Cluster-Aware Virtual Machine Collaborative Migration in Media Cloud

Weizhan Zhang, *Member, IEEE,* Yuxuan Chen, Xiang Gao,Zhichao Mo, Qinghua Zheng, *Member, IEEE* and Zongqing Lu, *Member, IEEE*

Abstract—Media cloud has become a promising paradigm for deploying large-scale streaming media applications at a reduced cost. Due to dynamic and diverse demands of users, media cloud presents two crucial characteristics: high resource consumption and dynamic traffic among media servers. Consequently, Virtual Machine (VM) migration in media cloud is highly required to suit varying resource requirements and the dynamic traffic patterns. Moreover, migration of such bandwidth-intensive media applications in media cloud needs cautious handling, especially for the internal traffic of Data Center Networks (DCN). However, existing media cloud resource management schemes or traffic-aware VM deployment approaches are insufficient for media cloud, ignoring the characteristics of either cloud infrastructure or media streaming requirements. In this paper, we propose a cluster-aware VM collaborative migration scheme for media cloud, tightly integrating clustering, placement, and dynamic migration process. The scheme employs a clustering algorithm and a placement algorithm to obtain ideal migration strategies for newly perceived media server clusters, and a migration algorithm to effectively accomplish the migration process of media servers. Evaluation results demonstrate that our scheme can effectively migrate virtual media servers in media cloud, while reducing the total internal traffic in DCN under the resource consumption constraints of media streaming applications.

Index Terms—Cluster-aware, Virtual Machine Collaborative Migration, Media Cloud, Data Center Network.

1 Introduction

Media streaming service has become one of the most popular applications over the Internet. Nowadays, according to various emerging media applications and services, people tend to consume the dramatically increasing media contents. Thus streaming service providers are required to continuously increase capacity of their own physical resources. Meanwhile, cloud computing becomes a promising paradigm, which provides highly reliable and elastic resource provisioning in a cost-effective manner [1]. Cloud computing platform has become a fundamental infrastructure providing a shared pool of computing, storage and network resources with a modest cost. Therefore, media cloud naturally arises from the integration of media streaming and cloud computing, which becomes the most effective way for providers to deploy their large-scale media streaming services.

Accommodating large-scale media services and applications, media cloud usually has two typical characteristics: high resource consumption and dynamic traffic among media servers. Firstly, media processing and transcoding need to suffice the dramatically increasing trends of media content and consumption, which tend to highly utilize CPU and memory resources in media cloud. While, both prerecorded and live media streaming require high demands of bandwidth resources to accommodate peak loads. Secondly, owing to diverse and dynamic user behaviors and demands, network traffic among media servers is volatile and erratic. For example, due to dynamic and unpredictable burst of users' requests for live streaming, the operation of a single node brings heavy load, and thus we need a number of nodes to do parallel processing. Some nodes are in charge of video transcoding tasks, some are responsible for streaming tasks, and others focus on media analytics, sharing and delivery tasks. After the processing of each node, the results will be aggregated with synchronous integration or sequential composition and then sent to the user. Therefore, media server clusters are required to cooperate based on the dynamic traffic among media servers. Especially, this research is motivated from managing the dynamic media server cluster in media cloud for large-scale multimedia elearning services. Notice that the outcomes of the research may also benefit different application domains with dynamic traffic in cloud environment. For example, both of content delivery servers for web services and master-slave computing cluster for Hadoop applications can benefit from the clustering-based migration, which share storage or servers.

Because of the aforementioned two characteristics of media cloud, the dynamic and unpredictability of the user's requests for resources are inevitable. Original media server cluster placement may be inappropriate due to up-to-date resource requirements and varying traffic patterns. High resource consumption may overload the physical servers. Dynamic traffic among media servers may cause traffic congestion in DCN and degrade network performance. And it is not easy to accurately predict the dynamic resource requirements and varying traffic patterns in fairly large DCN. Therefore, VM migration in media cloud is highly required. Unfortunately, ordinary migration approaches may

Weizhan Zhang, Yuxuan Chen, Xiang Gao, Zhichao Mo, and Qinghua Zheng are with the MOEKLINNS Lab, Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, 710049 China, email: {zhangwzh@, zxcvghjkmnb@stu, gaox24l@stu, mozhichao@stu, qhzheng@}.xjtu.edu.cn, Fax: 86-29-82663860, Phone: 86-13325450992. Zongqing Lu is with the Department of Computer Science and Engineering, Pennsylvania State University, PA, 16802 USA, email: zongqing@cse.psu.edu

not work in media cloud and may cause unexpected and unacceptable results. For example, media server migration for a video on demand application will bring extra network loads to some links of DCN, resulted from the frequent communication between media servers and storage servers. Moreover, inappropriate reallocation of a media server for live streaming to an overloaded host may inadvertently cause significant degradation for application performance.

Moreover, internal bandwidth has become the crucial resource of cloud infrastructures and is prone to be the bottleneck of DCN [2], and Guo et al. [3] pointed out that internal traffic in DCN has reached 80% of the total traffic. Naturally, migrating traffic-intensive applications requires cautious handling because correlated VMs of such applications exchange lots of data and thus generate massive internal traffic in DCN. For example, some real-time live streaming services, such as video conferencing and virtual classroom, may employ a tree-based media server cluster. They usually let root VM to stream the video to other VMs in order to provide real-time large-scale streaming services. In this situation, the traffic between the root node and the leaf nodes is the major reason for internal traffic in the data center. The delivery tree needs cautious maintain to reduce the internal traffic. Therefore, at the scenario of VM migration in media cloud, it is of great importance to optimize the intra-bandwidth consumption in DCN while considering the characteristics of both cloud infrastructure and media streaming.

However, existing work about migration in media cloud is deficient. On one hand, related works about media cloud is primarily from the perspective of how the cloud on demand can accommodate media streaming applications [4, 5, 6]. These researches focus on reasonable allocation of cloud resources for media servers from application layers. They try to satisfy temporal and spatial dynamics of demands from media streaming users, and also to fulfill the features of resource management in cloud platform. Although these researches have considered the resource allocation among different VM-hosted servers, they do not take network traffic as an input for the placement and migration problem, which may result in high network traffic on the core links of the data center. In order to decrease the networking cost for VM migration, traffic patterns of VMs need to be taken into account under the DCN infrastructure of media cloud.

On the other hand, existing researches about VM migration concentrate on the approaches of placing and migrating VMs in cloud environment, to maximize the cloud utility [7], to minimize the energy consumption [8], or to optimize various migration costs [9, 10, 11]. However, they are not suitable for media cloud environment, because they ignore the characteristics of high resource consumption and dynamic traffic among media servers, which may lead to unnecessary network traffic within DCN, even crush the media services. Although several researches consider the combination of network traffic among VMs and the capacity limits of server sides [12, 13, 14], those traffic-aware VM placement or migration researches do not demonstrate a clear and efficient migration mechanism. Therefore, we can see there is a huge gap between media cloud computing and VM migration.

In this paper, we propose a cluster-aware collaborative VM migration scheme to satisfy resource constraints and reduce the internal traffic of DCN in media cloud. We assume that media servers have already been deployed, providing various resource-intensive media streaming applications concurrently. Especially, communication among these media servers will bring dynamic traffic in DCN, but the cluster relationships among media servers are not perceived initially. The only precondition required from the media cloud is the dynamic communication traffic information among virtual media servers in the given data centers. In this manner, we provide a generalized solution for the whole media cloud shared by concurrent applications. Firstly, we design a two-tier clustering algorithm that achieves the cluster recognition based on the dynamic traffic patterns among VMs. The algorithm forms two types of VM clusters: host-oriented clusters that group VMs that communicate the most in order to place them on the same physical server, and partition-oriented clusters that group host-oriented clusters that communicate the most in order to eventually place physical servers that exchange more traffic on the same switch. These two types of clusters are tailored to satisfy the resource requirements of physical hosts and DCN partition. Secondly, we design a placement algorithm that obtains an ideal migration destination for each media cluster in DCN. The algorithm determines the cluster placement, while considering the cost of migrating a host-oriented cluster to another host and migrating a partition-oriented cluster to another partition. Finally, we propose a collaborative VM migration algorithm. The algorithm performs the migration process, trying to find the maximum concurrence degree while avoiding network link congestion, which further optimizes the internal traffic of DCN in media cloud and reduces the total migration time. Evaluation results demonstrate that our scheme can effectively migrate virtual media servers in media cloud, while reducing the total internal traffic in DCN under the resource consumption constraints of media streaming applications.The scheme can roughly reduce half internal traffic compared with the traditional load balance VM placement, and it also outperforms the traffic-aware VM placement benchmarks under different settings.

The main contribution of this paper is summarized as follows:

- The proposed scheme does not concentrate on deploying a special application in media cloud. It targets at media cloud shared by concurrent media applications, and provides a generalized solution to solve the problem of VM migration in media cloud, tightly integrating clustering, placement, and dynamic migration process.
- The proposed placement algorithm is migration-oriented. Notice that existing traffic-aware VM placement algorithms usually do not consider the migration costs. Furthermore, the placement algorithm considers the server-side constraints of media servers, and can be carried out without predefined clusters. In this manner, the placement algorithm, together with the clustering algorithm, chooses appropriate locations for media servers in media cloud, so as to

reduce the upcoming communication and migration costs.

- The proposed migration algorithm takes the perceived cluster as the migration target and provides a synergistic collaboration solution. The migration algorithm fulfills a competition-reduced parallel migration of VMs to reduce the migration time, and considers the network gains at each migration step to further cut down the migration costs.

The remainder of the paper is organized as follows. Section 2 provides a brief overview of related work, and Section 3 gives the model and problem formulation. Section 4 provides a detailed description of the proposed scheme, and Section 5 evaluates the performance of the proposed scheme. Section 6 concludes the paper with suggestions for future work.

2 Related Work

2.1 Traffic-Aware VM Placement

VM placement problem is often described as a multi-dimensional vector packing problem [15], and its approximate solution can be obtained by a greedy algorithm. The optimization objectives mainly are: resource utilization [12, 16, 17, 18, 19], network overhead [12, 16, 20, 21] and energy consumption [22, 23, 24, 25]. The constraints mainly are: service-level protocol [5, 16, 26] and application correlation [12, 27]. The scenarios include static placement and dynamic placement.

Recently, researchers have proposed many traffic-aware VM placement strategies to decrease network traffic in data centers. For example, Shrivastava et al. [12] proposed a scheme called AppAware to rebalance workloads across physical machines by shifting overloaded VMs to underloaded physical machines. The goal is to minimize the network traffic while satisfying all of the server-side constraints. Meng et al. [28] formulated the traffic-aware VM placement problem and proposed an approximation algorithm called Cluster-and-Cut to find clusters by computing virtual-to-physical machine mappings in the presence of application dependencies. Tevfik et al.[13] proposed a traffic-aware VM placement algorithm to decrease the traffic between racks. Similarly, based on the traffic patterns among VMs and the network topology, Dias et al. [14] proposed a placement algorithm called VMP to mitigate the problem of traffic concentration in DCN. VMP places VMs with tight communication to the nearest location by analyzing the amount of traffic exchanged, and aggregates VMs into clusters by searching communities in the traffic. Finally, VMP fits all the clusters into the qualified partitions by a heuristic algorithm. Dong et al. [29] proposed a VM placement scheme meeting multiple resource constraints, such as the physical server size and network link capacity to improve resource utilization. Zhang et al. [24] proposed an energy aware virtual network embedding scheme to reduce energy consumption, which considers both of the node and link requirements.

Although using internal traffic as a constraint, both AppAware and Cluster-and-Cut [12, 28] cannot apply to an environment without predefined clusters. Also, Cluster-and-Cut [28] ignores the server-side constraints. Some related VM placement researches [13, 14, 29] take the cluster placement into account, but they do not consider the migration costs. Thus, the algorithm is too ideal to achieve the dynamic migration process, which is unacceptable in a real large-scale data center environment. Some novel virtual network embedding and migration schemes are proposed [24, 25], which consider both of the node and network constraints. However, the proposed schemes also do not consider the migration costs in advance when they decide the target positions of embedding or migration. Therefore, these works cannot effectively solve the problem of VM placement under a DCN architecture of media cloud.

2.2 Schedule of VM Migration

The essential of VM migration is the scheduling sequence of VMs, which can be roughly categorized as serial migration and parallel migration. The purpose is to find a suitable sequence to reduce various migration costs, such as migration time [9], transmission data [10], and virtual machine downtime [9, 11].

Serial migration achieves migration of VMs based on certain strategies of migration priority. For example, Wood et al. [7] proposed a greedy migration manager in Sandpiper based on a migration queue, which is in accordance with the descending order of the total migration time. However, the serial migration naturally is not an efficient way to provide large-scale migration of VMs. Therefore, parallel migration is more preferred to solve the migration problem in a large scale of VMs. Nevertheless, unreasonable sequences of parallel migration can induce a large amount of bandwidth competition among VMs. This not only extends the migration time, but also causes the network congestion, even effects the service performance of VMs [30]. Especially, when the network bandwidth becomes the main bottleneck during the migrations, the migration time increases dramatically [31]. Therefore, in the last few years, there have been many researches focusing on scheduling strategies of parallel VM migration for different optimization objectives. Chen et al. [30] proposed a network-bandwidth sharing strategy to allocate reasonable bandwidth resources for all links to reduce the migration costs. Liu et al. [32] proposed a coordinate VM migration scheme called VMbuddies, which is one of the earliest researches to solve the correlated migration problem of multi-tier application in cloud environments. The scheme migrates tightly-coupled VMs simultaneously, and minimizes the migration costs of multi-tier applications.

Although considering bandwidth resources for parallel migration, the proposed bin-packing algorithm [30] neglects the intensive traffic between VMs which significantly affects bandwidth resources for all network links, thus the algorithm may lead to tremendous pressure and undesirable effects on the underlying infrastructure. VMbuddies [32] mainly focuses on the bandwidth resource constraints for the correlated VM migration, but does not consider the clustering and the placement issues in media cloud. Therefore, we can see that the existing researches cannot directly apply to the VM migration problem in media cloud.

3 Model and Problem Formulation

In the context of VM migration in media cloud, the internal network traffic in DCN mainly derives from two parts: communications among VMs and migrations of VMs. Firstly, we define the internal traffic cost of communications among VMs as C_{com}, which is closely related to the traffic among VMs and the numbers of switches on the path between VMs. Suppose that the network topology of VMs can be described as a weighted undirected graph $G(S, V, E, w)$. The vertex set S in G is the switch topology in DCN, the vertex set V in G indicates the VM set of media severs and E denotes the set of edges between connected VMs. w indicates the edge weight between the connected pair of vertices. That is, $w(v_i, v_j)$ denotes the traffic flow between VM v_i and v_j. Moreover, in order to guarantee the fundamental service quality of media streaming, we consider three most representative constrained resources for media servers: CPU (measured by the number of cores), memory, and network bandwidth. In the process of finding the right spot for each cluster of media servers, the cluster can only be instantiated on physical host h_j which has enough CPU, memory and bandwidth resources to handle all the VMs designated to it. To be specific, we use P, R and B to represent the CPU, memory and bandwidth resources of one physical host, respectively. Note that different capacities of host physical machines can also be fit into our scheme. Also, $p(i)$ is the required CPU resource of VM v_i, $r(i)$ is the required memory resource, and $b(i)$ is the required bandwidth resource.

Secondly, as for the VM migration, the operating system firstly needs to copy the original memory. If the memory page of one VM has been modified when the VM is being migrated, the operating system must preserve the contents of that page so that it can be accessed and migrated later, and we call those preserved pages as dirty pages. This process is continuously iterated to a certain round, where the number of dirty pages production is less than a given threshold. Undoubtedly, this process will lead to massive traffic in DCN. Thus the internal traffic cost of VM migrations, defined as C_{mig}, is highly related to the transmitted migration data and also the numbers of switches on the migration path between VMs.

Our goal is to minimize the internal network traffic in DCN with these server-side constraints. Therefore, the problem can be formulated as:

$$Min(C_{com}, C_{mig}) \ \ s.t. \ \forall h_k, \begin{cases} \sum\limits_{v_i \in h_k} p(i) \leq P(k) \\ \sum\limits_{v_i \in h_k} r(i) \leq R(k) \\ \sum\limits_{v_i \in h_k} b(i) \leq B(k) \end{cases} \tag{1}$$

where

$$C_{com} = \sum_{i=1}^{n} \sum_{j=1}^{n} w(v_i, v_j) \times H_{com}(v_i, v_j)$$

$$C_{mig} = \sum_{i=1}^{n} m(v_i) \times H_{mig}(v_i, p_j)$$

Here, n is the number of VMs. $H_{com}(v_i, v_j)$ represents the numbers of switches on the path from VM v_i to v_j, and $m(v_i)$ represents the transmitted migration data by scheduling VM v_i. $H_{mig}(v_i, p_j)$ represents the numbers of switches on the migration path from VM v_i to partition p_j, where partition represents all servers accommodated by one access switch in DCN.

From (1), we note that the optimization consists of two parts: C_{com} and C_{mig}. In the part of C_{com}, $w(v_i, v_j)$ is the indispensable application layer data communication between VMs to provide media streaming service, so we cannot decrease it. $H_{com}(v_i, v_j)$ is associated with the placement of VMs, we can optimize it by grouping VMs that communicate the most through cluster-aware collaborative migration scheme, to reduce the number of switches on the path between the VM pairs. The scheme designs a two-tier clustering algorithm that achieves the cluster recognition based on the dynamic traffic patterns among VMs. Notice that, in our scheme, the traffic topology among virtual media servers in the given data centers is the only precondition required from the media cloud. As for the part of C_{mig}, similarly, we can choose the migration destination for each virtual media server to reduce $H_{mig}(v_i, v_j)$ from dynamic VM migration. Moreover, parallel migration induces more link competitions and bring more transmitted migration data from the dirty page generations, which may lead to the degradation of the available link bandwidth for subsequent VM migrations. Therefore, by calculating the network gain for migrating each VM, we prioritize VM migrations with larger network gain in DCN to increase the available link bandwidth for subsequent VM migrations. In this manner, we can reduce the total migration data from the dirty page generations, which helps to optimize C_{mig} further. In the proposed two objective model, decreasing C_{com} will reduce the inter-VM traffic with permanent effect, while decreasing C_{mig} will reduce the intra-server traffic with temporary effect. Thus, we pay much attention to C_{com}. In detail,we use the clustering algorithm to reduce the communication costs associated with C_{com}, then we take the output of VMs clustering result as the input of the placement algorithm, which determines the placement of VMs based on the migration cost. Finally, the migration algorithm actually executes the migration. The traditional VM placement aims to select the target host, which can be seen as a knapsack problem. It is well known that even the simplest form of knapsack problems is NP-hard. Meng et al. [28] had shown us that the problem of traffic aware VM placement is NP-complete, without considering the server-side constraints. We further consider the actual migration process to solve a correlated VM migrations problem [33], which makes the scheme has stronger constraints and more difficult to solve. Therefore, our problem is also a NP-complete problem. Due to the limited space, we omit the proof in our paper. To solve the problem, in this paper, we propose a cluster-aware VM migration scheme for media cloud. Notice that the proposed scheme is aiming to reduce the networking cost of data centers, regardless of whether the client and management network are separated or not.

4 Cluster-Aware VM Collaborative Migration Scheme for Media Cloud

The scheme includes three components: a clustering algorithm, a placement algorithm and a migration algorithm. The clustering algorithm identifies clusters, the placement algorithm determines the migration position for the clusters, and the migration algorithm actually carries out the migration process. Notice that the clustering algorithm and the placement algorithm are strategy-oriented algorithms, they do not migrate the VMs but decide the objects and destinations of migration, respectively. The clustering algorithm identifies host-oriented clusters hc and partition-oriented clusters pc for the facility of the placement algorithm, since the placement algorithm needs to consider the infrastructure of DCN. The placement algorithm chooses the destination of migration considering the migration cost for deducing the real migration cost of the migration algorithm. In this manner, the scheme tightly integrates clustering, placement, and dynamic migration process. The clustering and placement algorithm obtains ideal migration strategies for newly perceived media server clusters, and the migration algorithm accomplishes the migration process of media servers, which is designed to reduce both the migration time and migration data, while achieving parallel migration without potential link competitions. colorblueThe overview of the scheme can be illustrated in figure 1.

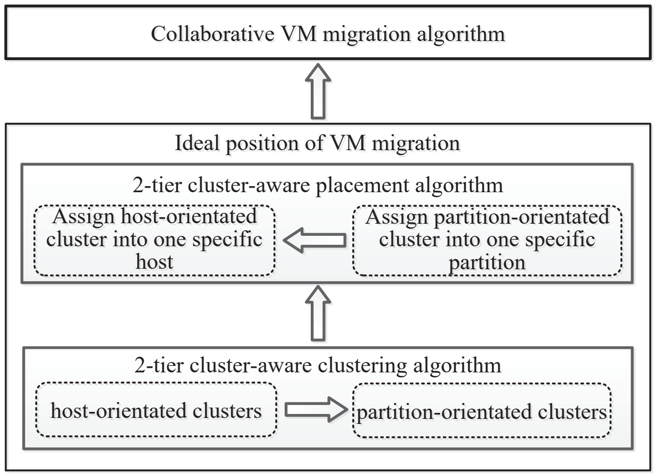
Collaborative VM migration algorithm
Ideal position of VM migration
2-tier cluster-aware placement algorithm
Assign host-orientated cluster into one specific host
Assign partition-orientated cluster into one specific partition
2-tier cluster-aware clustering algorithm
host-orientated clusters
partition-orientated clusters

Fig. 1: The overview of the proposed scheme.

4.1 Two-tier cluster-aware clustering algorithm

Being such a traffic-intensive application, media streaming requires a fine granularity clustering method to consolidate VMs with frequent communications. Therefore, we propose a 2-tier cluster-aware clustering algorithm, which contains host-oriented cluster clustering process and partition-oriented cluster clustering process. Referring to the main idea of the research from Yapicioglu et al. [13], our algorithm adopts 2-tier granularity of clustering. However, the placement algorithm of the referred research ignored the migration costs. The proposed algorithm has three preconditions: the topology of the data center, the traffic topology among all VMs, and the physical requirement of each VM. The existing topologies of the data center can be roughly categorized as hierarchical model, recursive model, or rack-to-rack model [34]. Regardless of the DCN architectures, virtual machines are generally connected to the physical machine. The traffic exchanged between each pair of VMs is used to create the traffic topology. According to the traffic topology, the algorithm models a weighted undirected graph. In the graph, we use vertices to represent VMs, and the edge weight to represent “how much“ one VM is connected to another. Obviously, the value is low when VMs are not much correlated, and the value is high when VMs share services that are traffic-intensive. Based on the traffic topology between VMs and the concept of community detection [35], the algorithm firstly identifies all the connected subgraphs which are part of the main graph containing all VMs. Each qualified subgraph is called as a “host-oriented“ cluster. Then, based on the traffic matrix between host-oriented clusters, the algorithm clusters the subgraphs to form bigger subgraphs, which is called as “partition-oriented“ clusters. To be specific, firstly, according to the traffic topology, the pair of VMs corresponding to the maximum flow is tried to be merged into cluster, while satisfying the server side constraints. Unless all VMs are clustered, all of the flows are processed. After clustering host-oriented clusters, internal traffic within partitions is minimized. Next phase is to minimize overall network traffic at the core and intermediate layer in DCN. Following the same method of clustering host-oriented clusters, we place the clusters communicating with each other more frequently into the same partition, so that communication path between partition-oriented clusters would be minimized.

Firstly, the algorithm merges VMs in G into multiple clusters $\{G_{hc}^1, G_{hc}^2, ..., G_{hc}^k\}$. Here, G_{hc} is the subgraph of G that represents the traffic topology of one host-oriented VM cluster hc. The host-oriented VM cluster groups VMs that communicate the most, in order to place them on the same physical machine in the future. We start to initialize each VM in G as a host-oriented cluster. Then according to the traffic pattern among VMs, the algorithm merges VMs with frequent communication into a bigger host-oriented cluster, as long as the cluster can accommodate server-side constraints. Thus the size of host-oriented cluster is highly-related to the CPU, memory and bandwidth resource of one physical machine. Our merging process aims to maximize the internal flow within clusters, and minimize the flow across clusters. So we introduce the traffic gain of merging clusters, denoted as ΔQ_{hc}^{mn}. Therefore, when determining whether to merge two host-oriented clusters hc_m and hc_n into a bigger host-oriented cluster hc_k, the algorithm calculates ΔQ_{hc}^{mn} as following:

$$\Delta Q_{hc}^{mn} = \begin{cases} \dfrac{\Big(\sum\limits_{v_i, v_j \in G_{hc}^k} w(v_i, v_j)\Big) * w(hc_m, hc_n)^2}{M_1 \Big(\sum\limits_{v_i, v_j \in G_{hc}^m} w(v_i, v_j)\Big)\Big(\sum\limits_{v_i, v_j \in G_{hc}^n} w(v_i, v_j)\Big)}; & M_1 \leq 100\% \\ 0; & M_1 > 100\% \end{cases} \tag{2}$$

$$M_1 = max\Big\{(\sum_{v_i \in G_{hc}^k} p(i))/P, (\sum_{v_i \in G_{hc}^k} r(i))/R, (\sum_{v_i \in G_{hc}^k} b(i))/B\Big\}$$

subject to:

$$\begin{cases} \sum\limits_{v_i \in G_{hc}^k} p(i) = \sum\limits_{v_i \in G_{hc}^m} p(i) + \sum\limits_{v_i \in G_{hc}^n} p(i) \\ \sum\limits_{v_i \in G_{hc}^k} r(i) = \sum\limits_{v_i \in G_{hc}^m} r(i) + \sum\limits_{v_i \in G_{hc}^n} r(i) \\ \sum\limits_{v_i \in G_{hc}^k} b(i) = \sum\limits_{v_i \in G_{hc}^m} b(i) + \sum\limits_{v_i \in G_{hc}^n} b(i) \\ \sum\limits_{v_i, v_j \in G_{hc}^k} w(v_i, v_j) = \sum\limits_{v_i, v_j \in (G_{hc}^m \bigcup G_{hc}^n)} w(v_i, v_j) \end{cases}$$

Here, the unit of M_1 is %, and M_1 represents the maximum percentage of occupied physical resources (CPU, memory and bandwidth) after merging clusters, and $M_1 \leq 100\%$ aims to guarantee the server-side resource constraints for "host-oriented" clusters, which means each physical machine has enough CPU, memory and bandwidth resources to handle all of the virtual servers designated to it. $\frac{1}{M_1}\sum_{v_i,v_j \in G_{hc}^k} w(v_i, v_j)$ indicates the internal flow within hc_k per host resource unit. $w(hc_m, hc_n)$ is the traffic flow between two host-oriented clusters hc_m and hc_n, hence $w(hc_m, hc_n)/\big(\sum_{v_i,v_j \in G_{hc}^m} w(v_i, v_j)\big)$ indicates the distribution of traffic flow between clusters compared to the traffic within hc_m. Then, by determining whether $\Delta Q_{hc}^{mn} > 0$, we decide to implement the merging operation of two host-oriented clusters. If true, merge the cluster pair with maximum ΔQ_{hc}^{mn}, and then update all weights of edges related to hc_k in the whole traffic topology. Repeat the above operations until there is no $\Delta Q_{hc}^{mn} > 0$. After that, we continue to merge indirectly connected host-oriented clusters and the remaining isolated host-oriented clusters if physical resources are satisfied. The above steps help us to obtain host-oriented clusters $\{G_{hc}^1, G_{hc}^2, ..., G_{hc}^k\}$, which guarantee us that the internal traffic of a host-oriented cluster will be optimized in one host without passing through any switch.

Secondly, the algorithm merges multiple host-oriented clusters $\{G_{hc}^1, G_{hc}^2, ..., G_{hc}^k\}$ into bigger partition-oriented clusters $\{G_{pc}^1, G_{pc}^2, ..., G_{pc}^n\}$ as long as the partition can accommodate the host-oriented cluster. G_{pc} is also the subgraph of G which represents the traffic topology of one partition-oriented VM cluster pc. The partition-oriented VM cluster groups host-oriented clusters that communicate the most, in order to eventually place physical machines that exchange more traffic on the same switch in the future. The number of host-oriented cluster in one partition-oriented cluster depends on the port number of connected access switch. In details, the algorithm continues to take G_{hc} as input, and initializes hc in G_{hc} as one pc in G_{pc}. According to the traffic pattern among host-oriented clusters, the algorithm merges partition-oriented clusters with frequent communication. Similarly, determining whether to merge pc_m and pc_n into pc_k, the algorithm defines ΔQ_{pc}^{mn} for clustering G_{pc} as below:

$$\Delta Q_{pc}^{mn} = \begin{cases} \dfrac{(\sum\limits_{v_i,v_j \in G_{pc}^k} w(v_i, v_j)) * w(pc_m, pc_n)^2}{M_2(\sum\limits_{v_i,v_j \in G_{pc}^m} w(v_i, v_j))(\sum\limits_{v_i,v_j \in G_{pc}^n} w(v_i, v_j))}; & M_2 \leq 100\% \\ 0; & M_2 > 100\% \end{cases} \tag{3}$$

$$M_2 = max\Big\{(\sum_{hc \in G_{pc}^k} 1)/N, (\sum_{v_i \in G_{pc}^k} b(i))/L_{avi}\Big\}$$

subject to:

$$\sum_{v_i,v_j \in G_{pc}^k} w(v_i, v_j) = \sum_{v_i,v_j \in (G_{pc}^m \bigcup G_{pc}^n)} w(v_i, v_j)$$

Here, N is the number of servers in one partition, represented as the port number of connected access switch. L_{avi} is the available capacity of links between switches (especially for up-links), to guarantee the bandwidth limitation for G_{pc}. The unit of M_2 is %, and M_2 represents the percentage of occupied ports within one partition, and $M_2 \leq 100\%$ guarantees that the access switch has enough port to accommodate pc_k designated to it. $w(pc_m, pc_n)$ indicates the traffic flow between two partition-oriented clusters pc_m and pc_n. Similarly, by determining whether $\Delta Q_{pc}^{mn} > 0$, we decide to merge two partition-oriented clusters. If true, merge the cluster pair with maximum ΔQ_{pc}^{mn}, and then update all weights of edges related to pc_k in the traffic topology. Repeat the operations of merging until there is no $\Delta Q_{pc}^{mn} > 0$. Now, we have successfully obtained partition-oriented clusters $\{G_{pc}^1, G_{pc}^2, ..., G_{pc}^n\}$, which can guarantee the internal traffic of a partition-oriented cluster will be restricted in one partition and this part of traffic will only pass through one switch. According to the traffic gain, the two-tier cluster-aware clustering algorithm groups VMs with frequent communication as much as possible, which aims to decrease C_{com} by optimizing H_{com}.

The pseudocode for the algorithm is described in Algorithm 1.

4.2 Two-tier cluster-aware placement algorithm

The 2-tier cluster-aware placement algorithm decides of clusters placement, while considering the cost of migrations (migrating a host-oriented cluster to another host and migrating a partition-oriented cluster to another partition). It mainly contains of 2 steps: partition-oriented cluster placement and host-oriented cluster placement. The proposed algorithm works as follows: first the algorithm identifies all the partition-oriented cluster, and calculates the cost of migrating the cluster to each partition. Each partition-oriented cluster is tested to fit into one of the partitions while minimize the total migration cost. Two actions can happen here, if the partition can support it, the cluster is allocated into the partition, occupying one port of the partition and marking as placed clusters from the main graph. The second action happens if the cluster can not fit into any partition, forcing the algorithm to recheck the left clusters. Once all the partition-oriented cluster are tested, the first step is over. Then, similarly, the algorithm calculates the cost of migrating the host-oriented cluster to each host. Also, two actions

Algorithm 1 2-tier cluster-aware clustering algorithm

```
INPUT: Traffic topology of VMs $G(S, V, E, w)$.
Number of hosts in partition $N$, CPU resources of one host $C$, Bandwidth resources of one host $B$.
OUTPUT: Host-oriented cluster topology $G_{hc}$, Partition-oriented cluster topology $G_{pc}$.
1: Take each VM in $G$ as a host-oriented cluster to initialize $G_{hc}$, haveAddCluster = True;
2: while haveAddCluster do
3:   for $hc$ in $G_{hc}$ do
4:     Calculate the maximum $\Delta Q$ of each edge in $E$ belonging to $hc$;
5:   end for
6:   if the maximum $\Delta Q > 0$ then
7:     Merge corresponding two clusters with the maximum $\Delta Q$;
       Update information about related edge in $G_{hc}$;
8:   else
9:     haveAddCluster = False;
10:  end if
11:  Under resource constraints, merge indirectly connected clusters $hc$ in $G_{hc}$;
     Merge the remaining isolated clusters $hc$ in $G_{hc}$ considering server side constriants;
12:  Take each host-oriented cluster in $G_{hc}$ as a partition-oriented cluster to initialize $G_{pc}$, haveAddCluster = True;
13: end while
14: while haveAddCluster do
15:   for $pc$ in $G_{pc}$ do
16:     Recalculate the maximum $\Delta Q$ of each edge in $E$ belongs to $pc$;
17:   end for
18:   if the maximum $\Delta Q > 0$ then
19:     Merge corresponding two clusters with the maximum $\Delta Q$;
        Update information about related edge in $G_{pc}$;
20:   else
21:     haveAddCluster = False;
22:   end if
23: end while
24: return $G_{hc}$,$G_{pc}$
```

can happen here, if the host can support it, the cluster is allocated into the host, removing the allocated resources on the host. The second action happens if the cluster can not fit into the host, the algorithm select another host with enough capacity. Once all the host-oriented clusters are tested, the placement is over.

In this section, when we discuss the placement algorithm, our main consideration is the migration cost instead of communication cost. The reason can be explained as follows. In accordance with our 2-tier cluster-aware clustering algorithm, we have already clustered VMs with close traffic patterns into a host-oriented cluster, also clustered host-oriented clusters with frequent communication into a partition-oriented cluster. This way, compared with the overall internal traffic of media servers in DCN, traffic among partition-oriented clusters is such a small part that we can ignore the impact from this part. Studies [13, 14] gave us the similar conclusion, where a newly generated cluster of VMs is placed in an arbitrary available partition. Although the above traffic-aware VM placement researches ignore the actual migration progress, they do not consider the effect of VM placement on migration cost. Indeed, an appropriate VM placement strategy can optimize the generation of internal traffic in DCN from dynamic migration of VMs. Since our model is migration oriented, our placement algorithm focus on the migration cost when we choose an appropriate position for each VM to be migrated.

First of all, we define two types of migration cost: the migration cost of partition-oriented cluster to partition (C_p) and the migration cost of host-oriented cluster to host (C_h).

1) Definition of C_p

VM migration needs one VM to continuously copy its entire image and dirty pages, also needs the VM continue to copy the remaining dirty pages until reaching an iterative threshold. In the whole migration process of VM v_i, the transmitted migration data $m(v_i)$ can be calculated as below:

$$m(v_i) = v_i(c) \sum_{k=1}^{n} \left(v_i(r)/L\right)^{k-1} \tag{4}$$

Here, n is the default number of iterations. $v_i(r)$ is the dirty page production rate of VM v_i, L represents the available link bandwidth, which equals to the difference between the links bandwidth capacity and the corresponding bandwidth consumption among VMs communication. In our paper, the links bandwidth capacity is defaulted as constant, however, VMs communication leads to dynamic bandwidth consumption. Thus, L is a variable. Then, we use $C_p(pc_i, p_j)$ to represent the migration cost from pc_i to an appointed partition p_j, its value equals to the sum amount of transmission data for migrating all VMs in pc_i to p_j, $C_p(pc_i, p_j)$ can be expressed in the following:

$$C_p(pc_i, p_j) = \sum_{v_k \in pc_i} m(v_k) \times H_{mig}(v_k, p_j) \tag{5}$$

2) Definition of C_h

The partition-oriented cluster placement step has decided the cluster placement, while considering the cost of migrating a partition-oriented cluster to another partition. This step is a many-to-one mapping, which can guarantee that all partition-oriented clusters have been mapped to specific partitions with minimum migration cost, depending on the port number of the connected switch, host-oriented clusters belong to a partition-oriented cluster can only choose servers within the specified partition, so we just need to find the mapping relationship between each VM in a host-oriented cluster and each host in the corresponding partition. To be specific, if one VM needs to be migrated to a specified partition, no matter which server in that partition it chooses, the migration cost is the same. Here, for a hc_i in one corresponding pc, $C_h(hc_i, h_j)$ represents the migration cost from hc_i to destination host h_j, where:

$$C_h(hc_i, h_j) = \sum_{v_k \in hc_i \wedge v_k \notin h_j} m(v_k) \tag{6}$$

In this paper, we define two assigning-matrices: M_{pc} for assigning partition-oriented clusters to partitions, and

M_{hc} for assigning host-oriented clusters to hosts. In both matrices, each element belong to the matrix can take the value 1 or 0 to indicate whether to execute the assignment or not, respectively. In academia, the assignment problem is widely treated as the maximum weighted bipartite matching problem. The classical solution to the assignment problem is given by the Hungarian algorithm or Kuhn-Munkres algorithm, originally proposed by Kuhn[36] and refined by Munkres[37]. The Hungarian algorithm solves the assignment problem in O(n^3) time, where n is the size of one partition of the bipartite graph. The proposed algorithm firstly calculates $C_p(pc_i, p_j)$ for each pc_i in G_{pc}. Then the algorithm continues to obtain M_{pc} for partition-oriented clusters with the minimum migration cost by using Hungarian Algorithm. Secondly, the algorithm calculates $C_h(hc_i, h_j)$ for each hc_i in G_{hc}, and then obtains M_{hc} for host-oriented clusters with minimum migration cost by using Hungarian Algorithm. Note that in the above process, the algorithm ensures each partition-oriented cluster corresponds to a partition, and each host-oriented cluster corresponds to a host.

Above all, the migration cost C_{mig} can be expressed as following:

$$C_{mig} = \sum_{i=1}^{m}\sum_{j=1}^{n} C_p(pc_i, p_j) * M_{pc}(i,j) + \sum_{i=1}^{p}\sum_{j=1}^{q} C_h(hc_i, h_j) * M_{hc}(i,j) \tag{7}$$

Here, m is the number of partition-oriented clusters, n is the number of partitions, p is the number of host-oriented clusters, and q is the number of hosts under one particular partition. From formula (7), our algorithm solves M_{pc} and M_{hc}, which places VMs with frequent communication as close as possible, to optimize C_{mig}.

The whole process is also described in the pseudocode of Algorithm 2.

Algorithm 2 2-tier cluster-aware placement algorithm

```
INPUT: Partition-oriented cluster topology $G_{pc}$, Host-oriented cluster topology $G_{hc}$.
Number of partition-oriented clusters $m$. Number of partitions $n$. Number of host-oriented clusters $p$. Number of hosts under one partition $q$.
OUTPUT: Assigned-matrix for partition-oriented clusters $M_{pc}$, Assigned-matrix for host-oriented clusters $M_{hc}$.
1: Calculate $(\forall i, \forall j) C_p(pc_i, p_j)$ to solve $C_p$ matrix;
2: Use Hungarian Algorithm to obtain $M_{pc}(i,j)$ from $C_p$,
   which has the minimum $\sum_{i=1}^{m}\sum_{j=1}^{n} C_p(pc_i, p_j) * M_{pc}(i,j)$
   subject to: $\sum_{i=1}^{m} M_{pc}(i,j) = 1$, and $\sum_{j=1}^{n} M_{pc}(i,j) = 1$
3: Calculate $(\forall i, \forall j) C_h(hc_i, h_j)$ to solve $C_h$ matrix;
4: Use Hungarian Algorithm to obtain $M_{hc}(i,j)$ from $C_h$,
   which has the minimum $\sum_{i=1}^{p}\sum_{j=1}^{q} C_h(hc_i, h_j) * M_{hc}(i,j)$
   subject to: $\sum_{i=1}^{p} M_{hc}(i,j) = 1$, and $\sum_{j=1}^{q} M_{hc}(i,j) = 1$
5: return $M_{pc}$, $M_{hc}$
```

4.3 Collaborative VM migration algorithm

The output of the 2-tier cluster-aware placement algorithm is the VMs migration location in the data center. Using the previous location and the suggested location of the VMs, the collaborative VM migration algorithm achieves migration with the maximum concurrent degree, while considering the available link bandwidth at each migration step and avoiding potential link competitions in DCN. The algorithm calculates $g(v_i)$ as the difference of C_{com} from the current position and the destination of VM v_i. The migration process is periodic based on the total migration gain $g(v_i)$ of all VMs in DCN. Notice that we set a threshold (α) for the migration gain. The migration algorithm executes periodically when the total migration gain is over the threshold. In detail, the algorithm optimizes the total migration time effectively by scheduling concurrent sequences of VM migrations. Meanwhile, the algorithm optimizes the total migration time effectively by scheduling concurrent sequences of VM migrations. On the other hand, the algorithm increases the available link bandwidth L for subsequent VM migrations to reduce the total migration data by prioritizing VM migrations with a larger network gain in DCN, and further optimizes C_{mig} in DCN.

In our algorithm, we tend to find the maximum concurrence degree of VM migration while avoiding network link congestion. The reason is that for VMs, the time of parallel migration with link sharing is more than the time of serial migration. Ye et al. [31] gave us the specific conclusion by comparing serial migration and parallel migration with various concurrent granularities (from 1 to 16). Moreover, parallel migration induces more link competitions, which may lead to the degradation of the available link bandwidth L for subsequent VM migrations. From formula (4), obviously, due to the intensive competition of network resources, link competitions among parallel VM migrations bring more transmitted migration data from the dirty page generations. Therefore, we should prefer migrating VMs with larger network gain in DCN. So in this paper, we denote $g(v_i)$ as the network gain from migrating VM v_i in DCN. The algorithm calculates $g(v_i)$ as the difference of C_{com} from the current position and the destination of VM v_i. Considering the DCN architecture (from the access layer to the root layer), traffic packets between VMs are usually traveling through different paths, or at least there are backup alternative physical paths. In this paper, by default, the algorithm takes the path with the lightest load as the communication path between physical machines. This rule also applies to the load balancing strategy for switches in real data centers. We assume the VM migration will run out of the bandwidth resources as much as possible on the migration path. Consequently, the algorithm takes the minimum bandwidth of the path as migration bandwidth usage, then the migration bandwidth is subtracted from the available bandwidth of all links on the path.

From the 2-tier cluster-aware placement algorithm, we have obtained the migration destination for each VM. For these VMs, our algorithm sets two queues for VM migration: migrating queue (MQ), and waiting queue (WQ). The total capacity of two queues equals to the number of VMs which need to be migrated, and the size of both MQ and WQ

depends on the resource requirement of each VM, whether the corresponding host can satisfy the requirement. In detail, VMs in MQ indicates that those VMs can be migrated immediately, and VMs in WQ means that those VMs cannot be migrated because of the server-side constraints. The algorithm takes the perceived whole cluster as the migration target to enqueue VMs based on the capacity of destination-host. VMs in MQ are in a descending order according to $g(v_i)$. The algorithm selects the candidate destination for VM with maximum $g(v_i)$, which satisfies all server constraints. Then the algorithm calculates the migration path for VM, tending to avoid link competitions. If the migration path satisfies the available bandwidth requirement, then start migrating and update the available bandwidth of associated links in DCN. When one VM migration is completed, the algorithm releases physical resources of its original host and available bandwidth of associated links. The migration process is repeated until each VM has been migrated into one appointed physical host. There may be a situation: MQ is null but WQ is not null. The algorithm detects it as the occurrence of deadlock, and we call it Deadlock Waiting for Resource. For example, we assume VM_1 needs to migrate from PM_1 to PM_2, and VM_2 migrates from PM_2 to PM_1. However, expected migrations are not allowed due to the resource limitation for both hosts, means both PM_1 and PM_2 do not have enough resources to accommodate VMs to migrate in. To break the deadlock, the algorithm chooses one close PM with enough capacity, which also has the least migration cost for VM_1 (or VM_2, depending on migration cost). Naturally, the algorithm firstly migrates VM_1 to the selected close PM, then migrates VM_2 to PM_1, and finally migrates VM_1 to PM_2.

The pseudocode for the algorithm is described in Algorithm 3.

Since the data center reconfiguration is a time sensitive issue and the problem of VM placement is NP-hard, the computational complexity of algorithms cannot be ignored. Some of the recent works[20, 28] focused on traffic-aware VM placement, however the computational complexity of the VM clustering algorithm in Mengs work is O(n4)[28], and the VM grouping algorithm in[20] is also O(n4), where n represents the number of VMs, which may not be scalable under highly variable network traffic. Thus, we need to analyze of the complexity of the proposed algorithms. Firstly, we consider the placement process. For n VMs, sorting the traffic gain (ΔQ) of merging host-oriented clusters takes O($n^2 \log n$) time. After sorting ΔQ in a descending order, the pair of VMs corresponding to the maximum ΔQ is tried to be clustered together. The time of finding maximum ΔQ is O(n). The while loop of adding new elements ends at around $n^2/2$ steps. Therefore, the total time complexity of the while loop is O(n^3). Then the same process for merging partition-oriented clusters, which will also take O(n^3) time. Next, the cluster placement algorithm calculates the cost of migrations for each cluster, which takes at the most O($n^2 \log n$) time. At last, we use the Hungarian algorithm to place the two type clusters into positions, this will cost O(n^3) time. Thus the time complexity of placement process is O(n^3). Secondly, we consider the migration process. For n VMs to be migrated, sorting the network gain for all VMs takes O($n^2 \log n$) time. The time of finding maximum network gain is O(n). The while loop of migrating VMs ends at most n steps, so the total time complexity of the while loop is O(n^2). Thus the time complexity of migration process is O($n^2 \log n$). In summary, the overall computational complexity of the proposed algorithms is O(n^3).

Algorithm 3 Collaborative VM migration algorithm

```
INPUT: Assigned-matrix for partition-oriented clusters $M_{pc}$,
Assigned-matrix for host-oriented clusters $M_{hc}$.
Bandwidth of links in DCN.
OUTPUT: Execution of VM migrations.
1: Push VMs into $MQ$ or $WQ$ according to resource constraints on appointed target host, where $MQ$ is in accordance with the descending order of $g(v_i)$;
2: for $v_i$ in $MQ$ do
3:   if migration is over then
4:     Release associated physical resources and link bandwidth of VM $v_i$;
5:   end if
6: end for
7: while $MQ \neq$ NULL do
8:   Pop VM $v_i$ from $MQ$, then calculate a migration path for $v_i$ and obtain available link bandwidth $L$ on the migration path;
9:   if $L$ is satisfied then
10:    Start migration of VM $v_i$ and update available physical resources and link bandwidth;
11:    Goto Line1;
12:  else
13:    Goto Line8;
14:  end if
15: end while
16: if $MQ$=NULL $\wedge$ $WQ \neq$ NULL then
17:   Migrate the selected VM to the closest host with enough capacity to break the deadlock, which has the least migration cost for the VM. Then update available physical resources and link bandwidth associated to the VMs;
18:   Goto Line1;
19: end if
```

5 Evaluation

In this paper, we build a simulation platform of media cloud based on CloudSim [38]. Cloudsim is the most popular simulator tool available for cloud computing [39]. After modifying the networking module and dirty page calculating module of CloudSim, the simulation platform can fully satisfy our simulation setup. In detail, on the basis of extending the networking module of CloudSim, we have added the VL2 and BCube architecture. In the same time, we have introduced new function into CloudSim. The function takes the dirty page rate and simultaneous memory writes into account, and calculates the migration time with iterative phases, by dividing the VM memory size by the available network bandwidth. The results prove that our proposed model can effectively solve the problem of VM migration in media cloud, while optimizing the internal traffic of DCN among media servers through simulation.

TABLE 1: Default parameters of simulation experiment

Type	Parameter	Value
VM	Number	128
	CPU	1799Mips*2 and 4799Mips*4
	Memory	1GB and 2GB
	Bandwidth	100Mbps and 200Mbps
	Dirty page production rate	10Mbps
Server	Number	64
	CPU	12000Mips*16
	Memory	30G
	Bandwidth	2000Mbps
DCN	Architecture	Tree/VL2/BCube
	Port numbers of aggregate switch	4
	Port numbers of access switch	4
	Link capacity between switches	1000Mbps

In detail, the default simulation settings are shown as Table 1. In the following simulation, besides the specific parameter settings stressed in each simulation, all of the other unmentioned parameters utilize the default values. Notice that the physical servers are connected to the access switches and the number of the switches is stationary if the DCN structure, the number of the physical servers, and the number of ports for switches are decided. For example, if k is the port number of each switch in VL2, there will be $k/2$ access switches and $k/2$ aggregation switches. Thus there are $5k2/4$ switches in total. According to the studies [40, 41, 42], the distribution of the arrival traffic in a node within the data center follows a log-normal pattern. Thus, in our experiment, each element of the traffic matrix is generated using a log-normal distribution, with mean (μ) and standard deviation (σ) as variables, which are shown in bellow:

$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\frac{-(\ln x-\mu)^2}{2\sigma^2} \tag{8}$$

In our simulation, the mean μ is set to 0, we want to set $log(x) \sim N(0,\sigma^2)$ without deviation, which is closer to the real cloud environment. The standard deviation σ is set to 4. With σ equals to 4, we find that all VMs are linked with a high traffic exchange and the traffic pattern is enough scattered. Meanwhile, with the increase of concurrent users in the burst cloud, on the one hand, the weight of communication traffic between VM pair increases. This change only increases the total amount traffic in DCN, however, it doesnt affect the proportion of flow distribution according to the traffic load model. Thus this change also applies to the proposed scheme. On the other hand, the communication traffic among VMs is more frequent. In this case, based on the work of Kandula [43], we introduce the sparse topology and the dense topology. Kandula has found that a server within a data center either talks to almost all the other servers within the rack or it talks to fewer than 25% of servers within the rack. Further, a server either doesn't talk to servers outside its rack or it talks to about 1%-10% of outside servers. Therefore, based on the work of Kandula, we introduce the sparse topology and the dense topology. To be specific, in the sparse topology, we define that each VM talks to 10% of servers within the rack, and 1.5% of outside servers. In the dense topology, we assume that each VM talks to 25% of servers within the rack, and 3.5% of outside servers. Note that all edges are randomly generated to simulate the uncertain media cloud.

In our evaluation, we set up 2 types of experimental scenarios. The first experimental scenario aims to prove the effectiveness of our 2-Tier Cluster-Aware Virtual Machine Clustering and Placement (TC-VMP) algorithms. Under this set of scenario, we compare TC-VMP with the Traffic-Aware Virtual Machine Placement (TA-VMP) algorithm [13], which is the most related to our proposed placement algorithm. We also provides the contrast of TC-VMP and the Best Fit with hierarchical clustering (BF-HC) algorithm [29], which is also one typical traffic-aware algorithm. Furthermore, the results of the Load Balancing Virtual Machine Placement (LB-VMP)algorithm are also provided. LB-VMP [14] is a classical placement algorithm, which aims to select the physical server with the smallest resource usage rate in each turn to achieve the load balance of resources on server sides. Finally, in order to make a comparison under the same condition, we agree that the above VM placement algorithms ensure all the physical servers are not overloaded. In the second experimental scenario, aiming to prove the superiority of Collaborative Virtual Machine Migration (C-VMM) algorithm, we compare C-VMM with other two algorithms: Random Virtual Machine Migration (R-VMM) algorithm and Concurrent Virtual Cluster Migration (C-VCM) algorithm [31]. R-VMM is a baseline algorithm that performs parallel migrations at each time step while greedily decide the order of migrations. In detail, R-VMM starts by identifying the VMs that can be directly migrated if the destination hosts have enough resources to accommodate these VMs, and there is enough available link bandwidth to accomplish the migration. This process is repeated until all VMs are migrated. C-VCM investigates the migration performance of multiple virtual clusters with various concurrent granularities, which is most related to C-VMM. Finally, to reduce the randomness of the simulation and experimental results, our evaluation results presented in this paper are the average results with the corresponding 95% confidence intervals over twenty runs.

5.1 The Effectiveness of TC-VMP

In this section, we will present the optimization results of the internal traffic in DCN by using LB-VMP, TA-VMP, and TC-VMP. The optimization focuses on the average number of switches passed by the communicating data between VMs in DCN, which is called as the *average packet exchange frequency* in this section. First of all, under three different DCN architectures, we compare the average packet exchange frequency with the above three algorithms. Next, we particularly take the Tree architecture as an example to carry out a thorough analysis.

In our experiment, under Tree, VL2 and BCube [31], we firstly compare the average packet exchange frequency with the algorithms. The results are shown in Figure 2 and Figure 3.

From Figure 2 and Figure 3, in Tree, VL2 and BCube, we notice that TA-VMP and TC-VMP have dramatic optimization results. The reason is that above two algorithms

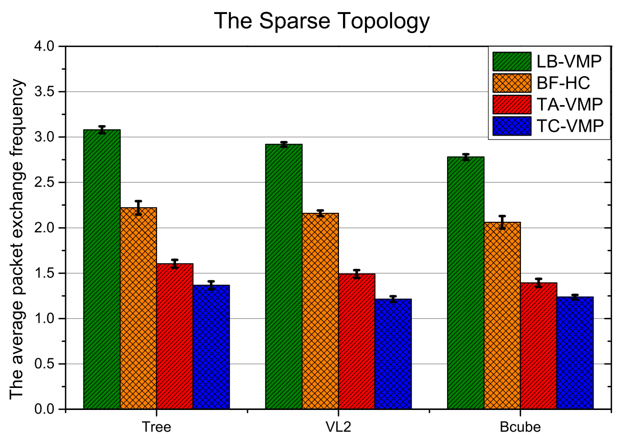

Fig. 2: The average packet exchange frequency under the sparse topology in DCN.

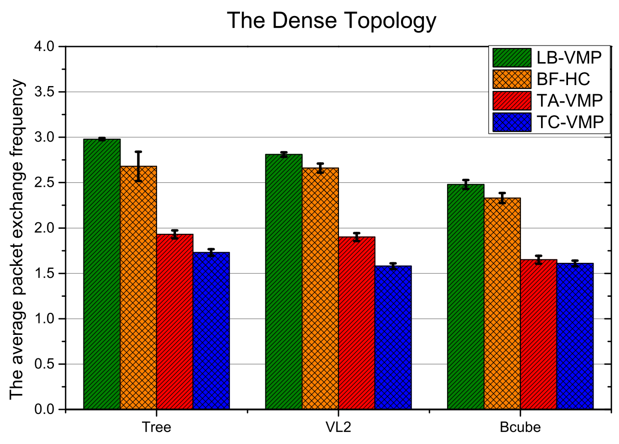

Fig. 3: The average packet exchange frequency under the dense topology in DCN.

cluster VMs based on traffic patterns and then place clusters into partitions. Hence, most of the internal traffics are optimized into partitions, even physical servers. Although BF-HC achieved an optimized result, the efficiency is much less than TA-VMP and TC-VMP. Considering VM neighbors have larger traffic between each other in a sequence, BF-HC obtained VM sequence by using the minimum cuts. Thus, a VM should be placed on the first host with enough resource requirements. Therefore, the adjacent VMs could be cut into different host or switch because of different the resource constraints.It is worth mentioning that TC-VMP performs much better than TA-VMP, because TC-VMP places more VMs with frequent communication into one physical server instead of one partition. Traffic within one server doesn't pass by any switch and traffic within one partition only passes by one access switch in DCN. Moreover, we can see that both in sparse and dense topology, the simulation result of BCube performs better than that of Tree and VL2. The main reason is that BCube architecture itself helps data packets bypass the switches on core layer or aggregate layer in DCN. Meanwhile, under the architecture of BCube, the

TABLE 2: The internal traffic distribution in DCN under the sparse topology

Algorithm	Average rate of traffic optimized within servers	Average rate of traffic optimized within partitions (not in servers)
LB-VMP	5.36%	28.22%
BF-HC	17.36%	50.60%
TA-VMP	51.89%	13.33%
TC-VMP	58.33%	12.61%

result of TC-VMP has the smallest change of the average packet exchange frequency compared with the results under Tree and VL2. This is because TC-VMP has gathered the most part of internal traffic within partitions, and only a little part of the traffic will pass by the core or aggregate switches, which also reflects the effectiveness of TC-VMP.

Aiming to understand the mechanism of internal traffic optimization in DCN, we further compare the traffic distribution results in DCN: within one host, within one partition, and outside one partition. In the following experiments, we take Tree architecture as an example to analyze more details. Experimental parameters are the same as that in the above experiment. The results are shown in Figure 4.

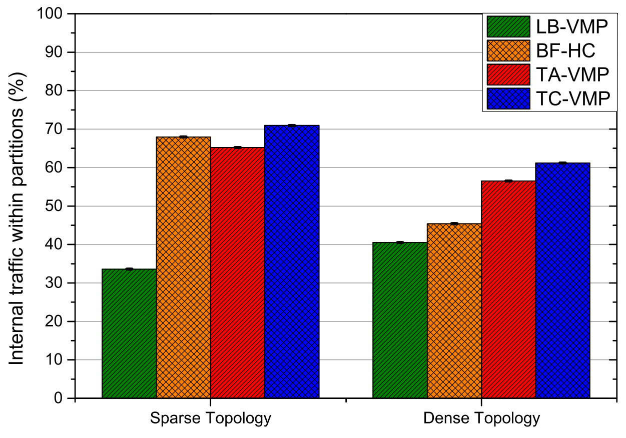

Fig. 4: The internal traffic rate within partitions in DCN.

From Figure 4, we obviously know that LB-VMP can only optimize around 35%-40% internal traffic within partitions, which means most parts of traffic will pass by high layer switches in DCN frequently. Hence, LB-VMP is inappropriate for media streaming applications with such tight communication. We focus on three features about BF-HC. First, the optimization effect is as good as TC-VMP for the sparse topology, and even find the situation was superior to TC-VMP. Second, the optimization effect drops greatly within the dense topology. Third, the randomness from replacement has contributed to an unstable result.Differently, TA-VMP and TC-VMP performs better on the distribution of internal traffic in DCN. Moreover, TC-VMP has a higher performance, about 5% for optimizing the internal traffic in DCN than TA-VMP does. Besides, TA-VMP even ignores the resource requirements of VMs, which indicates it is unable to make full use of server-side resources. Thus TA-VMP is unacceptable for a cloud provider to deploy large-scale media services. Furthermore, Table II shows us the specific distribution of the internal traffic in a DCN partition under the sparse topology. The results show that whether the internal traffic is optimized within a server or not. In Table II, compared with other two algorithms, TC-VMP optimizes the most parts of the internal traffic into partitions, also physical servers. This conclusion proves the superiority of traffic distribution with TC-VMP as well.

In addition, in order to verify the effectiveness of our algorithm under different structures of DCN, by changing the number of VMs (ranges from 64 to 256), the number of physical servers (ranges from 32 to 128) and the number of ports for access switch (ranges from 2 to 16), we compare and analyze the change of average packet exchange frequency under the dense topology in Tree architecture for all three algorithms. The other experimental parameters refer to the default parameters shown in Table I. The results are shown in Figure 5, Figure6, and Figure 7.

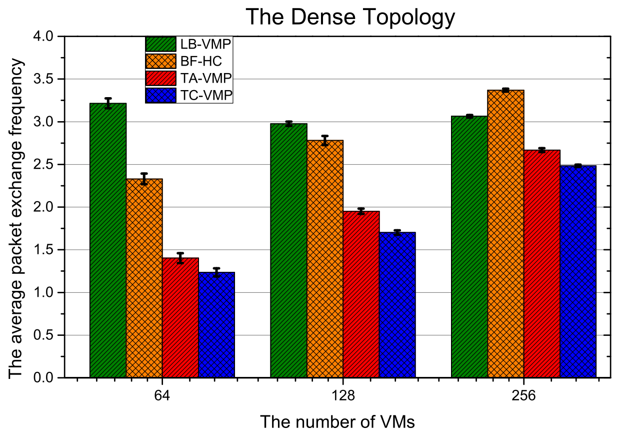

Fig. 5: The average packet exchange frequency under the dense topology in DCN by changing the number of virtual machines.

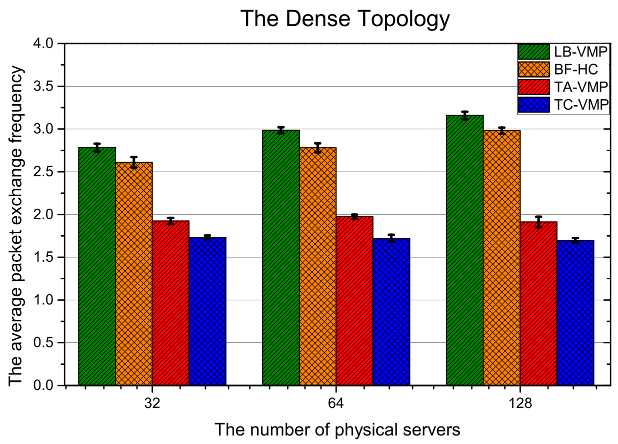

Fig. 6: The average packet exchange frequency under the dense topology in DCN by changing the number of physical servers.

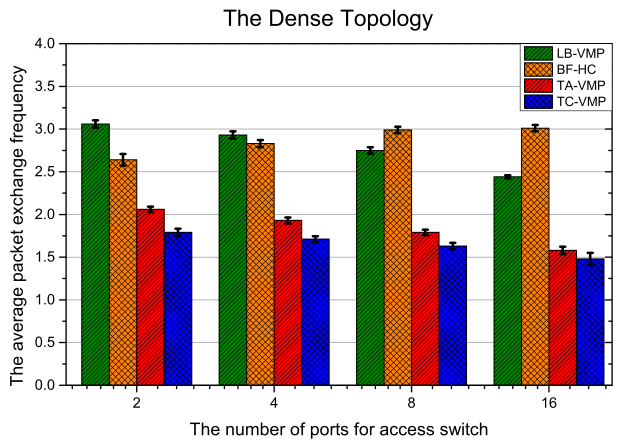

Fig. 7: The average packet exchange frequency under the dense topology in DCN by changing the number of ports for access switch.

From Figure 5, we notice that changing the number of VMs has little effect on optimization results for LB-VMP. The reason is that the increased VMs only increase the total amount of communication between VMs, but it barely affects the percentage of internal traffic distribution in DCN. In addition, the optimization reduces rapidly as the number of VMs increases, which is even worse than the effectiveness of LB-VMP when there are 256 VMs. This proves the inadequacy for BF-HC under dense topology once again. However, as for TA-VMP and TC-VMP, the results increase continuously by adding new VMs in DCN. The reason is that some hosts cannot accommodate more VMs due to the server-side constraints, thus that part of VMs with frequent communication are placed outside the same host, even the same partition. Meanwhile, we find that TC-VMP performs better than TA-VMP during the whole period.

From Figure 6, along with the increase of physical servers, LB-VMP distributes VMs on all of the physical servers in DCN considering the features of load balancing. This certainly increases the average packet exchange frequency from communication between VMs. Differently, as long as the number of physical servers could satisfy the resource requirements of VMs, the increased physical servers will not affect the average packet exchange frequency with BF-HC, TA-VMP and TC-VMP. Since the number of physical servers within one partition does not change, all the traffic-aware algorithms have merged VMs with frequent communication into clusters, and then placed clusters into corresponding partitions, thus the added physical servers will be placed under new partitions, which will not affect the optimization results.

From Figure 7, under the fixed number of VMs, the increase of switches leads to a more dispersed distribution of VMs before optimization, which finally increases the average packet exchange frequency. Both TC-VMP and TA-VMP use two-layer clustering mechanism, which try to place VMs as close as possible such as in one partition. With the increase number of switches, the number of VMs in one partition increases, which shows us a better result. However, BF-HC places VMs into a dispersed distribution, which is too scattering to hold the trend of optimization. Meanwhile, the results of LB-VMP change most intensely compared with TA-VMP and TC-VMP. Because more than 60% of the internal traffic in LB-VMP is outside partitions originally, now most parts of the traffic are optimized within partitions, even within servers. While the results of TC-VMP have the least significant change, because TC-VMP has already optimized around 60%-70% of the internal traffic into partitions, which further proves the effectiveness of our algorithm.

5.2 The effectiveness of C-VMM

Our scheme is migration oriented and the ultimate goal is to minimize the internal traffic in media cloud. Under this context, we use two performance metrics, total migration time and total data transmission, as the representatives to evaluate the quality of VM migration. We use simulations to answer how well our proposed migration mechanism performs in DCN.

In the experiment, by changing the initial number of VMs (ranges from 32 to 256), we compare the total migration time and total data transmission in Tree architecture for different migration strategies of VMs, respectively. Other experimental parameters refer to the default parameters. The results are shown in Figure 8 and Figure 9.

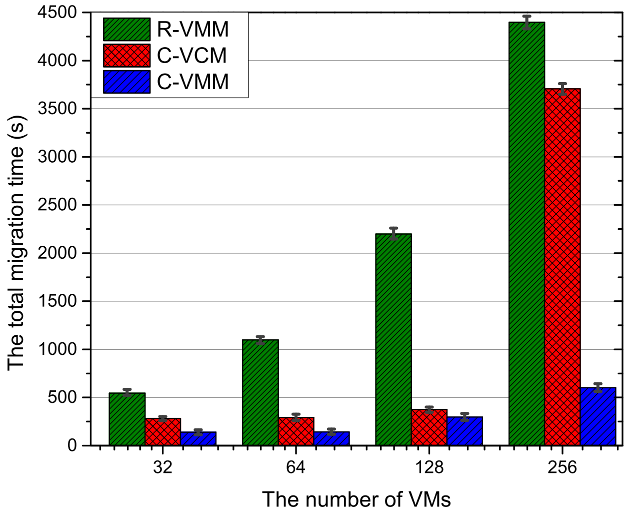

Fig. 8: The total migration time in DCN by changing the number of VMs.

From Figure 8, we find that with the increasing number of VMs, the total migration time of R-VMM is linear increase, which indicates the migration time of one single VM is basically the same. When the number of VMs ranges from 32 to 128, C-VCM performs much better than R-VMM. The reason is that the destinations for most clusters migration are different partitions, thus C-VCM avoids link collisions caused by concurrent migrations to a certain extent. While the total migration time of C-VCM suddenly increases when there are 256 VMs, because the concurrency degree of clusters migration is too high to avoid link congestion. Hence, C-VCM no longer performs significantly better than R-VMM when there are massive numbers of VMs to be migrated. Differently, C-VMM performs much better than the other two algorithms during the whole period. To be specific, when the number of VMs ranges from 32 to 64, C-VMM has almost the same migration time because the algorithm increases the concurrency degree of VM migrations. When the number of VMs ranges from 64 to 256, we see the total migration time exist a rough linear growth. It means that C-VMM estimates existing network bandwidth of DCN does not satisfy the increase of concurrent degree, thus avoids the concurrent migrations with link competitions.

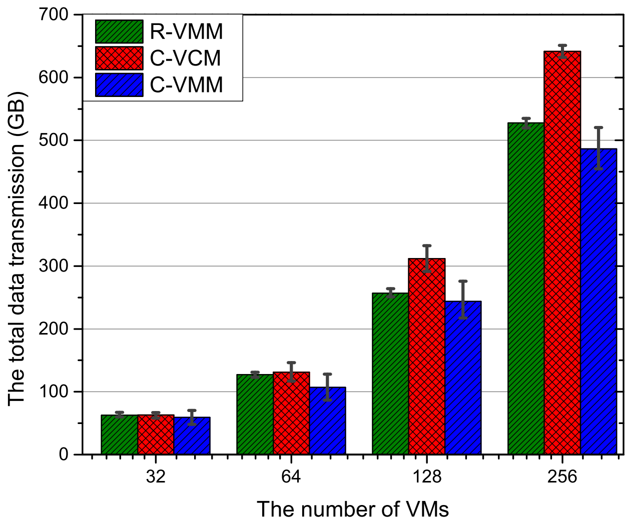

Fig. 9: The total data transmission in DCN by changing the number of VMs.

In Figure 9, similarly, we know that the total data transmission of R-VMM is basically proportional to the number of VMs. While C-VCM performs much worse than R-VMM, because the excessive concurrency degree of clusters migration leads to link congestion, which will extend the migration time and increase the total amount of migration data. In addition, C-VMM performs even better than R-VMM. There are two reasons. First, C-VMM effectively avoids potential concurrent migration with link competitions. Second, C-VMM prefers to migrate VMs which are conducive to the network condition in DCN, and hence the available bandwidth increases for subsequent VM migrations. Therefore, C-VMM increases the available bandwidth for VM migration in two ways, which can reduce the total migration data in media cloud.

6 CONCLUSION

Cloud-based media streaming applications in media cloud exhibit high resource consumption on server sides and dynamic traffic flow among media servers. Such characteristics need to be fully considered during the VM placement and migration processes. In this paper, we propose a cluster-aware VM collaborative migration scheme for media cloud, which highly accommodates the characteristics of media streaming under cloud environment. In detail, the proposed scheme simultaneously considers the infrastructure of the DCN and the application resource requirements of media servers. It introduces a generic scheme of VM migration to optimize the internal traffic in media cloud, which tightly integrates clustering, placement, and dynamic migration process. Simulation results demonstrate that the scheme can effectively reduce the total internal traffic of DCN in media cloud.

In this study, the proposed scheme does not fully considers the application characters of media streaming to optimize the VM migration in media cloud. The future work should try to devise a collaborative VM migration method focusing on the user experience of media streaming application. Furthermore, due to the limitation of experimental environment, our scheme is just verified on the simulation platform with the self-generated traffic patterns, and the scheme ignores some emergencies such as the failing issue of VM migrations. Future work may switch to other packet-level simulator, such as SimGrid or GreenCloud, and what is more, we will evaluate the efficiency of the scheme in a real situation to obtain the experimental judgments by setting up real VM-hosted media servers.Furthermore, the current media cloud researches, including the proposed scheme, do not fully considers the application characters of media streaming to optimize the VM migration in media cloud. The future work should try to devise a quality of experience driven collaborative VM migration method focusing on the user experience of media streaming application.

Acknowledgment

This research was partially supported by The Fundamental Theory and Applications of Big Data with Knowledge Engineering under the National Key Research and Development Program of China with grant number 2016YFB1000903, the National Science Foundation of China under Grant Nos. 61472317, 61502379, 61532015, 61532004, the MOE Innovation Research Team No. IRT13035, and the Co-ordinator Innovation Project for the Key Lab of Shaanxi Province under Grant No.2013SZS05-Z01, and the Project of China Knowledge Centre for Engineering Science and Technology.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[3] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75–86, 2008.

[4] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. of IEEE INFOCOM*. IEEE, 2012, pp. 460–468.

[5] Y. Zhao, H. Jiang, K. Zhou, Z. Huang, and P. Huang, "Meeting service level agreement cost-effectively for video-on-demand applications in the cloud," in *Proc. of IEEE INFOCOM*. IEEE, 2014, pp. 298–306.

[6] F. Wang, J. Liu, and M. Chen, "Calms: Cloud-assisted live media streaming for globalized demands with time/region diversities," in *Proc. of IEEE INFOCOM*. IEEE, 2012, pp. 199–207.

[7] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, no. 17, pp. 2923–2938, 2009.

[8] V. De Maio, R. Prodan, S. Benedict, and G. Kecskemeti, "Modelling energy consumption of network transfers and virtual machine migration," *Future Generation Computer Systems*, vol. 56, pp. 388–406, 2016.

[9] J. Zhang, F. Ren, and C. Lin, "Delay guaranteed live migration of virtual machines," in *Proc. of IEEE INFOCOM*. IEEE, 2014, pp. 574–582.

[10] H. Xu and B. Li, "Egalitarian stable matching for vm migration in cloud computing," in *Proc. of IEEE INFOCOM WKSHPS*. IEEE, 2011, pp. 631–636.

[11] J. Liu, Y. Li, and D. Jin, "Sdn-based live vm migration across datacenters," in *Proceedings of the 2014 ACM conference on SIGCOMM.* ACM, 2014, pp. 583–584.

[12] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *Proc. of IEEE INFOCOM*. IEEE, 2011, pp. 66–70.

[13] T. Yapicioglu and S. Oktug, "A traffic-aware virtual machine placement method for cloud data centers," in *Proceedings of the IEEE/ACM 6th International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2013, pp. 299–301.

[14] D. S. Dias and L. H. M. Costa, "Online traffic-aware virtual machine placement in data center networks," in *Global Information Infrastructure and Networking Symposium*. IEEE, 2012, pp. 1–8.

[15] S. Zou, X. Wen, K. Chen, S. Huang, Y. Chen, Y. Liu, Y. Xia, and C. Hu, "Virtualknotter: Online virtual machine shuffling for congestion resolving in virtualized datacenter," *Computer networks*, vol. 67, pp. 141–153, 2014.

[16] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," in *Proc. of IEEE INFOCOM*. IEEE, 2012, pp. 2876–2880.

[17] Y. Guo, A. L. Stolyar, and A. Walid, "Shadow-routing based dynamic algorithms for virtual machine placement in a network cloud," in *Proc. of IEEE INFOCOM*. IEEE, 2013, pp. 620–628.

[18] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[19] H. Xu and B. Li, "Anchor: A versatile and efficient framework for resource management in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1066–1076, 2013.

[20] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1,

pp. 179–196, 2013.

[21] M. Alicherry and T. Lakshman, "Optimizing data access latencies in cloud systems by intelligent virtual machine placement," in *Proc. of IEEE INFOCOM*. IEEE, 2013, pp. 647–655.

[22] K. Zheng, X. Wang, L. Li, and X. Wang, "Joint power optimization of data center network and servers with correlation analysis," in *Proc. of IEEE INFOCOM*. IEEE, 2014, pp. 2598–2606.

[23] L. Chen and H. Shen, "Consolidating complementary vms with spatial/temporal-awareness in cloud datacenters," in *Proc. of IEEE INFOCOM*. IEEE, 2014, pp. 1033–1041.

[24] Z. Zhang, S. Su, J. Zhang, K. Shuang, and P. Xu, "Energy aware virtual network embedding with dynamic demands: online and offline," *Computer Networks*, vol. 93, pp. 448–459, 2015.

[25] Z. Zhang, S. Su, K. Shuang, W. Li, and M. A. Zia, "Energy aware virtual network migration," in *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[26] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2007, pp. 119–128.

[27] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.

[28] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. of IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[29] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, and S. Cheng, "Energy-saving virtual machine placement in cloud data centers," in *Ieee/acm International Symposium on Cluster, Cloud and Grid Computing*, 2013, pp. 618–624.

[30] H. Chen, H. Kang, G. Jiang, and Y. Zhang, "Network-aware coordination of virtual machine migrations in enterprise data centers and clouds," in *IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2013, pp. 888–891.

[31] K. Ye, X. Jiang, R. Ma, and F. Yan, "Vc-migration: Live migration of virtual clusters in the cloud," in *ACM/IEEE International Conference on Grid Computing*. IEEE, 2012, pp. 209–218.

[32] H. Liu and B. He, "Vmbuddies: coordinating live migration of multi-tier applications in cloud environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1192–1205, 2015.

[33] M. J. Magazine and M.-S. Chern, "A note on approximation schemes for multidimensional knapsack problems," *Mathematics of Operations Research*, vol. 9, no. 2, pp. 244–247, 1984.

[34] B. Wang, Z. Qi, R. Ma, H. Guan, and A. V. Vasilakos, "A survey on data center networking for cloud computing," *Computer Networks*, vol. 91, pp. 528–547, 2015.

[35] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[36] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics*, vol. 52, no. 1, p. 7C21, 2005.

[37] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[38] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[39] A. Ahmed and A. S. Sabyasachi, "Cloud computing simulators: A detailed survey and future direction," in *Advance Computing Conference (IACC), 2014 IEEE International*. IEEE, 2014, pp. 866–872.

[40] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *Acm Sigcomm Computer Communication Review*, vol. 40, no. 1, pp. 92–99, 2010.

[41] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," *Infocom Proceedings IEEE*, vol. 131, no. 5, pp. 2876 – 2880, 2012.

[42] D. Ersoz, M. S. Yousif, and C. R. Das, "Characterizing network traffic in a cluster-based, multi-tier data center," in *International Conference on Distributed Computing Systems*, 2007, pp. 59–59.

[43] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements and analysis," in *ACM SIGCOMM Conference on Internet Measurement 2009, Chicago, Illinois, Usa, November*, 2009, pp. 202–208.

Weizhan Zhang received the BS degree in electronics engineering in 1999 from Zhejiang University, China, and the PhD degree in computer science in 2010 from Xi'an Jiaotong University, China. He served as a software engineer in the Datang Telecom Corporation from 1999 to 2002. Now he is an associate professor in the department of computer science and technology at Xi-an Jiaotong University, China. His research interests include multimedia systems for e-learning, cloud computing, and mobile computing.

Yuxuan Chen received the BS degree in Computer Science and Technology in 2014 from Xian Jiaotong University, China. Now he is studying for a master's degree at Xian Jiaotong University, China. His research interest is multimedia systems for e-learning.

Xiang Gao received the BS degree in Computer Science and Technology in 2016 from Lanzhou University, China. Now she is studying for a master's degree at Xian Jiaotong University, China. Her research interest is multimedia systems for e-learning.

Zhichao Mo received the BS degree in computer science in 2012 , and the MS degree in computer organization and architecture in 2015 from Xian Jiaotong University, China. Now he is a software engineer in Aviation Industry Corportation of China. His research interests include multimedia systems for e-learning and cloud computing.

Qinghua Zheng received the BS degree in computer software in 1990, the MS degree in computer organization and architecture in 1993, and the PhD degree in system engineering in 1997 from Xian Jiaotong University, China. He is a professor in the Department of Computer Science and Technology at Xi'an Jiaotong University. He serves as the vice chancellor of Xi'an Jiaotong University. His research areas include multimedia distance education, computer network security, intelligent e-learning theory and algorithm. He did postdoctoral research at Harvard University from February 2002 to October 2002 and Visiting Professor Research in Hong Kong University from November 2004 to January 2005. He got the First Prize for National Teaching Achievement, State Education Ministry in 2005, and the First Prize for Scientific and Technological Development of Shanghai City and Shaanxi Province in 2004 and 2003 respectively.

Zongqing Lu received the B.E. and M.E. degrees from Southeast University, China, and the Ph.D. degree in computer engineering from Nanyang Technological University, Singapore, 2014. He is currently working as a postdoctoral scholar in the Department of Computer Science and Engineering at the Pennsylvania State University. His research interests include networked mobile systems, mobile and pervasive computing, and networking.