}{k!}$, then λ represents the average number of arriving users during a unit of period of time. We set $\epsilon = \frac{0.69U}{\lambda}$, where U is a unit of time. Also, let $T(v_i^{t_e}, v_j^{t_e})$ be defined as

$$T(v_i^{t_e}, v_j^{t_e}) = \begin{cases} 1, & 0 \leqslant v_i^{t_e} - v_j^{t_e} \leqslant \epsilon, \\ 0, & \text{else.} \end{cases} \tag{1}$$

Then, TClose can be calculated by

$$\mathrm{TClose}(v_{\mathrm{new\_user}}, \mathrm{VS}) = \max\left\{ \sum_{v_j \in \mathrm{VS}} T(v_{\mathrm{new\_user}}^{t_e}, v_j^{t_e}), \sum_{v_j \in \mathrm{VS}} T(v_j^{t_e}, v_{\mathrm{new\_user}}^{t_e}) \right\}. \tag{2}$$

$T(v_i^{t_e}, v_j^{t_e})$ states if the two VMs' end time is within the range of ϵ[4)], and in the equation, one of v_i and v_j is the VM waiting for being placed, and the other one is the VM which has been placed in a PM. So, TClose counts the number of VMs in VS whose end time is within the ϵ range of the end time of a user's VMs. This means that TClose can be used to estimate which PM has more VMs whose end time is close to the end time of new arriving VMs. If the VMs with high TClose are placed in the same PM, it will free more slots for later arriving VMs with higher probability.

3.3 Inter-VM traffic estimate model

Before run time, the traffic is hard to predict exactly. DC can estimate the traffic range of each link according to the functions of VM pairs. According to the history data in DC, it can estimate the traffic of different types of link [36].

For the convenience of performance analysis, we partition traffic flows into several ranges $T_R = \{t_{r_1}, t_{r_2}, \ldots, t_{r_n}\}$. As the link traffic can be estimated as a range which falls in a subrange in T_R, in order to reduce the risk of traffic overflow to guarantee the QoS, all VM traffic through the network interface in PM p should obey the inequation (3). $p(\mathrm{vm})$ denotes the VMs that are currently placed in p.

$$\sum_{v_i \in p(\mathrm{vm})} \mathrm{tr}(v_i, *) - \sum_{v_i, v_j \in p(\mathrm{vm})} \mathrm{tr}(v_i, v_j) \leqslant B_c. \tag{3}$$

When conducting the simulation, the traffic of a link can only be estimated in a range of T_R. We denote link $e(v_i, v_j)$'s estimated traffic range by $e^{\mathrm{tr}}(v_i, v_j)$. So $\lceil e^{\mathrm{tr}}(v_i, v_j) \rceil$ and $\lfloor e^{\mathrm{tr}}(v_i, v_j) \rfloor$ are denoted as the upper and lower bound of this range respectively. Intuitively, $\lfloor e^{\mathrm{tr}}(v_i, v_j) \rfloor \leqslant \mathrm{tr}(v_i, v_j) \leqslant \lceil e^{\mathrm{tr}}(v_i, v_j) \rceil$. So if the condition satisfies inequation (4), it should also satisfy inequation (3).

4) We will show how we choose this ϵ value in the section of TClose analysis.

$$\sum_{v_i \in p(\mathrm{vm})} \lceil e^{\mathrm{tr}}(v_i, *) \rceil - \sum_{v_i, v_j \in p(\mathrm{vm})} \lfloor e^{\mathrm{tr}}(v_i, v_j) \rfloor \leqslant B_c. \tag{4}$$

In this paper, to verify the superiority of our algorithm, in the evaluation, we assign the link weight by estimating its traffic range so that the decision made can be more accurate.

3.4 System and runtime traffic model

In reality, with an explosive growth of DC traffic, network bandwidth constraint becomes a more and more critical issue. Recent studies [12, 37, 38] show the bursty nature of DC traffic. Most of the previous studies, except [18], assume that the inter-VM traffic is fixed, which is not the case in reality in a DC environment. In this study, we use random variables to describe the traffic characteristic between VMs. This can better represent such uncertainty of traffic features. So, the traffic of a VM pair is not a constant, and we use a random variable, denoted by X, to estimate inter-VM traffic. As the study in [18], we assume that the link traffic between two directly linked VMs follows normal distribution $N(\mu, \sigma^2)$ approximately, where μ represents the expect value of X, and σ represents the variance of X. As the traffic cannot be a negative value, we set the lower bound of traffic value to be 0. Let $\mathrm{tr}(v_i, v_j)$ stand for the traffic between VMs v_i and v_j. Then, $\mathrm{tr}(v_i, v_j)$ follows normal distribution with parameters μ and σ. There are actually heavy traffic and light traffic coexisting in a DC, and traffic flows of different VM pairs are unequal. Setting different values of μ and σ enables us to more accurately model the complex traffic demands of various applications in real DC.

Also, the communicating VMs belonging to the same user may be placed in different PMs. This will cause different traffic to use a physical link. We assume that traffic flows of VM pairs are independent from each other. There might be several traffic flows using the same physical link at the same time. Also, traffic flows produced from a set of linked VMs is the sum of the flows of VM pairs, whose one VM, namely v_i, and the other VM, namely v_j, are placed in different PMs. Let e_N be the set of links between VS_a and VS_b, we have

$$\mathrm{tr}_{e_N}(\mathrm{VS}_a, \mathrm{VS}_b) = \sum_{\substack{v_i \in \mathrm{VS}_a \\ v_j \in \mathrm{VS}_b}} \mathrm{tr}(v_i, v_j). \tag{5}$$

Also, let f_i be the traffic flow of the ith link in e_N, we have $F = \{f_1, f_2, \ldots, f_{|e_N|}\}$, $f_i \sim N(\mu_i, {\sigma_i}^2), i \in 1, 2, \ldots, |e_N|$. F is the combination of all VM pair flows, and it is the joint distribution of $f_1, f_2, \ldots, f_{|e_N|}$. Based on the assumption that each VM pair flow follows the normal distribution and they are independent of each other, F is a joint normal distribution, and follows the $N(\mu_F, \sigma_F^2)$.

$$\mu_F = \sum_{i=1}^{|e_N|} \mu_i, \tag{6}$$

$$\sigma_F^2 = \sum_{i=1}^{|e_N|} \sigma_i^2. \tag{7}$$

So the combined traffic $\mathrm{tr}_{e_N}(\mathrm{VS}_a, \mathrm{VS}_b)$ between VS_a and VS_b follows normal distribution $N(\mu_{e_N}, \sigma_{e_N}^2)$, where

$$\mu_{e_N} = \sum_{\substack{v_i \in \mathrm{VS}_a \\ v_j \in \mathrm{VS}_b}} e(v_i, v_j) \cdot \mu \text{ [5)]}, \quad \text{and} \quad \sigma_{e_N} = \sqrt{\sum_{\substack{v_i \in \mathrm{VS}_a \\ v_j \in \mathrm{VS}_b}} e(v_i, v_j) \cdot \sigma^2}.$$

Then, VS_a's out traffic is defined as

$$\mathrm{VS}_a^{\mathrm{out}} = \sum_{v_i \in \mathrm{VS}_a} \mathrm{tr}(v_i, *) - \sum_{v_i, v_j \in \mathrm{VS}_a} \mathrm{tr}(v_i, v_j). \tag{8}$$

5) $e(v_i, v_j) = 1$ if $e(v_i, v_j) \in E$ of $G(\mathrm{VM})$. Otherwise, $e(v_i, v_j) = 0$.

Follows the normal distribution $N(\mu_{e_N}^{\text{out}}, \sigma_{e_N}^{\text{out}^2})$, $\mu_{e_N}^{\text{out}} = \sum_{v_i \in \text{VS}_a} e(v_i, *) \cdot \mu - \sum_{v_i, v_j \in \text{VS}_a} e(v_i, v_j) \cdot \mu$, and

$$\sigma_{e_N}^{\text{out}} = \sqrt{\sum_{v_i \in \text{VS}_a} e(v_i, *) \cdot \sigma^2 - \sum_{v_i, v_j \in \text{VS}_a} e(v_i, v_j) \cdot \sigma^2}.$$

As the purpose of our scheme is traffic aware VM placement, we should reduce the out traffic of each grouped VMs, i.e., VS_a, by some means. In other words, we should increase $\sum_{v_i, v_j \in \text{VS}_a} e(v_i, v_j)$. The more inter-VM traffic absorbed by a PM, the less out traffic will be for the PM and thus will consume less physical network resource.

In normal distribution $X \sim N(\mu, \sigma)$, $P\{X < (\mu + 3 \times \sigma)\} \approx 99.87\%$. Since the probability of 99.87% satisfies high QoS requirement, such as service-level agreement (SLA), in order to reduce the risk of physical link overflow to guarantee required QoS, we choose $\mu + 3 \times \sigma$ as a VM pair traffic weight after the VM pair are placed in a PM. This way, the remaining physical link bandwidth can be more accurately estimated. So the traffic of VM pairs (v_i, v_j) which have been placed in PM p can be calculated as follows:

$$\text{tr}(v_i, v_j) = e(v_i, v_j) \cdot \mu + 3 \times e(v_i, v_j) \cdot \sigma. \tag{9}$$

Then the remaining bandwidth B_r of p is

$$\begin{aligned} B_r(p) &= B_c - \left(\sum_{v_i \in p} \text{tr}(v_i, *) - \sum_{v_i, v_j \in p} \text{tr}(v_i, v_j) \right), \\ &= B_c - (\mu_{e_N}^{\text{out}} + 3 \times \sigma_{e_N}^{\text{out}}). \end{aligned} \tag{10}$$

Let VS_w be the set of VMs waiting to be placed in PM p, and VS_p be the VMs that have been placed in p. In order to reduce the risk of overflow to guarantee QoS, the out traffic of PM p should obey the inequation (11). Let $B_r(p)$ be the remaining bandwidth of PM p, then we have

$$\begin{aligned} &\text{tr}^{\text{out}}(\text{VS}_w) + \text{tr}^{\text{out}}(\text{VS}_p) \leqslant B_c \\ &\Rightarrow \text{tr}^{\text{out}}(\text{VS}_w) \leqslant B_c - \text{tr}^{\text{out}}(\text{VS}_p) \\ &\Rightarrow \sum_{v_i \in \text{VS}_w} \lceil e^{\text{tr}}(v_i, *) \rceil - \sum_{v_i, v_j \in \text{VS}_w} \lfloor e^{\text{tr}}(v_i, v_j) \rfloor \leqslant B_r(p). \end{aligned} \tag{11}$$

Also, the number of VMs to be placed in a PM should not exceed the PM's capacity. That is

$$|p_{vm}| + |vs| \leqslant n_{ps}, \tag{12}$$

where $|p_{vm}|$ represents the number of VMs that have been placed in PM p, and $|vs|$ represents the number of VMs to be placed in PM p.

In summary, intuitively, the objective of our online VM placement algorithm is to find a PM (or some PMs in a rack), denoted by VS_x, for those new arrivals in VS_a, such that $\text{TClose}(\text{VS}_a, \text{VS}_x)$ is the maximum among those TClose values while follows the conditions of inequations (11) and (12), and to minimize the network cost (NC) defined by (13).

3.5 Problem statement

We denote the PM set in DC as $P = \{p_1, p_2, \ldots, p_m\}$. Each PM has several free slots, whose number range is from 0 to n_{ps}. The problem is to partition the set of linked VMs of a user into several subsets $\{G_{s_1}, G_{s_2}, \ldots, G_{s_k}\}$ such that the size of each subset is constrained by the number of free slots in each PM. We then map these subsets into PMs, as shown in Figure 2.

In order to measure the network resource consumption, we define the network cost (NC) as the performance metric of network bandwidth consumption.

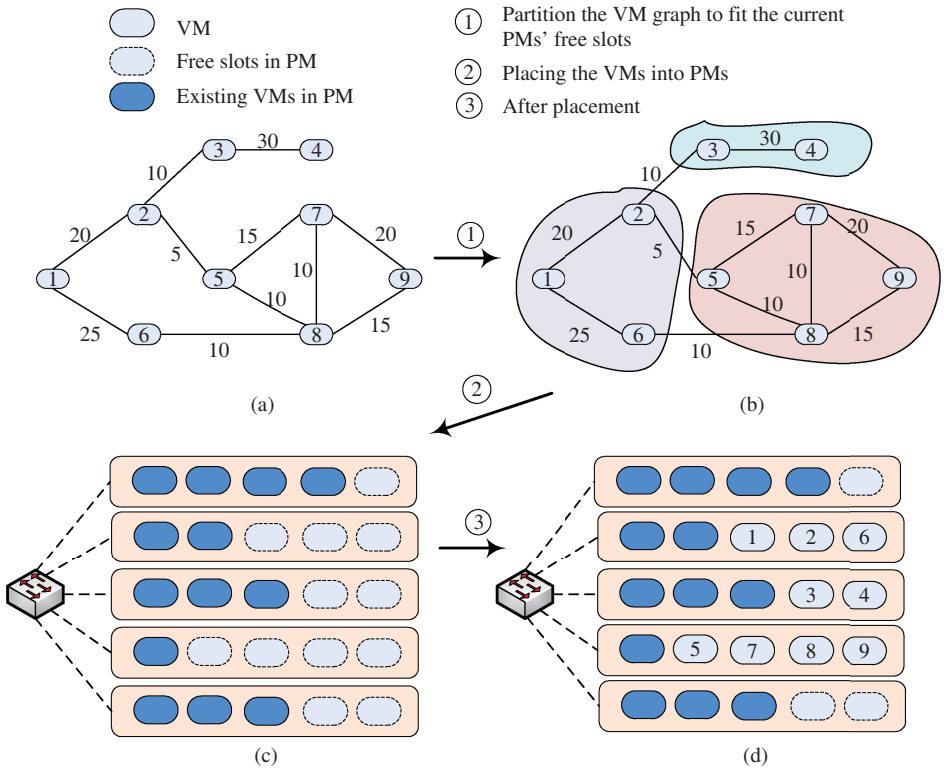

Figure 2 (Color online) Example of linked VM partition and placement. (a) Arriving VM graph of a user; (b) VM sub-graphs after partition; (c) PM states before VM placement; (d) PM states after VM placement.

Definition 1 (Network cost (NC))**.** NC measures the bandwidth consumed by the traffic of all VM pairs. The bandwidth consumed by VM pair (v_i, v_j) is determined by the number of hops along the shortest path of the two PMs P_{v_i} and P_{v_j} that accommodate v_i and v_j, respectively. NC can be calculated by

$$\mathrm{NC} = \sum_{p_{v_i} \neq p_{v_j}} [\mathrm{tr}(p_{v_i}, p_{v_j}) \times \mathrm{Hop}(p_{v_i}, p_{v_j})]. \tag{13}$$

The objective of VM placement algorithm is to minimize the network cost (NC). As shown in Table 1, $\mathrm{Hop}(v_i, v_j)$ represents the distance (the number of hops) between the residing location of VM v_i and the residing location of VM v_j.

When measuring NC, the real DC network topology should also be taken into account. Our objective is to develop an ideal VM consolidation scheme such that for each placement of a set of linked VMs, NC is minimized.

In our design, the placement problem is divided into two phases: (1) partition the set of linked VMs according to the number of free slots in PMs; (2) map the partitions to the right PMs, as shown in Figure 2. There are two major concerns that need to be addressed in the two-phase placement scheme: (1) The partition scheme can directly influence NC. The purpose of partition is to minimize the total traffic of inter-partitions. (2) The other concern is the free slot fragmentation/concentration. The less free slot fragmentation in a PM is, the more traffic will be absorbed by the PM, and thus the NC will be reduced.

In Sections 4 and 5, we will address these two aspects, and propose our solution for the linked VM placement problem.

4 Partition of linked VMs

Due to the constraint of free slots in PMs, the set of linked VMs needs to be divided into several subsets before the VMs are placed in PMs. As introduced before, the traffic flows of different VM pairs are not equal, inappropriate partition schemes will induce higher inter-partition traffic and will in turn result in heavier physical link load and thus higher NC.

As shown in Figure 2(a) and (b), each partition will be placed in a PM. The inter-partition traffic will bring traffic to inter PMs and consume DC network resource. So min-cut between partitions is essential for bandwidth saving. Designing an optimal (or near optimal) partition is critical for the DC manager to reduce the network cost.

As mentioned before, in this phase, a set of linked VMs is partitioned into several subsets of different sizes according to the available free slots in PMs. Also, the number of partions is determined by the number of available PMs. In order to save the energy consumption, we select minimum number of PMs to accommodate these partitions. The selection of PMs should subject to the following objectives:

(1) Satisfy with the bandwidth constraint of (11) to reduce the risk of traffic overflow;

(2) Minimize the number of active PMs;

(3) Reduce the physical network bandwidth usage.

The problem is described as follows.

There are n linked VMs that form a graph $G = (V, E)$. $V = \{v_1, v_2, \ldots, v_n\}$ denotes the VM set, and E denotes the link set. There are m available PMs $P = \{p_1, p_2, \ldots, p_m\}$ that have enough free slots for VM placement. We are to partition graph G into n_s sub graphs $G_s = \{G_1, G_2, \ldots, G_{n_s}\}$ according to the available free slots in P.

Objective: to minimize

(1)

$$\sum_{\substack{G_i \in Gs \\ G_j \in Gs \\ i \neq j}} \mathrm{tr}(G_i, G_j);$$

(2) $\|\mathrm{aPMs}\|$, the number of active PMs, while subject to (11) and (12).

Our VM placement is affected by the set of PMs who have enough free slots. There are two factors that affect the network cost.

(1) The distance (number of hops) between the selected PMs.

(2) The free slot concentration of PMs.

Basically, the more concentrated the free slots is, the better for bandwidth saving would be. Let PMset be the active PMs with free slots, the degree of free slot concentration δ can be measured by

$$\delta = \frac{\sum_{p_i \in \mathrm{PMset}} |p_i(fs)|}{|\mathrm{PMset}|}, \tag{14}$$

where $|p_i(fs)|$ represents the number of free slots in p_i. δ indicates the degree of free slots concentration. The higher the δ is, the more concentrated free slots would be.

Theorem 1. The optimal partition of G, the set of linked VMs, is an NP-hard problem.

Proof. As discussed previously, the problem of VM set partition is to partition the graph $G(V, E)$ into k sub-graphs of specific size with minimum cuts. If we set $k = 2$, the problem is reduced to partition G into 2 sub-graphs, $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$. By setting $|V_1| = |V_2|$, this problem becomes the minimum bisection problem (MBP), which is the special case of our problem. MBP has been proven to be NP-hard in [39, 40]. Thus, the problem that we are tackling is also an NP-hard problem.

5 Practical situation consideration

In this section, we will illustrate how VM's end time affect the system's performance.

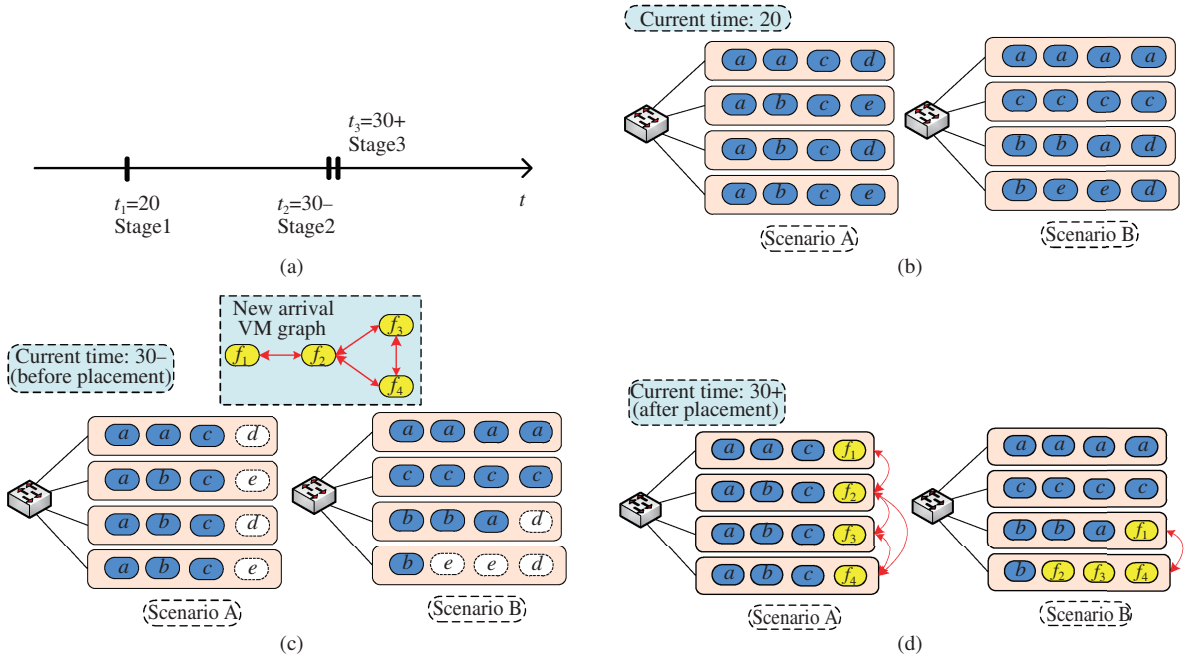

Figure 3 (Color online) An example of VM placement by taking end time into account (the VMs with the same label belong to the same user). (a) The time axis of each VM placement stage. (b) t_1: current time is 20. VMs' end time are (a: 56; b: 69; c: 47; d: 25; e: 24). (c) t_2: current time is 30. 4 new VMs (f_1, f_2, f_3, f_4) arrive. 4 VMs labeled by d or e are freed. (d) t_3: current time is 30. After the 4 new VMs are placed.

We observe that in reality, most of the times, some slots of some PMs are occupied by the VMs. Unfortunately, previous researches of VM consolidation have actually considered only the static status in that their algorithms are designed only for dealing with a single wave of arrivals without considering the future arrivals. Since each PM has n_{ps} slots and because the VMs' end time may be different, once the old VMs terminated, their occupied slots will be freed for the newly arriving VMs. This could somehow produce low free slot concentration if VMs' end time is not considered when making a placement decision. Our research aims at online VM placement. By considering the VMs' end time in the placement decision, since there will be less slot fragmentation, new arrivals get better chance to be put in the close proximity, thereby minimizing the bandwidth demand.

Since energy consumption is also an important issue in DC, and in order to save the energy and improve resource utilization, when devising our placement algorithm, we also aim at using as few number of PMs as possible.

Naturally, the placement algorithm has to locate the free slots for the newly arriving VMs. The ideal case is that when the new user's VMs are arriving, they were placed to be as close to each other as possible. Intutively, the free slots in close proximity could benefit those future arrivals. We then use an example, as the two scenarios shown in Figure 3, to show how the end time affect network cost in VM consolidation.

In Figure 3, the VMs with the same label belong to the same user. So they have the same end time. The VMs belonging to the same user will be freed at the same time. To simplify the discussion, we temporarily set the weight of each link in the graph to be 1. We introduced the graph model in Subsection 3.2. As introduced in Section 3, the Hop (the number of hops) between any two PMs using the same router in Figure 3 is 2 because the communications between the two PMs need to cross 2 physical links. The Hop of VMs in the same PM is 0 because they do not need to cross any physical link.

The placement scheme maps the graph to the PMs in a way that the physical network cost is minimized. Scenario A is the scheme that ignores the VMs' end time, where Scenario B is the scheme that takes the VMs' end time into consideration. Figure 3(a) shows each stage time of the three VM placement stages. Figure 3(b) shows that 16 VMs are placed into 4 PMs. Current time is 20, and all the 4 PMs are full now.

The VMs with the same label are the ones with the same end time. Then, in Figure 3(c), the current time is 30, and the VMs labeled by d and e have been freed. The 4 new VMs, f_1, f_2, f_3, and f_4, arrive and can be placed in the 4 free slots. Figure 3(d) shows the scenarios of two possible placements. In the figure, the arrowed arcs represent needed communications between the two PMs. From Figure 3(d), one can see that the NC in Scenario A is $(1+1+1+1)\times 2 = 8$, and the NC in Scenario B is $(1+0+0+0)\times 2 = 2$. Thus, we can see that Scenario A requires much higher network cost than Scenario B.

This example also tells us that the placement scheme to be devised should be a traffic-aware one. Based on the location of those free slots, intuitively, if the VM pairs with high inter-VM traffic are placed to be close to each other, it will not only reduce the network cost, but also leave more network resources for the VMs that will arrive later. In addition, the link overflow can also degrade the system performance. So, for each placement decision, we also need to control the total out-traffic of each PM and each rack to avoid the out-link overflow.

6 Algorithm

Based on the above discussions, in this section, we describe our traffic-aware algorithm that takes both graph[6)] partition and VM's end time[7)] into consideration. Our VM placement scheme can be divided into two important phases:

(1) Select the appropriate set of PMs to accommodate the VMs. Meanwhile, VMs' end time is taken into consideration.

(2) Partition the graph and map the partitioned subgraphs to the selected PMs. When partitioning the graph, reducing inter-subgraphs traffic is the main concern.

Definition 2. Let PMset be the set of PMs selected to accommodate the VMs of the new arriving user. We define $V_{\text{set}}(t_1, t_2, \text{PMset})$ as the set of VMs currently residing in PMset and will be freed during time range $[t_1, t_2]$.

We then use $\delta_F(t_1, t_2)$ to define the degree of slot concentration in PMset during the time range $[t_1, t_2]$. Let $G(V, E)$ be the set of linked VMs of the new arriving user. Then, we have

$$\delta_F(t_1, t_2) = \frac{|V_{\text{set}}(t_1, t_2, \text{PMset})| + |V|}{|\text{PMset}|}, \tag{15}$$

$$\text{Bef}[e(v_a, v_b), \text{VS}_S] = \begin{cases} \lfloor e^{\text{tr}}(v_a, v_b)\rfloor + \sum\limits_{v_i \in \text{VS}_S} \lfloor e^{\text{tr}}(v_a, v_i)\rfloor + \sum\limits_{v_i \in \text{VS}_S} \lfloor e^{\text{tr}}(v_b, v_i)\rfloor, & v_a \notin \text{VS}_S, v_b \notin \text{VS}_S, \\ \lfloor e^{\text{tr}}(v_a, v_b)\rfloor + \sum\limits_{v_i \in \text{VS}_S} \lfloor e^{\text{tr}}(v_b, v_i)\rfloor, & v_a \in \text{VS}_S, v_b \notin \text{VS}_S, \\ \lfloor e^{\text{tr}}(v_a, v_b)\rfloor + \sum\limits_{v_i \in \text{VS}_S} \lfloor e^{\text{tr}}(v_a, v_i)\rfloor, & v_a \notin \text{VS}_S, v_b \in \text{VS}_S. \end{cases} \tag{16}$$

Assume that t_e is the end time of $G(V, E)$. To measure the future benefit of new arriving user's VM placement, we assume that $G(V, E)$ is to be placed in the selected PMset. We set $t_1 = t_e - \epsilon$ or $t_2 = t_e + \epsilon$, which means that the VMs in PMset are satisfied with the TClose with $G(V, E)$. So $\delta_F(t_e - \epsilon, t_e)$ or $\delta_F(t_e, t_e + \epsilon)$ can measure the concentration degree of the free slots freed by $G(V, E)$ and the TClose VMs in the PMset. Higher δ_F indicates that there would be better concentrated free slots for the future user's VMs.

Let n be the number of VMs to be placed. Algorithm 1 is developed for selecting the best PM set for accommodating the VMs of new user. Lines 1–3 process the case with insufficient free slots for the current VM placement. Lines 4–6 describe the scenario that active PMs are insufficient to accommodate

6) For the convenience of algorithm description and discussion, from now on, the set of linked VMs to be placed is viewed and treated as a graph.

7) Note that VMs in the same graph, i.e., the set of VMs belonging to the same user, always have the same end time.

Algorithm 1 PMSelection & VMMapping(G) //Select suitable set of PMs to accommodate the partitions of G

Input: The VM group graph $G(V, E)$;
Output: The placement result of G;
1: **if** $\sum_{P_i \in \mathrm{DC}} |P_i(fs)| < n$ **then**
2: return false;
3: **end if**
4: **if** $\sum_{P_i \in \mathrm{aPMs}} |P_i(fs)| < n$ **then**
5: Activate minimum number of sleeping PMs s.t. $\sum_{P_i \in \mathrm{aPMs}} |P_i(fs)| \geqslant n$;
6: **end if**
7: **if** $\exists$ rack, such that

$$\sum_{\substack{P_i \in \text{ rack} \\ P_i \in \text{ aPMs}}} |P_i(fs)| \geqslant n$$

then
8: **for** each such rack **do**
9: Order the aPMs in the rack in descend order according to PMs' free slot numbers;
10: Select PM set PM_a s.t. $\sum_{P_i \in \mathrm{PM}_a} |P_i(fs)| \geqslant n$;
11: Calculate δ using (14);
12: **end for**
13: Set $\delta_1 = \max\{\delta\}$, and let the corresponding set of PMs be PM_1;
14: **for** each such rack **do**
15: Order the aPMs in the rack in descend order according to PMs' TClose calculated by (2) with G;
16: Select PM set PM_b s.t. $\sum_{P_i \in \mathrm{PM}_b} |P_i(fs)| \geqslant n$;
17: Calculate δ_F using (15);
18: **end for**
19: Set $\delta_2 = \max\{\delta_F\}$, and let the corresponding set of PMs be PM_2;
20: **if** $\delta_1 \geqslant \delta_2$ **then**
21: PMset = PM_1;
22: **else**
23: PMset = PM_2;
24: **end if**
25: Partitions = Partition(G, PMset);
26: Map Partitions to the selected PMset;
27: **else**
28: Cut G into two sub-graphs, sG_1 and sG_2, with mini-cut;
29: PMSelection & VMMapping(sG_1);
30: PMSelection & VMMapping(sG_2);
31: **end if**

the VMs, and the DC manager will activate right number of sleeping PMs for the placement. This way, it can keep the number of active PMs to be as small as possible, thereby saving energy for DC.

Since end time affects free slot concentration degree, and in turns determines NC, during the PM set selection, we take both VM's end time and inter-VM traffic into consideration. Lines 8–24 consider two cases: (1) Calculate δ_1 of current free slots; (2) If the VMs are placed in the rack, we calculate δ_2 after the VMs are freed in the rack. If $\delta_1 \geqslant \delta_2$, it means that the current free slots are better for VM placement. Otherwise, i.e., $\delta_1 < \delta_2$, which means that the placement scheme will produce more concentrated free slots in the future.

For Algorithm 2, we define $\mathrm{Bef}[e(v_a, v_b), \mathrm{VS}_S]$ for the measurement in graph partition. $\mathrm{Bef}[e(v_a, v_b), \mathrm{VS}_S]$ of (16) is used to measure the traffic absorbed when $e(v_a, v_b)$ is included in VS_S. The high $\mathrm{Bef}[e(v_a, v_b), \mathrm{VS}_S]$ indicates that link $e(v_a, v_b)$ is a good choice to be included in VS_S. Lines 11–30 perform the main task of partition. When selecting the best VM from G to be included in S, the decision is made based on the calculated Bef. Basically, the VM with maximum Bef will be selected. Lines 12–18 are to process the case when S can include only one more VM. In this case, Lines 13–15 are to find a pair of VMs such that one of them has been in S, and the other is not, and select the one whose Bef is the maximum. Note that Line 16 actually includes only one more VM, i.e., either v_a or v_b, as either of them has been in S. Lines 19–28 are to select link (v_a, v_b) with maximum Bef and include both VMs, v_a and v_b, in S. Once lines 11–30 have been executed, the number of VMs included in S is the same as the number of free slots in PM $P_s[i]$.

Algorithm 2 Partition(G, P_s) //Graph partition algorithm in which the graph is linked VMs.

Input: Graph G(VM) and PMset P_s;
Output: A set of $|P_s|$ partitioned subgraphs;

Order the PMs in P_s in descend order according to PMs' free slot number;
Let the sorted order be $P_s[i], i = 1, \ldots, p$;
Partitions $= \emptyset$;
for each $P_s[i]$ in P_s in descend order **do**
  let $n = |P_s[i].fs|$;
  if G(VM) $\leqslant n$ and obeys inequation (11) **then**
    Return(Partitions = Partitions $\cup\, G$);
  else
    Order the links of G in descend order by weight, denoted by Link[];
    $S = \emptyset$;
    while $|S| < n$ **do**
      if $(n - |S| = 1)$ **then**
        for (Each $e(v_a, v_b)$ in Link[] in which $v_a \in S$ or $v_b \in S$) **do**
          Get $e_i(v_a, v_b)$ s.t. Bef$[e_i(v_a, v_b), S]$ is maxmized;
        end for
        $S = S \cup \{v_a, v_b\}$;
        Delete $\{v_a, v_b\}$ and $e_i(v_a, v_b)$ from G(VM);
        Delete all links in S from Link[];
      else
        Get the first link $e(v_a, v_b)$ from Link[];
        $S = S \cup \{v_a, v_b\}$;
        Link[] = Link[] $- e(v_a, v_b)$;
        for (Each $e(v_a, v_b)$ in Link[]) **do**
          Get $e_i(v_a, v_b)$ s.t. Bef$[e_i(v_a, v_b), S]$ is maxmized;
        end for
        $S = S \cup \{v_a, v_b\}$;
        Delete $\{v_a, v_b\}$ and $e_i(v_a, v_b)$ from G(VM);
        Delete all links in S from Link[];
      end if
    end while
    if S obeys inequation (11) for the PM of $P_s[i]$ **then**
      Partitions = Partitions $\cup S$;
      Delete $P_s[i]$ from P_s;
    else
      Backtracking the partition until S obeys inequation (11) for a PM P ($P \in P_s$) and min($|P.fs - |S||$);
      Partitions = Partitions $\cup S$;
      Delete P from P_s;
      Reorder P_s;
    end if
  end if
end for

7 TClose analysis

How to measure the TClose is an important factor that will affect the network cost in VM placement. The more accurate TClose measurement is, the higher benefit the online placement algorithm will gain. In reality, there is usually a time interval between the two arriving users. So we use ϵ in inequation (17) to measure the closeness between the end time of two VM sets.

$$|v_i^{t_e} - v_j^{t_e}| \leqslant \epsilon. \tag{17}$$

If VMs v_i and v_j satisfy inequation (17), then v_i and v_j can be put together so that their occupied slots could be freed together during a time interval defined by ϵ. As mentioned previously, users' arriving follows Poisson distribution. In a long enough period of time, the average number of arriving users during a unit of time approaches λ. To simplify the analysis, we assume that these λ users' arriving times are evenly distributed over this unit of time U.

As the users' arriving follows Poisson distribution parameter λ, as shown in (18), we could estimate

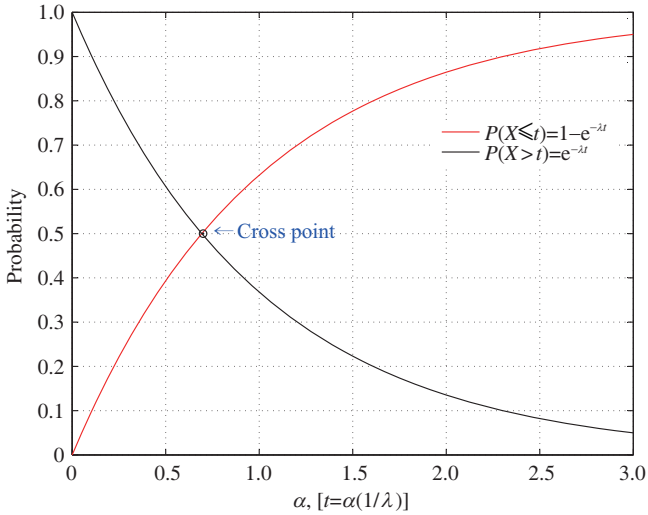

Figure 4 (Color online) Poisson distribution.

next user's arriving time.

$$P(N(t)=n)=\frac{(\lambda t)^n \mathrm{e}^{-\lambda t}}{n!}. \tag{18}$$

Let $P(X>t)$ be the probability of no arriving user during next time interval t. Then, we have

$$P(X>t)=P(N(t)=0)=\frac{(\lambda t)^0 \mathrm{e}^{-\lambda t}}{0!}=\mathrm{e}^{-\lambda t}. \tag{19}$$

Moreover, we denote $P(X\leqslant t)$ as the probability that some user(s) arrive during next time interval t. Then, we have

$$P(X\leqslant t)=1-P(X>t)=1-\mathrm{e}^{-\lambda t}. \tag{20}$$

The ideal case is that if different VM sets with close enough end time, they likely will finish at around the same time, and their occupied slots would be freed together for the new arrivals. If the time interval is too short, the ϵ is less meaningful, because the probability that the VM sets can be put together is too low. If the time interval is too long, the ϵ is also less meaningful, because the probability that the slots occupied by the bundled VM sets may be realeased at very different times and thus would not benefit the new arrivals.

To determine the right time interval, we set

$$\mathrm{e}^{-\lambda t}=1-\mathrm{e}^{-\lambda t}\Rightarrow \lambda t\approx 0.693 \tag{21}$$

to obtain 0.69, which is the cross point shown in Figure 4, to help calculate the right time interval. Then, we have $\epsilon=\alpha\times\frac{U}{\lambda}=\frac{0.69U}{\lambda}$. This means that about a user arrives per $\frac{0.69U}{\lambda}$ unit time on average. We thus set $\epsilon=\frac{0.69U}{\lambda}$. If v_i and v_j satisfy inequation (17) and are placed in the same PM, v_i and v_j will be freed together during ϵ time interval.

8 Evaluation

8.1 Simulation settings

We assume that the topology of the DC is FatTree. We set the parameter of FatTree K to be 8, i.e., there are 8 pods in DC, and each pod has 8 racks. Each rack can accommodate 16 PMs, and each PM has n_{ps} slots. The users' arriving rate follows Poisson distribution. The average rate of arrivals is λ. Each user requests several VMs to finish his/her job(s). Users' VMs may need to cooperate and communicate with each other to finish the job. We use the same assumption as the study and experiment settings in [41]

that the number of VMs requested by each user is exponentially distributed around a mean of 49. Also, a user may demand many more VMs in order to be able to finish larger scale applications. For example, in NFV, it may need hundred instances of virtual network functions (vnf) to cooperatively complete large scale services. So the range of the number of VMs required by a user may be from several to hundreds. In our simulation, we set the mean number of VMs that each user needs as a parameter. We set different values of VM numbers to observe its impact on system performance.

In reality, the number of free slots (n_{ps}) in each PM is determined by the amount of resources with which a PM is furnished and the amount of resources required by a VM. Per the study of [13,35,41] and compared with the real Amazon EC2 VM instance, the number of free slots may range from several to dozens. We thus also set different n_{ps} values in our simulation. In the simulation of users' arrivals, in Poisson distribution, we evenly divide a time unit to be 24 time intervals, which means that, on average, there are λ arriving users during 24 hours, for example. The lifespan of each user, i.e., the lifespan of the user's VM, is randomly selected from the range of [0, 150]. The number of VMs per user obeys the exponentially distribution with mean value numVM.

The bandwidth constraint B_{out} is set as 1 Gbps. In our simulation, since inter-VM traffic cannot be predicted accurately before placement, we adopt a method to estimate the inter-VM traffic. In reality, different inter-VM traffic flows are not equal, and we set the range of traffic to be in the range of [0, 300] Mbps. Then, using the analysis discussed in Subsections 3.3 and 3.4, an inter-VM traffic falls in a subrange of $[0, 30], [30, 60], \ldots, [270, 300]$. In the simulation, when generating a set of linked VMs for a user, we randomly select a subrange as the inter-VM traffic for randomly selected pairs of VMs. Then, the traffic of the VM pairs of the generated graph, the linked VMs, is dependent on μ and σ. Thus, using (9), we randomly generate the μ and σ for each VM pair's traffic such that $\mu + 3 \times \sigma$ falls in a subrange of the 10 subdivided ranges.

In our simulation, performance comparisons of the compared algorithms are carried out under the same input, and each test result is the average of 30 runs with different random inputs. Also, for a PM with no residing VM is put in sleeping mode so that the energy consumption can be correctly measured. Furthermore, in order to be more convenient in showing the results in a figure, the experimental results are min-max normalized using (22) such that the experimental results are confined in range [0, 1].

$$\text{NC}_i = \frac{\text{NC}_i^{\text{ori}} - \min\{\text{NC}\}}{\max\{\text{NC}\} - \min\{\text{NC}\}}, \tag{22}$$

where NC represents the set of simulation results, NC_i is the result after normalized, NC_i^{ori} is the origina simulation result, min{NC} represents the minimum one among the simulation results, and max{NC} represents the maximum one among the simulation results.

8.2 Validation of using TClose to achieve a better performance

The analysis presented in Section 7 shows how the right value of ϵ is determined in order to achieve better performance of our on-line VM placement. In this subsection, we verify that choosing the right value of ϵ does help on-line VM placement to offset the disadvantages of not knowing the future arrivals. As shown in Section 4 that on-line linked-VM placement is NP-hard, it is hard to obtain its optimal solution. We thus compare our algorithm with an off-line algorithm, an off-line version of BFD, to show that our on-line VM placement scheme is indeed a smart design. When an off-line algorithm is making a placement decision, the start time, the end time, the inter-VM traffic, and the number of VMs of future arrivals have been known, and it thus can make near optimal placement decision. As stated at the beginning of Subsection 8.2, unlike an off-line one, an online VM placement algorithm cannot foresee or predict the details of future arrivals. We thus use ϵ calculated using (21) to help the on-line algorithm estimate the future arrivals for making a better decision in VM consolidation. The main difference between the off-line algorithm and the on-line one is that δ_1 used in Algorithm 1 is known to both version, and δ_2 is known only to off-line version. Thus, in our design, the on-line algorithm uses ϵ to help estimate the future arrivals. Our simulation results show that using the idea of TClose with accurately selected ϵ, the

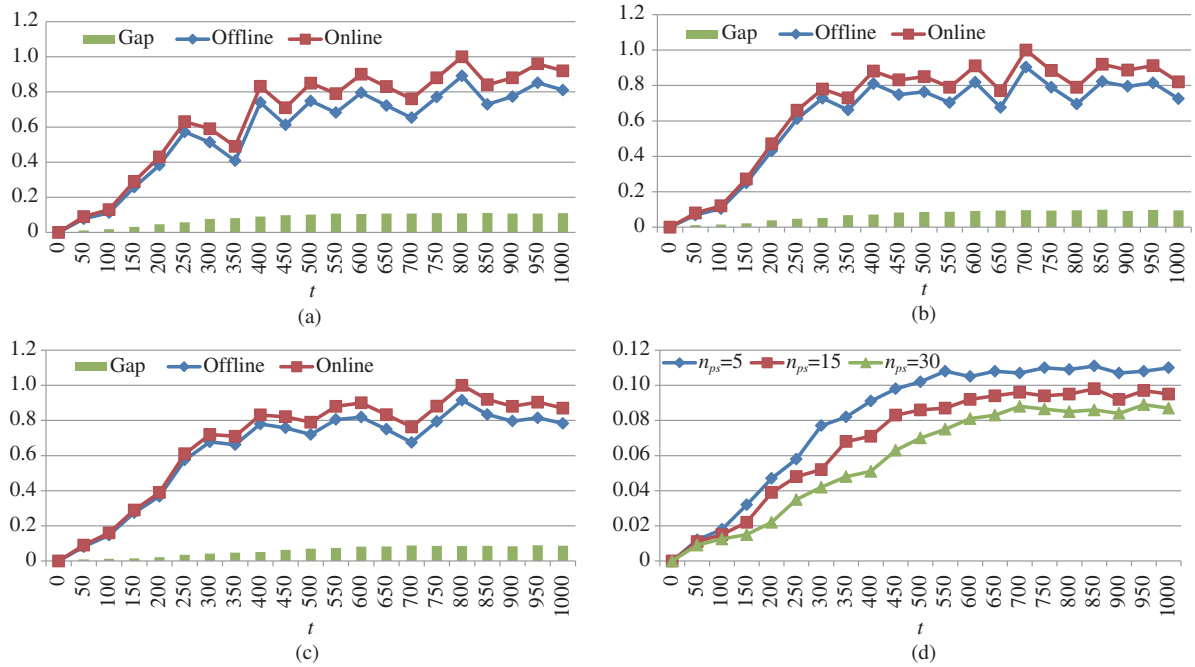

Figure 5 (Color online) NC gaps between ours and off-line algorithm with respect to different n_{ps} values. (a) $n_{ps} = 5$; (b) $n_{ps} = 15$; (c) $n_{ps} = 30$; (d) performance gaps with respect to different n_{ps} values.

performance of our on-line algorithm is very close to that of the off-line one in terms of network cost and energy consumption. In one of the simulations for network cost comparisons, the simulation parameters are set as $\lambda = 4$, numVM = 50, and $U = 24$ for both off-line and on-line version, and the simulations are performed for different n_{ps} values in which n_{ps} =5, 15, 30. The simulation results are shown in Figure 5. The results shown in the figure have been normalized using (22). The figure also shows the performance gap between our on-line algorithm and the off-line one, and one can see that the performance of our on-line placement scheme is very close to the performance of the off-line one, as the gap is less than 0.1 for different n_{ps} values. In another test, we let n_{ps}=15 and observe how different λ values' impact on the network costs would be. The simulation results are shown in Figure 6. One can see that the performance of our on-line scheme is very close to the performance of the off-line one, as the performance gap is also always less than 0.1 for different λ values.

We also compare the performance of our on-line scheme with the off-line one in terms of energy consumption. To reduce the energy consumption, the empty PMs can be set in sleeping mode. This way, the energy consumption of a DC is determined by the number of active PMs in the DC. The energy consumption is denoted by EC and can be measured by

$$\mathrm{EC} = \sum_{P \in \mathrm{aPMs}} E_P \times T^P_{\mathrm{run}},$$

where P is an active PM in DC, E_p is the energy consumption per unit time by each PM, and T^P_{run} is the run time of P. We also denote EC_r as the measurement of the ratio of the energy consumption of our on-line scheme to the energy consumption of the off-line one, i.e.,

$$\mathrm{EC}_r = \frac{\mathrm{EC}_{\mathrm{ours}}}{\mathrm{EC}_{\mathrm{off}}}. \tag{23}$$

The simulation results are shown in Figure 7. From the figure, one can see that in the time range [0–20], EC_r is almost equal to 1. In other words, the two algorithms are equally good. This is due to the fact that in the initial stage, there are not too many users' VMs to be freed, and the off-line algorithm thus does not gain any superiority. However, when time goes by, EC_r increases. This is because that as time goes by, there are more and more occupied slots are freed, and on the other hand, there are also

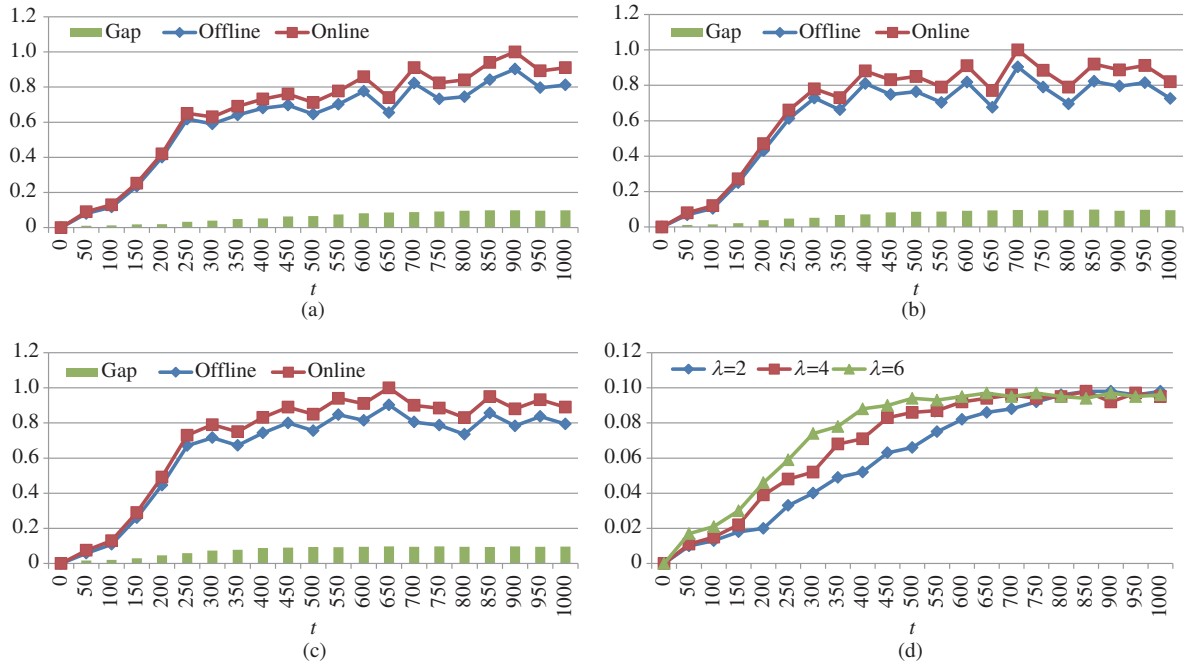

Figure 6 (Color online) NC gaps between ours and off-line algorithm with respect to different λ values. (a) $\lambda = 2$; (b) $\lambda = 4$; (c) $\lambda = 6$; (d) performance gaps with respect to different λ values.

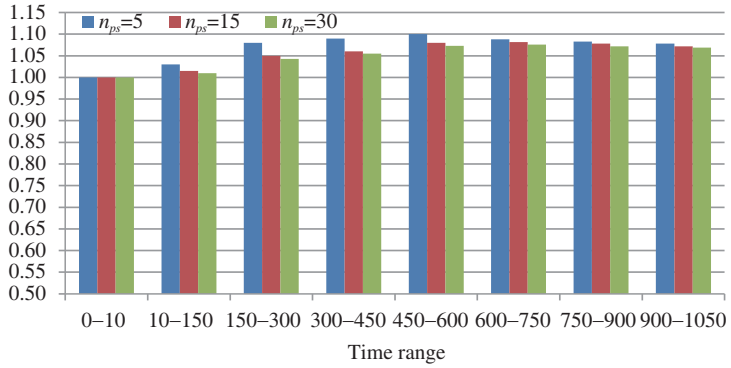

Figure 7 (Color online) EC_r between ours and off-line algorithm with respect to different n_{ps} values.

some new arriving VMs. This way, the off-line algorithm gains its superiority, as the off-line algorithm possesses complete information of the end time of all of the VMs including the current ones and the future arrivals. However, our on-line scheme uses TClose and the right choice of ϵ to offset the inferiority of not being able to foresee the new arrivals. Therefore, the performance gaps between our on-line scheme and the off-line one are always confined within 0.11 for different n_{ps} values in any time range.

8.3 Compared with best fit decreasing and improved best fit decreasing

It is difficult to make a straightforward comparison of the performances of those existing VM consolidation schemes and ours due to the following reasons. First, most of the existing VM consolidation algorithms are off-line scheme. Second, some on-line schemes take a set of independent VMs as input without taking the inter-VM traffic into account for making consolidation decision. Third, those on-line schemes either evaluate their schemes' performance using different performance metrics or in different experimental settings, and those studies were conducted under different assumptions or different constraints. Meanwhile, best fit decreasing (BFD) has been proven to be an effective greedy method for obtaining near optimal solutions for some NP-hard problems such as bin packing problem and VM consolidation [19,42]. Therefore, to verify that our VM consolidation scheme outperforms other on-line schemes in terms of both

network cost and energy consumption, we compare our scheme with BFD scheme and improved best fit decreasing (IBFD) scheme. Both schemes considered are on-line version. Since the traditional BFD takes independent VMs as input without considering the inter-VM traffic (which should be the case in reality), we revise BFD to be the one that takes linked VMs as input and also takes inter-VM traffic into account when making VM consolidation decisions. We name this improved version as IBFD, shown in Algorithm 3.

Algorithm 3 IBFD(G) //Improved best fit slot-decreasing algorithm

Input: G, graph of user's linked VMs;
Output: The placement result of G;

while G is not empty **do**
  Order the active PMs aPM[] in non-increasing order in terms of PMs' free slot numbers;
  par = false;
  n = max number of free slots in aPM[];
  while par $\neq$ true **do**
    VS = NodeExcision(G, n); // Invoke Algorithm 4.
    Get aPMset from aPMs whose free slot number is n;
    if $\exists$aPM $\in$ aPMset s.t. VMset VS can be placed in it and obeys inequation (11) **then**
      Find an aPM s.t. VS^{out} is the closest to B_r(aPM);
      Place VS in aPM;
      par = true;
      $G = G \backslash$VS; //Delete VS from G.
    else
      $n--$;
    end if
    if $n = 0$ **then**
      Active a new PM whose free slot number is n_{ps};
      $n = n_{ps}$;
    end if
  end while
end while

Algorithm 4 NodeExcision(G, n) //Greedy algorithm for excising n nodes from G

Input: G, a graph of linked VMs;
Output: a VMset of size n;

VMset = $\emptyset$;
$n_G = 0$;
Find a link $e(v_i, v_j)$ with the largest weight in G;
VMset = VMset $\cup v_i \cup v_j$;
$n_G = 2$;
while $n_G < n$ **do**
  $v_k = \max\{v_a | \sum_{v_b \in \text{VMset}} e(v_a, v_b), v_a \notin \text{VMset}\}$;
  VMset = VMset$\cup v_k$;
  n_G++;
end while
return (VMset);

As our on-line scheme makes VM consolidation decisions by taking VMs' end time and the topology of DC architecture, including inter-PMs and inter-racks traffic, into account, we will show that our scheme far outperforms BFD and IBFD in terms of both network cost and energy consumption. Per the discussion in previous sections, the degree of free slot concentration during the course of VM placement affects the network cost and also the energy consumption. Let SlotToPm = $\frac{N_{fs}}{N_{fp}}$, where N_{fs} is the total number of free slots in DC and N_{fp} is the total number of PMs with free slots. Then, SlotToPm reflects the degree of free slot concentration in DC. In the comparison of different performances of BFD, IBFD, and our scheme, we first compare the performance of our scheme with the performances of BFD and IBFD in terms of SlotToPm. Based on the insights we gain from our study, we believe that the higher degree of free slot concentration is, the more inter-VM traffic will be absorbed in the placement of future

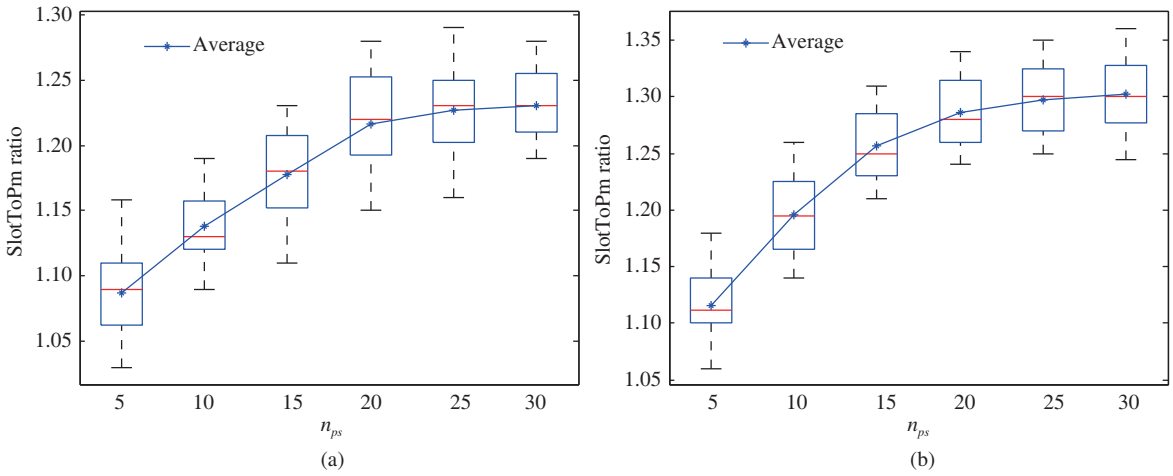

Figure 8 (Color online) SlotToPm ratios of our scheme to IBFD (a) and BFD (b), respectively, with different n_{ps} values.

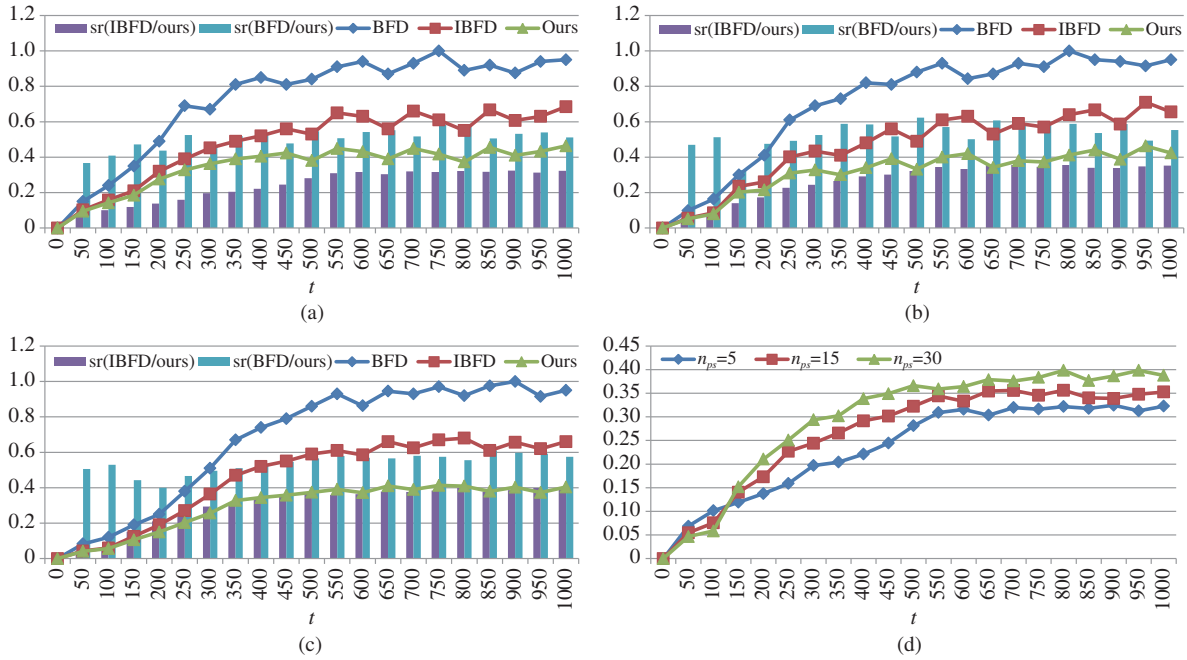

Figure 9 (Color online) Saving ratios sr of NC with respect to different n_{ps} values. (a) $n_{ps} = 5$; (b) $n_{ps} = 15$; (c) $n_{ps} = 30$; (d) sr(IBFD/ours) with respect to different n_{ps} values.

arrivals, thereby reducing both the network cost and the energy consumption during the entire course of VM consolidation. The results shown in Figure 8 are calculated by $\frac{\text{SlotToPm(ours)}}{\text{SlotToPm(IBFD)}}$ and $\frac{\text{SlotToPm(ours)}}{\text{SlotToPm(BFD)}}$. In the figure, each box plot is drawn based on 30 sets of simulation results for each n_{ps} value. As one can see from Figure 8 that the ratio is always more than 1, which verifies that the degree of free slot concentration of our scheme is higher than that of IBFD scheme and the concentration degree of IBFD is in turn higher than that of BFD scheme. When n_{ps} goes larger, the ratio increases accordingly. This is because with more free slots in a PM, there is better chance to free more slots from the same PM in our scheme, and thus the degree of free slot concentration is higher. This phenomena also explains the results shown in Figure 9.

In our simulations, we compute the saving ratio (sr) using

$$\text{sr(IBFD/ours)} = \frac{\text{NC}_{\text{IBFD}} - \text{NC}_{\text{ours}}}{\text{NC}_{\text{ours}}}, \tag{24}$$

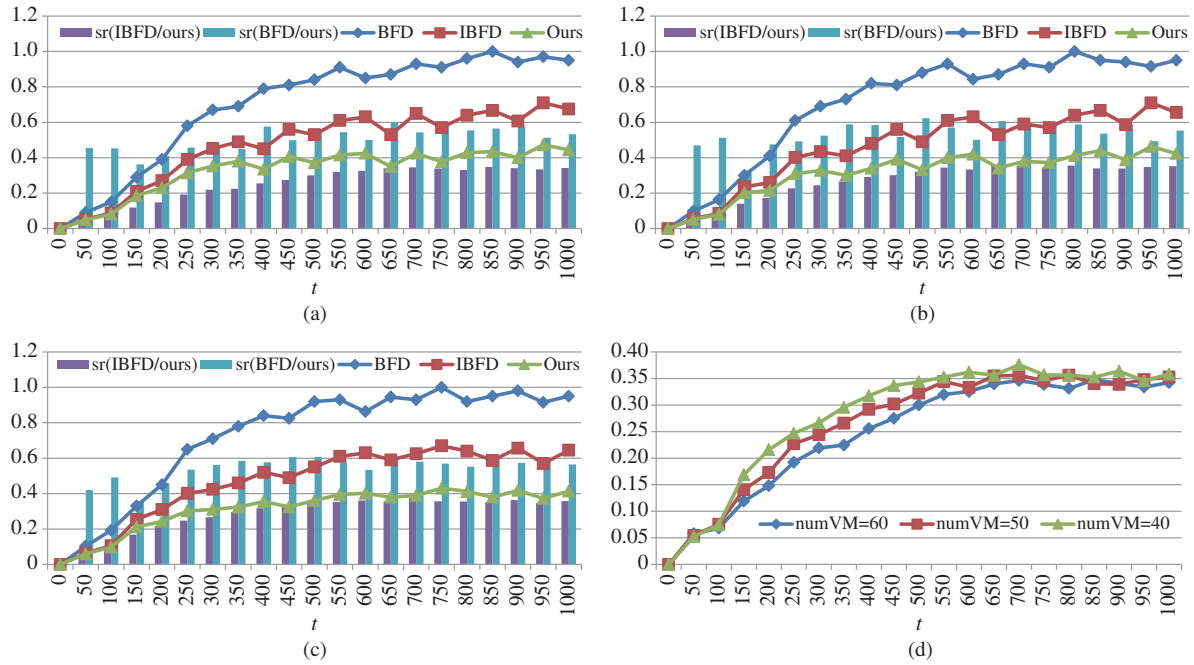

Figure 10 (Color online) Saving ratios sr of NC with respect to different numVM values. (a) numVM = 40; (b) numVM = 50; (c) numVM=60; (d) sr(IBFD/ours) with respect to different numVM values.

$$\mathrm{sr(BFD/ours)} = \frac{\mathrm{NC_{BFD}} - \mathrm{NC_{ours}}}{\mathrm{NC_{ours}}}, \tag{25}$$

where $\mathrm{NC_{IBFD}}$, $\mathrm{NC_{BFD}}$, and $\mathrm{NC_{ours}}$ represent the network cost of IBFD, BDF, and our scheme, respectively. Thus, sr serves as an index showing the superiority of our algorithm compared with IBFD and BFD. The higher value sr is, the more superior our algorithm will be. We conduct the experiments to observe the network costs with respect to different n_{ps}. The simulation parameters include $\lambda = 4$ and numVM = 50. Figure 9(a) shows that when $n_{ps} = 5$, in terms of network cost, our scheme outperforms IBFD by at least 30%, and outperforms BFD by at least 50%. Also, Figure 9(c) shows that when $n_{ps} = 30$, in terms of network cost, our scheme outperforms IBFD by almost 40% for $t \geqslant 600$, and outperforms BFD by 60% for $t \geqslant 500$. In deed, Figure 9(d) reveals that when n_{ps} increases, sr also increases accordingly. This is because our scheme takes end time into account when making the consolidation decision, and when there are more slots in a PM, there will be better chance to have higher degree of free slot concentration, and then there will be better chance to host those new arriving VMs with the same end time. This means that it can absorb more inter-VM traffic. Thus, intuitively, a PM with higher n_{ps} in general can save more network cost if using our consolidation scheme. From Figure 9(d), one can see that the performance gap between the test results of $n_{ps} = 5$ and that of $n_{ps} = 30$ is around 0.1 when $t \geqslant 250$.

We also perform a simulation to observe the effect of different numVM on NC with $n_{ps} = 15$ and $\lambda = 5$ as simulation parameters. We conduct three tests for numVM = 40, 50, and 60 and compare our scheme with IBFD and BFD. The simulation results are shown in Figure 10. In Figure 10, numVM is the mean number of VMs that a user requests, and each dot is the average of the results of 30 runs. In order to study how the current placement affects the subsequent placements, we set different numVM values. We calculate sr at 10 moment times shown in the abscissas of the figure. Though the results shown in the figure do not indicate that numVM has much effect on network cost saving, the results do show that as time goes by, the sr increases, and after $t \geqslant 600$, there are more than 35% network cost saving for any numVM values.

We also test the effect of different λ values on network cost. However, Figure 11 shows that λ values have no significant effect on network cost. The reason is that in our scheme, when calculating TClose we set $\epsilon = \frac{0.69U}{\lambda}$, and thus tolerance time ϵ in inequation (17) adapts to different λ values. Nevertheless, with

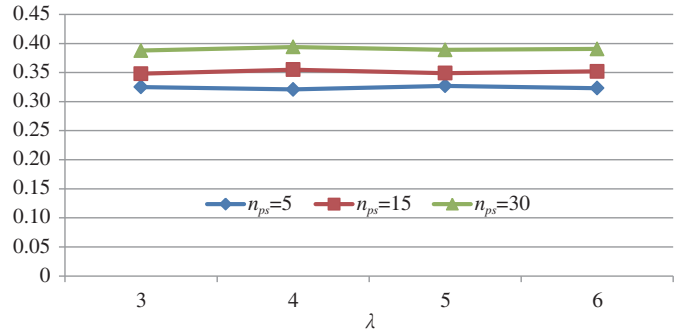

Figure 11 (Color online) Saving ratios sr of NC with respect to different λ and n_{ps} values.

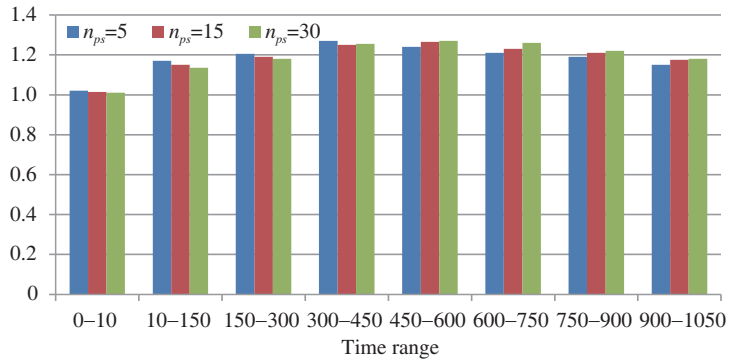

Figure 12 (Color online) EC_r with different n_{ps} values.

the same λ, when n_{ps} increases, sr also increases. In summary, by taking VMs' end time and inter-VM traffic into account and using the idea of TClose to devise our scheme, among numVM, λ, and n_{ps}, only n_{np} has significant effect on the network cost.

We also perform simulations to see how much energy savings our on-line scheme can gain compared with IBFD. The test results shown in Figure 12 are calculated by (26) using parameters numVM = 50 and $\lambda = 4$. The figure shows the energy consumption ratio of our scheme to IBFD with respect to different n_{ps} values. From the figure, one can see that to begin with, EC_r is very close to 1. This is because that during the initial stages, there were not too many slots released and the TClose mechanism adopted in our scheme cannot help gain much. However, when time goes by, there are more and more slots released and there are also more and more arriving VMs. In such scenario, TClose mechanism can help the placement algorithm with consolidation and produce more concentrated VM placement, thereby reducing the number of active PMs.

$$\mathrm{EC}_r = \frac{\mathrm{EC}_{\mathrm{IBFD}}}{\mathrm{EC}_{\mathrm{ours}}}. \tag{26}$$

Moreover, we also design a simulation environment using parameters numVM = 50 and $n_{ps} = 15$ to test the effect of different λ values on energy savings. The test results are shown in Figure 13. One can see that the influences of λ values on energy consumption are not very significant. In summary, our scheme outperforms IBFD on energy savings by 10%–25% under various scenarios.

9 Conclusion

Network bandwidth, though is an important resource, is one of the constraints in DCs. The increasing scale of cloud applications has made bandwidth increasingly becoming a bottleneck in the cloud. Virtualization-based DC is thus facing new challenges in the use of network resources. Particularly, the needs for users' VMs to communicate with each other consume a lot of network bandwidth. Meanwhile, energy consumption is another important issue in cloud computing. Fortunately, effective and efficient

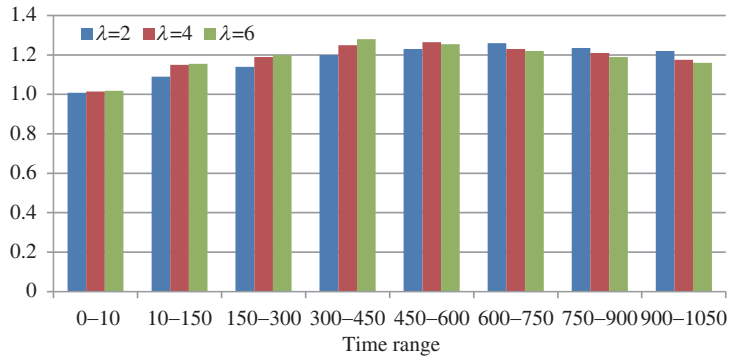

Figure 13 (Color online) EC_r with different λ values.

VM placement could largely reduce both network cost and energy consumption. This paper thus proposed an online traffic-aware VM Placement in cloud DCs. Our scheme makes a VM placement decision by taking not only the inter-VM traffic but also the VM end time into account. In particular, our VM placement is treated as a context-sensitive process in the sense that the current placement decision is made to have positive influence on the subsequent placements in terms of network cost savings and energy savings. Our simulations verified that our scheme could save more than 35% network cost and more than 10% energy consumption compared with IBFD scheme under various scenarios.

Acknowledgements This work was supported in part by National Key Research Development Program of China (Grant No. 2016YFB1000502), National Natural Science Foundation of China (Grant Nos. 61525204, 61732010), SJTU Overseas Visiting Scholars Program.

References

1 Chen R, Chen H B. Asymmetric virtual machine replication for low latency and high available service. Sci China Inf Sci, 2018, 61: 092110
2 Machida F, Kim D S, Park J S, et al. Toward optimal virtual machine placement and rejuvenation scheduling in a virtualized data center. In: Proceedings of IEEE International Conference on Software Reliability Engineering Workshops, 2008. 1–3
3 Kochut A. On impact of dynamic virtual machine reallocation on data center efficiency. In: Proceedings of IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, 2008. 1–8
4 Gao Y, Guan H, Qi Z, et al. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci, 2013, 79: 1230–1242
5 Hao F, Kodialam M, Lakshman T V, et al. Online allocation of virtual machines in a distributed cloud. IEEE/ACM Trans Netw, 2017, 25: 238–249
6 Deng W, Liu F, Jin H, et al. Reliability-aware server consolidation for balancing energy-lifetime tradeoff in virtualized cloud datacenters. Int J Commun Syst, 2014, 27: 623–642
7 Huang D, He B, Miao C. A survey of resource management in multi-tier web applications. IEEE Commun Surv Tutorials, 2014, 16: 1574–1590
8 Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Commun ACM, 2008, 51: 107–113
9 Xu F, Liu F, Jin H. Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud. IEEE Trans Comput, 2016, 65: 2470–2483
10 Xia M, Shirazipour M, Zhang Y, et al. Network function placement for NFV chaining in packet/optical datacenters. J Lightw Technol, 2015, 33: 1565–1570
11 Cohen R, Lewin-Eytan L, Naor J S, et al. Near optimal placement of virtual network functions. In: Proceedings of IEEE Conference on Computer Communications, 2015. 1346–1354
12 Meng X, Pappas V, Zhang L. Improving the scalability of data center networks with traffic-aware virtual machine placement. In: Proceedings of INFOCOM, 2010. 1–9
13 Guo Y, Stolyar A L, Walid A. Shadow-routing based dynamic algorithms for virtual machine placement in a network cloud. IEEE Trans Cloud Comput, 2018, 6: 209–220
14 Cisco. By 2014, cloud traffic will surpass traditional data center traffic. Cisco Whitepaper, 2011. http://www.cablinginstall.com/articles/2011/12/cisco-cloud-will-surpass-traditional-data-center.html
15 Bulk of data center traffic internal: Cisco. Cisco Whitepaper, 2011. https://insights.dice.com/2012/10/23/bulk-of-data-center-traffic-internal-cisco/
16 Guo C X, Wu H T, Tan K, et al. Dcell: a scalable and fault-tolerant network structure for data centers. SIGCOMM Comput Commun Rev, 2008, 38: 75

17 Fang W, Liang X, Li S, et al. VMPlanner: optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. Comput Netw, 2013, 57: 179–196
18 Wang M, Meng X Q, Zhang L. Consolidating virtual machines with dynamic bandwidth demand in data centers. In: Proceedings of INFOCOM, 2011. 71–75
19 Xu J L, Tang J, Kwiat K, et al. Enhancing survivability in virtualized data centers: a service-aware approach. IEEE J Sel Areas Commun, 2013, 31: 2610–2619
20 Cisco. Cisco ucs director administration guide, release 6.0, chapter: managing lifecycles. Cisco Whitepaper, 2011. https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/ucs-director/administration-guide/6-0/b_Cisco_UCSD_Admin_Guide_Rel60/b_Cisco_UCSD_Admin_Guide_Rel60_chapter_010000.html
21 Klempous R, Nikodem J. Innovative Technologies in Management and Science. Berlin: Springer, 2014. 10: 158–159
22 Quang-Hung N, Thoai N. Eminret: heuristic for energy-aware vm placement with fixed intervals and non-preemption. In: Proceedings of IEEE International Conference on Advanced Computing and Applications, 2015. 98–105
23 Alharbi F, Tain Y C, Tang M L, et al. Profile-based static virtual machine placement for energy-efficient data center. In: Proceedings of IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems, Sydney, 2016. 1045–1052
24 Usmani Z, Singh S. A survey of virtual machine placement techniques in a cloud data center. Procedia Comput Sci, 2016, 78: 491–498
25 Wang X, Xie H, Wang R, et al. Design and implementation of adaptive resource co-allocation approaches for cloud service environments. In: Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering. New York: IEEE, 2010. 2: 484–488
26 Le K, Bianchini R, Zhang J, et al. Reducing electricity cost through virtual machine placement in high performance computing clouds. In: Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2011. 22
27 Zhang X, Zhao Y, Guo S, et al. Performance-aware energy-efficient virtual machine placement in cloud data center. In: Proceedings of IEEE International Conference on Communications. New York: IEEE, 2017. 1–7
28 Mann Z A. Multicore-aware virtual machine placement in cloud data centers. IEEE Trans Comput, 2016, 65: 3357–3369
29 Bin E, Biran O, Boni O, et al. Guaranteeing high availability goals for virtual machine placement. In: Proceedings of the 31st International Conference on Distributed Computing Systems. New York: IEEE, 2011. 700–709
30 Yanagisawa H, Osogami T, Raymond R. Dependable virtual machine allocation. In: Proceedings of IEEE INFOCOM. New York: IEEE, 2013. 629–637
31 Zhou A, Wang S, Cheng B, et al. Cloud service reliability enhancement via virtual machine placement optimization. IEEE Trans Serv Comput, 2017, 10: 902–913
32 Yang S, Wieder P, Yahyapour R, et al. Reliable virtual machine placement and routing in clouds. IEEE Trans Parallel Distrib Syst, 2017, 28: 2965–2978
33 Wang S, Zhou A, Hsu C H, et al. Provision of data-intensive services through energy- and QoS-aware virtual machine placement in national cloud data centers. IEEE Trans Emerg Top Comput, 2016, 4: 290–300
34 Xu F, Liu F, Liu L, et al. iAware: making live migration of virtual machines interference-aware in the cloud. IEEE Trans Comput, 2014, 63: 3012–3025
35 Li X, Wu J, Tang S, et al. Let's stay together: towards traffic aware virtual machine placement in data centers. In: Proceedings of IEEE Conference on Computer Communications. New York: IEEE, 2014. 1842–1850
36 Li X, Qian C. Traffic and failure aware vm placement for multi-tenant cloud computing. In: Proceedings of IEEE 23rd International Symposium on Quality of Service. New York: IEEE, 2015. 41–50
37 Benson T, Anand A, Akella A, et al. Understanding data center traffic characteristics. In: Proceedings of the 1st ACM Workshop on Research on Enterprise Networking. New York: ACM, 2009. 65–72
38 Kandula S, Sengupta S, Greenberg A, et al. The nature of data center traffic: measurements & analysis. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. New York: ACM, 2009. 202–208
39 Andreev K, Racke H. Balanced graph partitioning. Theor Comput Syst, 2006, 39: 929–939
40 Garey M R, Johnson D S, Stockmeyer L. Some simplified NP-complete problems. In: Proceedings of the 6th Annual ACM Symposium on Theory of Computing. New York: ACM, 1974. 47–63
41 Ballani H, Costa P, Karagiannis T, et al. Towards predictable datacenter networks. SIGCOMM Comput Commun Rev, 2011, 41: 242
42 Breitgand D, Epstein A. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds. In: Proceedings of IEEE INFOCOM. New York: IEEE, 2012. 2861–2865