Ling *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:43
https://doi.org/10.1186/s13677-020-00193-8

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Network perception task migration in cloud-edge fusion computing

Chen Ling[1], Weizhe Zhang[1,2*], Hui He[1] and Yu-chu Tian[3]

Abstract

With the development of cloud computing, edge computing has been proposed to provide real-time and low-delay services to users. Current research usually integrates cloud computing and edge computing as cloud-edge fusion computing for more personalized services. However, both cloud computing and edge computing suffer from high network consumption, which remains a key problem yet to be solved in cloud-edge fusion computing environments. The cost of network consumption can be divided into two parts: migration costs and communication costs. To solve the high network consumption problem, some virtual machines can be migrated from overloaded physical machines to others with the help of virtualization technology. Current network perception migration strategies focus more on the communication cost by optimizing the communication topology. Considering both communication and migration costs, this paper addresses the high network consumption problem in terms of the communication correlations of virtual machines and the network traffic of the migration process. It proposes three heuristic virtual machine migration algorithms, LM, mCaM and mCaM2, to balance communication costs and migration costs. The performance of these algorithms is compared with those of existing virtual machine migration algorithms through experiments. The experimental results show that our virtual machine migration algorithms clearly optimize the communication cost and migration cost. These three algorithms have a lower network cost than AppAware, an existing algorithm, by 20% on average. This means that these three algorithms improve the network performance and reduce the network consumption in cloud-edge fusion computing environments. They also outperform existing algorithms in terms of operation time by 70% on average.

Keywords: Virtual machine migration, Network perception, Edge computing, Heuristic algorithm

Introduction

As a new type of pay-as-you-go computational model, cloud computing is widely studied and applied in academia and industry. Edge computing is an extension of cloud computing that provides lower delay and higher real-time services from the devices closer to users. Current research usually integrates cloud computing and edge computing as cloud-edge fusion computing to provide more personalized services than either of the two individual computational models. Cloud computing includes applications and datacenters that provide cloud operating environments. However, the continuous expansion of datacenters increases the cost. Statistics show that the energy consumption of a datacenter is the highest among its different maintenance costs [1]. High energy consumption equates to a continuous increase in maintenance fees. In 2010, the electricity cost of all datacenters was 11.5 billion dollars [2]. Meanwhile, as the network bandwidth increases, the datacenter's network performance, network traffic and network energy consumption become increasingly important [3, 4]. Network consumption encounters many problems, such as network congestion [5], which reduces the network performance and usability. One estimation indicates that the cost of a datacenter's network energy consumption is 10%–20%

*Correspondence: wzzhang@hit.edu.cn
[1]School of Computer Science and Technology, Harbin Institute of Technology, Xidazhi Street, Harbin, China
[2]Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China
Full list of author information is available at the end of the article

of the total cost [6]. Given that edge computing devices have much less computing and network resources than cloud computing, edge computing also suffers from a high network consumption problem. Therefore, in cloud-edge fusion computing environment, high network consumption is a key problem yet to be solved.

As the underlying technology of cloud computing, virtualization integrates users' applications as a virtual machine and allows users to manage and use this virtual machine conveniently [2]. It also makes the optimization of network consumption possible. Virtual machines can be migrated to a small number of physical machines and unused physical machines can be powered off to achieve a higher resource utilization ratio. This kind of migration strategy only focuses on the computing resources of the physical machines, such as the CPU, memory and disk storage. However, virtual machine migration is a resource-intensive procedure [7] and continuously demands not only computing, but also network resources. Most existing studies focus on the computing resources and limited studies have considered the influence of network resources on virtual machine migration [8]. Therefore, designing a virtual machine migration algorithm that can improve network performance and reduce network consumption would be significant in improving cloud-edge fusion computing performance.

V. Shrivastava et al. [3] proposed Application Aware (abbreviated as AppAware), as a network perception virtual machine migration strategy. It considers not only the computing resources, but also the topological structure of the underlying physical machines. AppAware migrates all virtual machines on a physical machine with a workload that exceeds its resource limits to other physical machines that have a better topological network communication structure. The migration of virtual machines faces several challenges in network performance. First, the migrated virtual machine needs a appropriate destination physical machine which has better communication correlations. Second, the virtual machine migration network traffic also reduces network performance, which need to be minimized. While AppAware optimizes the network performance, the migration also creates network traffic which in turn reduces network performance. Therefore, a virtual machine migration strategy should consider migration traffic and ensure the network performance of cloud-edge fusion computing in both virtual machine transmissions and normal working processes.

In cloud-edge fusion environment, edge computing devices have limited computing resources and network resources. Therefore, we consider that the virtual machine migrations create large volume of network traffic and limit the network performance of cloud-edge fusion computing. Considering that the less researches focus on the optimization of virtual machine migration network traffic, we propose a method which can balance the communication consumption and network traffic. Our contribution is summarized as follows:

- We take both communication costs and migration costs into the virtual machine migration problem to have a better optimization on the cloud-edge fusion computing environment. Then we give the mathematical definition of these two costs and a linear programming definition of the virtual machine migration problem.
- We propose three virtual machine migration algorithms, LM, mCaM and mCaM2, to solve the virtual machine migration problem. In the design of these algorithms, we use the weighted coefficients to balance the optimization of the communication cost and the migration cost.
- By simulation experiments, we demonstrate that our algorithms have a better optimization on datacenters. To compare the performance of these three algorithms, some experiments are conducted with various settings. In addition, we also compare the performance of the three algorithms with that of AppAware. Experimental results show that our dynamic algorithm reduces network and migration costs by 20% on average comparing with AppAware. And the proposed algorithms also outperform existing algorithms in terms of operation time by 70% on average.

For cloud-edge fusion computing, these algorithms significantly improve the network performance and reduce the power consumption of the edge network devices. It contributes to the development and further utilization of cloud-edge fusion computing. Moreover, the communication correlations between cloud center and edge devices is optimized, resulting in less network traffic and lower communication delay.

The paper is organized as follows: "Related work" describes related work on virtual machine migration cost and network perception migration methods. "Definition of the network-aware virtual machine migration problem" describes the model of the network perception virtual machine migration problem and improvements to this problem model. In "Optimization of the virtual machine dynamic migration algorithm", we introduce the design of the LM, mCaM and mCaM2. In "Experiment", we evaluate the performance of these three algorithms in terms of total cost, communication cost and migration cost. The performance results are compared with the existing algorithm AppAware. Finally, in "Conclusion" sections

we summarize our algorithms and draw up some conclusions.

Related work

In this section, we summarize research on virtual machine migration problem. This is divided into two parts: the migration cost problem and the network perception migration problem. The research on the migration cost problem clarifies what can influence the performance degradation brought about by migration. The research on network perception migration focuses on power consumption, datacenter network congestion and communication correlations between application layer virtual machines.

Research on the cost of virtual machine migration

A. Verma et al. [9] studied the cost of virtual machine migration. Their experiment revealed that migration is irrelevant to workload and it is only decided by the virtual machine itself. Specifically, the migration cost can be measured by the throughput and performance reduction brought about by migration. L. Lefèvre [10] mentioned that online migration from the source physical machine to the destination physical machine involves a physical memory page exchange. If the virtual machine is running, it will be paused during the duplication from the first to the last memory page. H. Liu et al. [11] considered the online migration cost in their examination of datacenter performance and power consumption management. Through an experiment on XEN, they revealed that the power consumption of the migration of a virtual machine has a positive correlation with network communication traffic.

Research on network perception virtual machine migration

Network perception virtual machine migration is the key point of this study. Research on this topic has shown limited progress. Most current studies on virtual machine migration and physical machine integration have focused on the conservation of system resources, such as CPU, memory, and disk storage. However, the network costs of current datacenters accounts for an important proportion of the total cost and is reflected in the following aspects. First, Wen et al. [5] found that the network cost of a datacenter accounts for 10%–20% of the total cost. Second, a datacenter has a large network communication system, which requires high network bandwidth usage. Thus, network congestion and other issues occur at several switches and switch ports [3]. If no measures are in place to handle these problems, the performance of the datacenters will reduce sharply, and the maintenance costs will increase rapidly. However, research on network perception virtual machine migration is still in the initial stages and has not received sufficient attention. Here, we review the research on network perception virtual machines from different angles.

Research on the angle of network power consumption

F. Feleke et al. [12] proposed a virtual machine placement algorithm based on both bin-packing heuristics and physical machines' power efficiently Medium-Fit to reduce SLA violation. They found that the resource utilization level threshold of overloaded physical machines influences the power efficiency. The algorithm was proposed to meet different demands of power efficiency. B. Heller et al. [13] found that network power consumption accounts for 10%–20% of the total datacenter power consumption. They indicated that the ElasticTree model can serve as a topological graph of datacenter physical machines. To minimize network consumption, they introduced an algorithm for calculating the optimal network subset. The algorithm abstracts the optimization problem as a subset and flow assignment problem. They also compared the differences among a formal algorithm, a greedy algorithm, and a topological perception algorithm in terms of solution quality and extensibility. M. Zhani et al. [14] approached the allocation problem from virtual machines to the virtual machine datacenter and regarded several resources as the fitness value of optimization. This algorithm aims to maximize benefits of virtual machine migration and minimize the energy consumption of datacenters. Sukhpal Singh et al. [15] proposed a heterogeneous self-optimization system for cloud energy efficiency. To optimize the energy consumption of cloud computing, an algorithm was proposed to estimate the energy consumption of each task and assign each task to the virtual machine with the lowest energy consumption. Sambit Kumar Mishra et al. [16] proposed a virtual machine selection algorithm to minimize the energy consumption based on dynamic voltage frequency scaling technology. The proposed algorithm estimates the energy consumption and execution time of each virtual machine and assigns each task to the virtual machine with minimal energy consumption and execution time. Next, for further optimization of the energy consumption, the algorithm finds the voltage frequency pairs that resulting in the minimal energy consumption for each virtual machine. Mohit Kumar et al. [17] proposed an architecture that acts as a cloud resource broker for dynamic task scheduling. To improve the optimization effect, an improved Particle Swarm Optimization (PSO) method was proposed to optimize not only the energy consumption, but also the QoS parameters. With the constraint of deadline, this method distinctly reduces the energy consumption based on the results of a simulation experiment.

Research on avoidance of datacenter network congestion

M. Bhatu et al. [18] proposed a task scheduling and resource allocation system to manage virtual machines and resources. The system uses an analytic hierarchy system to assign a rank to each task based on its length and runtime. Then they implemented a bandwidth-aware divisible scheduling method to allocate the computing resources and bandwidth, and a preemption methodology was proposed to check for overloaded virtual machines. Finally, the system uses divide-and-conquer methodology to break up tasks to solve the overloading problem. X. Wen et al. [5] investigated virtual machine migrations to alleviate the network congestion problem. They abstracted the problem into a secondary allocation problem and proposed Virtual Knotter, an online algorithm, to alleviate network congestion. The algorithm maintains low time complexity in the process of reducing the network communication cost. An experiment showed that the network cost of Virtual Knotter was only 5%–10% of that of the baseline algorithm. D. Kliazovich et al. [19] considered the reduction of network power consumption and the maintenance of network performance. They proposed the DENS algorithm to balance power consumption, the performance of each task, and the communication correlations to minimize the number of physical machines used. The algorithm maintains perfect communication correlations to avoid network congestion in the datacenter. In addition, Mohit Kumar et al. [20, 21] proposed an architecture to serve as the cloud resource broker for dynamic task scheduling and load balancing with elasticity. This architecture manages task assignment and resource provision for each virtual machine. Based on this architecture, [20] proposed a dynamic load balancing method to minimize the makespan time, which increases the QoS of cloud computing. [21] proposed an improved PSO method to optimize multiple QoS parameters. To solve the trade-off problem for task scheduling, the improved PSO method generates several schedule plans based on Pareto efficiency theory.

Research on communication correlations between application layer virtual machines

X. Meng et al. [22] investigated the problem of virtual machine migration to a physical machine on the basis of network perception. Their experiments on datacenters indicated that the distribution of virtual machines' traffic in a datacenter is extremely uneven. Virtual machines that communicate with each other have high traffic, and virtual machines with minimal communication have low traffic. They abstracted the network perception allocation of virtual machines into an NP-hard optimization problem and proposed a two-layer approximate algorithm to solve this problem. Their experiment showed that their algorithm can improve network performance and reduce the network communication costs. D. Jayasinghe et al. [23] focused on the virtual machine migration problem with the server's resource constraints. They proposed a virtual machine migration algorithm to improve performance and availability of IaaS cloud services. This algorithm aims to minimize the network traffic of datacenters by satisfying the resource demands and availability constraints of physical machines. The authors also analysed the NP-completeness of this problem and divided the problem into three NP-completeness optimization problems. First,They abstracted the minization of the communication cost into a clustering problem, which is essentially a K-Min-Max-cut problem. Second, they abstracted the allocation of virtual machine clusters to physical machine sets into a cluster allocation problem. Third, they abstracted the allocation of each machine to each physical machine into an allocation problem. They also proposed four heuristic algorithms to solve these problems. However, they didn't consider the limitations in the physical machine's system resources (CPU, memory, disk storage, etc.). The resources that the server provides must be sufficient for the virtual machine.

To solve this problem, V. Shrivastava et al. [3] considered the limitations in network communication and server resources and analysed a situation in which virtual machines are overloaded. They abstracted a datacenter into a model that considers the dependencies of the virtual machines and the topological relationships of the underlying physical machine. They proposed a greedy algorithm AppAware for selecting a server with the minimum communication cost as the destination physical machine for each virtual machine migration.

With regard to the research on overloaded virtual machines, T. Wood et al. [24] proposed Sandpiper to solve the problem of overloaded virtual machine migration. Sandpiper includes two algorithms. The first is used to estimate the virtual machine migration time, and the second is used to determine the migration method and the destination physical machine. Sandpiper uses greedy and heuristic methods to migrate the virtual machine from the physical machine with the highest overload ratio to that with the lowest one. Sandpiper is a dynamic virtual machine migration algorithm that selects one or several virtual machines from the virtual machine set on the basis of load variety. However, Sandpiper ignores the cost of virtual machine migration. H. Liu et al. [11] mentioned that the data transferred by virtual machine migration are positively related to the energy consumption. Thus, the data transferred by virtual machine migration cannot be ignored.

Definition of the network-aware virtual machine migration problem

In 2011, V. Shrivastava et al. [3] proposed Application Aware, a virtual machine migration strategy. AppAware gives the basic definition of virtual machine migration problem as follows.

For the virtual machine migration problem, we should consider three factors:

1. The communication correlations of VMs and the network topologies of physical machines. The former includes the settings of VMs and the communication relationships between them. The latter includes the settings of physical machines and network distance between them.
2. The resource demands of VMs and resource capacities of physical machines. The former means the computing resource demands of VMs, such as CPU, memory and disk storage. The latter means the quantity of the computing resources.
3. The sizes of VMs, which represent the amount of network traffic created by the VM migration.

For introducing the virtual machine migration problem briefly, we use some abbreviations in both mathematical functions and texts. The detailed abbreviations instructions are shown in Tables 1 and 2. In a datacenter, the virtual machine set is defined as $V = \{V_1, V_2, V_3, \ldots, V_n\}$, in which a subset of o number of virtual machines is defined as $O = \{V_1, V_2, V_3, \ldots, V_o\}$. The physical machine set is $S = \{S_1, S_2, S_3, \ldots, S_m\}$.The communication correlations between virtual machines can generate a network topological dependence graph, $G = (V, E)$, where E represents the set of the edges of the communication correlations between virtual machines. The definition of E is $E = \{(V_i, V_j) \mid V_i \textit{ and } V_j \textit{ have a communication correlation}\}$. $W(V_i, V_j)$ represents the communication demand between virtual machines V_i and V_j.

Table 1 Abbreviations

Abbreviations	Descriptions
VM	virtual machine
PM	physical machine
VMIG	Virtual Machine Migration
LM	LocalSearch for minimum cost
mCaM	Min-Communication andMigration Cost
PSO	Particle Swarm Optimization
AppAware	Application-aware
BRITE	Boston universityRepresentative Internet Topology gEnerator

Table 2 Symbols

Symbols	Descriptions
V	The set of virtual machines
S	The set of physical machines
V_i	A virtual machine
S_i	A physical machine
G	Network topological dependence graph of virtual machines
E	Communication correlations between vitual machines
Load	Resource demand of virtual machines
Capacity	Resource capacity of physical machines
Cost	Communication cost of a pair of virtual machines
Distance, D	Delay of hops between physical machines
W	Communication demand of virtual machines
X	Decision variables of virtual machines distribution
Cost_Com	Communication cost of virtual machines
Cost_Com_std	Normalized communication cost of virtual machines
Size	Size of virtual machines
Cost_Mig	Migration cost of virtual machine migrations
Cost_Mig_std	Normalized migration cost of virtual machine migrations
Cost_Total	Total cost of virtual machine migrations, which is weighted sum of the communication cost and the migration cost.

Next is the definition of virtual machine resource demand and physical machine source. $Load(V_i)$ is the resource demand of virtual machine V_i, and $Capacity(S_i)$ represents the CPU, memory, and disk storage capacity of physical machine S_j. The migration cost of virtual machine V_i from physical machine S_k to physical machines S_l is $Cost(V_i, S_k, V_j, S_l) = Distance(S_k, S_l) \times W(V_i, V_j)$, where virtual machines V_i and V_j have communication demand $W(V_i, V_j)$ and $Distance(S_k, S_l)$ represents the delay or hops between S_k and S_l.

To describe the relationship between virtual machines and physical machines, X_{ik} indicates whether virtual machine V_i is distributed to physical machine S_k. If so, then X_{ik} is equal to 1; otherwise, X_{ik} is 0. Similarly, $X_{ik}^{jl} = X_{ik} * X_{jl}$ indicates whether virtual machines V_i and V_j are distributed to physical machines S_k and S_l. If the distribution exists, $X_{ik}^{jl} = 1$; otherwise, it is equal to 0.

On the basis of these definitions, the total communication cost of a datacenter is expressed as

$$\sum_{A} Cost(V_i, S_k, V_j, S_l) \times X_{ik}^{jl} \tag{1}$$

where $A = \{(i,j,k,l \mid i < j,\ k < l,\ j < |V|,\ l < |S|)\}$. Meanwhile, the constraints of the problem are expressed as follows:

$$\sum_{k}^{|S|} X_{ik} = 1, \forall V_i \in O \tag{2}$$

$$\sum_{i}^{|V|} Load_i \times X_{ik} \leq Capacity_k, \forall S_k \in S \tag{3}$$

Condition (2) indicates that every virtual machine must be allocated to a specific physical machine. Condition (3) indicates that the summation of the resource demands of the virtual machines in each physical machine must be lower than the amount of resources the physical machine provides.

In addition, AppAware presumes that the communication correlations between virtual machines is fixed. Thus, AppAware's optimization objective function is

$$Minimize \sum_{A} Cost\left(V_i, S_k, V_j, S_l\right) * X_{ik}^{jl} \tag{4}$$

To simplify this problem, we presume that the allocation of underloaded virtual machines to physical machines is fixed. Thus, the allocation of underloaded virtual machine V_i to physical machine S_k is represented by x_{ik}, which is defined as

$$x_{ik} = C \tag{5}$$

For convenience of description, we call the virtual machine migration problem VMIG. The virtual machine is called VM, and the physical machine is called PM.

Improvement of the problem model

Communication cost model

In our work, the communication cost still follows the definition in (1). Here, we use *Cost_Com* to represent the total communication cost in the datacenter.

$$Cost\_Com = \sum_{A} Cost\left(V_i, S_k, V_j, S_l\right) \times X_{ik}^{jl} \tag{6}$$

As the calculations of the communication cost and migration cost are totally different, the dimensions of these two costs are also different, which unbalances the influences of these two costs to the total cost. To eliminate this unbalance, we need to make sure the values of these two costs in objective function are in the same order of magnitude. Tiwari et al. [25], Zhao et al. [26], to increase the models' extensibility and adaptability and consider the migration cost in the objective function, we normalize the communication cost to a value between 0–1. The normalized communication cost is

$$Cost\_Com\_std = \frac{\sum_{A} Cost\left(V_i, S_k, V_j, S_l\right) \times X_{ik}^{jl}}{e_{Topo\_vm} \times \max\limits_{i,j \in V}\left(W_{ij}\right) \max\limits_{k,l \in S}\left(D_{KL}\right)} \tag{7}$$

where e_{Topo_vm} represents the number of VM pairs that are communicating with each other in the datacenter. It also represents the total number of edges in the network topological dependence graph. Given that each pair of VMs that communicate with each other corresponds to a $Cost\left(V_i, S_k, V_j S_l\right)$, we use the communication cost divided by the number of VM pairs that communicate with each other, the maximum VM communication demand $\left(\max\limits_{i,j \in V}\left(W_{ij}\right)\right)$ and the maximum distance $\max\limits_{k,l \in S}\left(D_{kl}\right)$ between PMs, in which D_{kl} is $Distance\left(S_k, S_l\right)$ and W_{ij} is $W\left(V_i, V_j\right)$.

Migration cost model

We define the migration cost as the product of the amount of VM migration data and the distance between the source and destination PMs. The VM is essentially a software container. It packages a complete set of virtual hardwares and softwares that includes the operating system and all applications. During VM migration, the migration operation is similar to the transfer and duplication of other software files. Therefore, the VM migration cost is related to the VM size, which is called *Size*. The function of the migration cost of VM V_i from S_l is

$$Cost\_Mig(V_i) = \sum_{k \in S} Size_i \times D_{lk} \times X_{ik} \tag{8}$$

where S_l is the physical machine where the overloaded VM V_i initially located. For an overloaded VM that needs to be reallocated, if $X_{ik} = 1$, then VM V_i will migrate to S_k.

The function of the total migration cost of a datacenter is

$$Cost\_Mig = \sum_{i \in O} \sum_{k \in S} Size_i \times D_{lk} \times X_{ik} \tag{9}$$

As the communication cost need to be normalized, the migration cost also need to be normalized to the same order of magnitude with the communication cost. To increase the models' extensibility and adaptability, we also normalize the migration cost. The normalized value will be used in the improvement of the objective function. The normalized migration cost is

$$Cost\_Mig\_std = \frac{\sum\limits_{i \in O} \sum\limits_{k \in O} Size_i \times\ D_{lk} \times X_{ik}}{N_o \times \max\limits_{i \in O}(Size_i) \max\limits_{l,k \in S}(D_{lk})} \tag{10}$$

where $\max_{i \in O}(Size_i)$ is the maximum size of all VMs, and $N_O = |O|$, which is the number of VMs in the overloaded PM. The migration strategy addresses the migration of these overloaded VMs. Thus, the number of migrated VMs following the selection of the migration strategy should be lower than N_O. The total migration cost generated by redeploying VMs onto these PMs is N_O orders of magnitude.

Improvement of the objective function

Next is the definition of the new objective function for this problem. The total cost, *Cost_Total*, is the weighted sum of the communication cost and the migration cost. This both enables our algorithm to estimate the network cost close to reality and ensures the extensibility and adaptability of the algorithm. We use the weight coefficients, α and β (which must sum to 1) to adjust the effect of these two factors in the optimization objective function.

$$Cost\_Total = \alpha \times Cost\_Com\_std + \beta \times Cost\_Mig\_std \tag{11}$$

These two coefficients can balance the optimization of the communication cost and the migration cost effectively. The administrators of datacenters can adjust these two coefficients based on the network condition to have different optimization of the communication cost and the migration cost. In the following experiment, we set $\alpha = 0.7$ and $\beta = 0.3$. With these coefficient values, the performance of our proposed algorithms in simulated datacenters with different number of virtual machines is best in our experiments. We also finished the experiment in "Experiment" section to find the most appropriate coefficient values with the best optimization of the total cost.

Optimization of the virtual machine dynamic migration algorithm

Procedure of the dynamic migration algorithm

Initially, we provide the definition of an overloaded PM. For a PM S_{src}, if the sum of the VM resource demands (*Load*) is larger than the sum of available resources (*Capacity*), S_{src} will be referred to an overloaded PM. And the VMs on it are referred to overloaded VMs. Several of the VMs on overloaded PMs need to be migrated based on a migration algorithm, thus making the PMs underloaded. Following is the general procedure of the dynamic migration algorithm.

Algorithm 1 General procedure of dynamic migration

Input: S_{src}, an overloaded PM; *VMList*, a set of VMs on the PM;
Output: *AllocationList* that consists of every V_{mig} that must be migrated and S_{dest} the destination PM of the migration;

```
VMList ← getVMList(S_src)
while Σ_i^|V| Load_i × X_i src > Capacity_src do
    (V_mig, S_dest) ← Call VM-PM-Choose(S_src)
    if S_dest ≠ S_src then
        Add (V_mig, S_dest) to AllocationList
        X_mig src ← 0, X_mig dest ← 1
    end if
end while
return AllocationList
```

Direct optimization

Direct optimization refers to the optimization of the total cost, which is expressed as (11). We use a greedy strategy to migrate the overloaded VM with the minimum total cost *Cost_Total*, which guarantees that the total cost of the datacenter is also minimal.

Based on the direct optimization method, we propose the local search for min-cost algorithm (LM) to optimize the datacenter performance. For each overloaded VM V_i, LM calculates the total cost to all the PMs and selects the destination PM with the lowest total cost, $Cost(V_i, S_k)$. By minimizing each VM migration total cost, the total cost of the datacenter can be made as low as possible.

Indicated below are the functions of communication cost and migration cost of VM V_i to PM S_k, which is $Cost\_Com(V_i, S_k)$ and $Cost\_Mig(V_i, S_k)$.

$$Cost\_Com(V_i, S_k) = \frac{\sum_{\forall j \in TW(V_i)} Cost(V_i, S_k, V_j, S_l) \times X_{ik}^{jl}}{|TW(V_i)| \times \max_{A}(Cost)} \tag{12}$$

$$Cost\_Mig(V_i, S_k) = \frac{Size_i \times D_{lk} \times X_{ik}}{\max_{i \in O}(Size_i) \max_{l,k \in S}(D_{lk})} \tag{13}$$

where $|TW(V_i)|$ represents the number of VMs that communicate with V_i, such as V_j on the PM S_l. $\max_{i \in O}(Size_i)$ is the maximum *Size*, and $\max_{l,k \in S}(d_k)$ is the maximum *Distance* in the PM topological structure.

According to Functions (12) and (13), $Cost\_Total(V_i, S_k)$ is the weighted summation of these two costs. The detailed function is expressed as follows

$$\begin{aligned} Cost\_Total(V_i, S_k) = {} & \alpha \times Cost\_Com(V_i, S_k) \\ & + \beta \times Cost\_Mig(V_i, S_k) \end{aligned} \tag{14}$$

On the basis of these functions, the LM algorithm is designed as shown below.

Algorithm 2 LM (LocalSearch for min-cost)

Input: S_{src}, an overloaded PM; *VMList*, a set of VMs on the PM

Output: (V_{mig}, S_{dest}): VM V_{mig} will migrate from PM S_{src} to the destination PM S_{dest}.

$Cost[1, 2, \ldots, |V|]$: Minimum costs of the VMs in *VMList*

$Des[1, 2, \ldots, |V|]$: Destination PMs of the VMs in *VMList*

$Cost_{min} \leftarrow \infty$
for $V_i \in VMList$ **do**
  if $Check\_PM\_Constraints(V_i, Sj) = True$ **then**
    break
  end if
  $Cost_i \leftarrow \infty$
  for $S_j \in S$ **do**
    $Cost_{tmp} \leftarrow Compute\_Impact(V_i, S_j)$
    if $Cost_{tmp} < Cost_i$ **then**
      $Cost_i \leftarrow Cost_{tmp}$
      $Des_i \leftarrow S_j$
    end if
  end for
  if $Cost_i < Cost_{tmp} \&\& Des_i \neq S_{src}$ **then**
    $Cost_{min} \leftarrow Cost_i$
    $V_{mig} \leftarrow V_i$
    $S_{dest} \leftarrow Des_i$
  end if
end for
return (V_{mig}, S_{dest})

The time complexity of LM is $o(m \cdot n)$, where m is the number of VMs on PM S_m and n is the number of PMs. Compared with AppAware, the advantages of the LM algorithm mainly lie in two aspects. The first aspect is the migration object. AppAware migrates all the overloaded VMs at once. By contrast, LM only migrates a small number of the VMs and makes the overloaded PM's capacity just larger than the sum of the load. The second aspect is the optimization effect of the network cost. AppAware only consider the VM communication cost, but the LM considers the optimization of both the communication cost and the migration cost.

Difference value optimization

LM optimizes the total network cost with the communication cost and migration cost, that is

$$\min(Cost\_Total) \Leftrightarrow \min(\alpha \cdot Cost\_Com + \beta \cdot Cost\_Mig) \tag{15}$$

Before VM migration, the set of VMs has an unreasonable original allocation plan within the set of PMs. According to Function (4), the allocation plan corresponds to a communication cost C_0. Based on the VM migration algorithm, overloaded VMs are migrated and a new VM allocation plan is generated with a communication cost called C. In theory, the purpose of VM migration is to minimize communication costs and migration costs. Specifically, the algorithm should maximize the difference in the communication cost, $\Delta C = C_0 - C$, and minimize the migration cost, *Cost_Mig*.

After importing ΔC, the optimization object becomes $\min(\alpha \cdot C_0 - \alpha \cdot C + \beta \cdot Cost\_Mig)$. Here, we define the new optimization objective function as

$$\max(\alpha \cdot \Delta C - \beta \dot{C}ost\_Mig) \tag{16}$$

We define the new optimization objective function as the network benefit, *Benefit*.

$$Benefit = \alpha \cdot \Delta C - \beta \cdot Cost\_Mig \tag{17}$$

In the new optimization objective function, the network benefit, $Benefit(V_i, S_k)$, is expected to be the maximum for each VM migration. This is essentially different from LM, whose object is the minimization of the communication cost and migration cost for each migration.

Algorithm mCaM

mCaM imports the prediction mechanism in LM. Initially, mCaM finds the overloaded PM S_m based on the general procedure. Then, mCaM finds the overloaded VM migration with the maximum network benefit. In detail, mCaM calculates the minimum $Cost\_Total(V_i, S_k)$ for migrating each VM V_i to PM S_k. Then, mCaM calculates the network benefit $Benefit(V_i, S_k)$ for all the overloaded VMs and migrates VM V_i with the maximum network benefit to the destination PM S_k.

Below is the detailed pseudocode of mCaM.

mCaM can ensure that the migrated VM V_i has the highest possibility of reducing the communication cost. The time complexity of this algorithm is $o(m \cdot n)$, where m is the number of VMs on PM S_m and n is the number of PMs.

Algorithm mCaM2

Now we analyse the network benefit of each VM migration. mCaM migrates the VM with the maximum network benefit to the PM with the minimum total cost, until all PMs become underloaded. However, mCaM may never migrate those VMs with both high network benefit and high migration cost. For the datacenters with high communication costs, mCaM can only optimize the VMs that have low migration cost, which has a limited optimization on the communication performance. Here, we propose mCaM2 to solve this problem.

First, we describe the optimization objective of mCaM2. We define the reduction of communication cost as communication benefit, $Benefit\_Com = \Delta C$. To describe the

Algorithm 3 mCaM (Min-Communication and Migration Cost)

Input: S_{src}, an overloaded PM; *VMList*, a set of VMs on the PM

Output: (V_{mig}, S_{dest}): VM V_{mig} will migrate from PM S_{src} to destination PM S_{dest}.

$Cost[1, 2, \ldots, |V|]$: Minimum costs of the VMs in *VMList*

$Benefit[1, 2, \ldots, |V|]$: Maximum network benefits of the VMs in *VMList*

$Des[1, 2, \ldots, |V|]$: Destination PMs of the VMs in *VMList*

$Benefit_{max} \leftarrow 0$
for $V_i \in VMList$ **do**
  if $Check\_PM\_Constraints(V_i, Sj) = True$ **then**
    break
  end if
  $Benefit_i \leftarrow 0$
  $Cost_i \leftarrow \infty$
  for $S_j \in S$ **do**
    $Cost_{tmp} \leftarrow Compute\_Impact(V_i, S_j)$
    if $Cost_{tmp} < Cost_i$ **then**
      $Cost_i \leftarrow Cost_{tmp}$
      $Des_i \leftarrow S_j$
    end if
  end for
  $Benefit_i \leftarrow \alpha \times Cost\_Com(V_i, S_m) - \beta \times Cost_i$
  if $Benefit_i > Benefit_{max}$ **then**
    $Benefit_{max} \leftarrow Benefit_i$
    $V_{mig} \leftarrow V_i$
    $S_{dest} \leftarrow Des_i$
  end if
end for
return (V_{mig}, S_{dest})

Algorithm 4 mCaM2 (Min-Communication and Migration Cost)

Input: S_{src}, an overloaded PM; *VMList*, a set of VMs on the PM

Output: (V_{mig}, S_{dest}): VM V_{mig} will migrate from PM S_{src} to destination PM S_{dest}.

$Benefit\_Com[1, 2, \ldots, |V|]$: Maximum communication benefits of the VMs in *VMList*

$Benefit\_Ratio[1, 2, \ldots, |V|]$: Benefit ratios of the VMs in *VMList*

$Des[1, 2, \ldots, |V|]$: Destination PM of the VMs in *VMList*

$Benefit\_Ratio_{max} \leftarrow 0$
for $V_i \in VMList$ **do**
  if $Check\_PM\_Constraints(V_i, Sj) = True$ **then**
    break
  end if
  $Benefit_i \leftarrow 0$
  $Benefit\_Com_i \leftarrow 0$
  for $S_j \in S$ **do**
    $Benefit\_Com_{tmp} \leftarrow Compute\_Impact(V_i, S_j)$
    if $Benefit\_Com_{tmp} < Benefit\_Com_i$ **then**
      $Cost_i \leftarrow Benefit\_Com_{tmp}$
      $Des_i \leftarrow S_j$
    end if
  end for
  $Benefit\_Ratio_i \leftarrow \frac{Benefit\_Com(V_i, S_k)}{Cost\_Mig(V_i, S_k)}$
  if $Benefit\_Ratio_i > Benefit\_Ratio_{max}$ **then**
    $Benefit\_Ratio_{max} \leftarrow Benefit\_Ratio_i$
    $V_{mig} \leftarrow V_i$
    $S_{dest} \leftarrow Des_i$
  end if
end for
return (V_{mig}, S_{dest})

benefit of the migration, we define the optimization object as *Benefit_Ratio*, which is

$$Benefit\_Ratio(V_i) = \frac{Benefit\_Com(V_i, S_k)}{Cost\_Mig(V_i, S_k)} \tag{18}$$

Second, we describe the migration strategy of mCaM2. mCaM2 divides the optimization into two periods. In the first period, mCaM2 optimizes the communication cost. It calculates the communication benefit *Benefit_Com* for all overloaded VMs to all PMs. For each VM, mCaM2 selects the PM with maximum communication benefit as the destination PM. In the second period, mCaM2 optimizes the migration cost. mCaM2 calculates the benefit ratio *Benefit_Ratio* for each overloaded VM and migrates the VM with the maximum benefit ratio, until all PM become underloaded.

The time complexity of mCaM2 is $o(m \cdot n)$, where m is the number of VMs on PM S_m and n is the number of PMs. The operation process of mCaM2 is shown below.

Experiment

Introduction of the experimental environment

We tested our algorithms in an environment, that simulates a datacenter with multilayer virtual machines. For the VM communication correlations, we use the Boston University Representative Internet Topology Generator (BRITE) to generate the bandwidth dependence relationship between VMs. For the topology of the PMs, we used a topological generation algorithm to generate a tree structure. The experiment was executed on an Ubuntu Server 11.04, and each server is a Sugon blade server that has an octuple core CPU (two Intel Xeon E5506 2.13GHz 8) and 16 GB 1333 MHz memory.

BRITE generation of the bandwidth dependence relationship between VMs

BRITE [27] is a general network topology generator built by Alberto Medina et al. at Boston University. Through BRITE, we can simulate the topological structure, such

as the hierarchical model, and the communication dependence distribution of the simulated network topology. In addition, BRITE can also use some network models to generate simulated network topologies, such as Router Waxman and Router BarabasiAlert. Given that the VMs we consider are locally homogenous, we only need to use the generation function of the LAN topology in BRITE.

Physical topology structure

In the present datacenters, the network topology usually extends from triple-layer network structure [22]. In the bottom layer, every PM connects to one (or two) access switch. In the aggregation layer, the access switch connects to one or two switches, and every aggregation layer switch connects to multiple core layer switches. We call this structure a tree structure. The aggregation layer is usually a tree structure, and the root node is located in the core layer. If this triple-layer network structure is a multi-tree structure, the data package will be transferred based on the aggregation layer VLAN and the maximum weight spanning tree.

Algorithm experiment

Algorithm parameter configuration

With different numbers of VMs and workloads, we finish the VM migration algorithm simulation. We compared the performance of AppAware, LM, mCaM and mCaM2 in the simulated environment. In our experiments, the VM overload ratio is 0.4, which indicates the ratio of overloaded VM set O in VM set V. The distributions of VM size, VM load and PM capacity are normal distribution. The average VM size is 0.4GB with a variance of 0.2, the average VM resource demands is 0.4GB with a variance of 0.1, and the average PM capacity is 0.8GB with a variance of 0.2. The bandwidth dependence relationship between VMs is generated by BRITE automatically, and includes the weight parameter W. The number of VM is change from 60 to 200 in steps of 20, and the number of PMs is 100. The following experiment results are the average of 10 operations.

Weight coefficient in the dynamic migration objective function

For these migration algorithms, the weight coefficients α and β are vital to the optimization effect. The weight coefficient can effectively balance the optimization of communication cost and migration cost. Therefore, we need to find the appropriate weight coefficient values. Given that the value of the objective function reflects the effect of migration, the difference between maximum and minimum needs to be as small as possible. Here, we use different values of α(from 0.0 to 1.0 in steps of 0.1) to calculate the total cost for the VMs (the number of VMs increased from 60 to 200 in steps of 20). To obtain a better weight coefficient value conveniently, we calculate the differences between the maximum and minimum total costs in the weight coefficient experiments, and the result is shown in Fig. 1, which describes the performance of each weight coefficient.

As shown in Fig. 1, when $\alpha = 0.7$, the total cost of the VIMG experiment has a minimum different value of 0.014733, in which the maximum value is 0.390296 from 100 VM application requests, and the minimum value is 0.379792 from 80 VM application requests. According to the results of the simulation experiment, the weight coefficients in the VMIG model are set to $\alpha = 0.7$, and $\beta = 0.3$ in the following experiment.

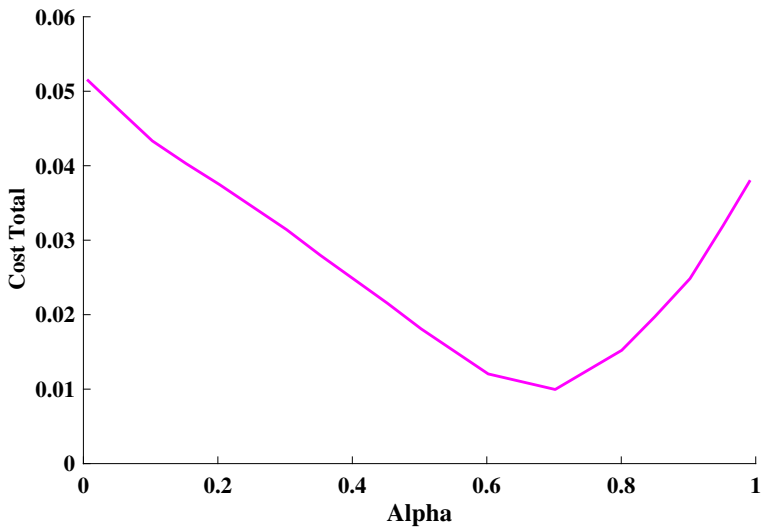

Fig. 1 Difference value weight factor adjustment

Table 3 Object function values

Number of VMs							
Algorithm	60	80	100	120	140	160	200
LM	0.366	0.359	0.403	0.392	0.395	0.409	0.397
mCaM	0.364	0.352	0.395	0.383	0.388	0.401	0.389
mCaM2	0.417	0.392	0.439	0.417	0.422	0.422	0.400

Optimization algorithm performance comparison

Based on appropriate weight coefficients, we compared the total cost of each algorithm, including LM, mCaM and mCaM2. The detailed data are shown in Table 3 and Fig. 2.

As shown in Table 3 and Fig. 2, mCaM has the minimum total cost among the three algorithms. This indicates that mCaM has better optimization for datacenter and has lowest migration cost in the VMIG. Given that LM directly optimizes the total cost for each VM migration, it only migrates VMs with high communication costs to PMs with low migration costs, which ignores the communication cost history information of VMs in the original network. Therefore, LM may never migrate VMs that have high communication cost, resulting in a limited optimization of the total cost.

However, mCaM changes the form of the objective function. In the algorithm process, mCaM focuses on the network benefit to primarily optimize the communication performance. To have a better optimization, mCaM calculates the network benefits for each overloaded VM and migrates the VM with the maximum network benefit. Therefore, mCaM solves the limitations of LM. It maximizes the migration performance benefit and has a lower total cost after optimization.

mCaM2 considers the reduction of communication cost after migration to maximize the network performance. mCaM2 separates the optimization into two periods: the selection of destination PM and the selection of migration VM. In the selection of the destination PM, only the reduction of communication cost is considered. Migration cost is considered in the selection of migration VM. It exerts a negative influence on the migration cost and increases the total cost. Therefore, the total cost of mCaM2 is higher than the other two algorithms.

To compare the communication cost and migration cost optimization effect, we also analysed the communication cost and the migration cost of these three algorithms. Figures 3 and 4 show the detailed communication cost and migration cost values with different numbers of VMs.

Figure 3 shows that mCaM2 has the best communication cost optimization performance among the three algorithms, followed by mCaM. LM directly optimizes the total cost. In every VM migration, LM only focuses on the communication cost after VM migration and never considers the communication cost before migration.

In the migration VM selection, mCaM considers the VM cost information in the original network and migrates the VM with the greatest network benefit. Therefore, it has a better communication cost optimization than LM. mCaM2 considers the costs separately in different periods. In the destination PM selection period, mCaM2 only considers the optimization of the communication cost. It ensures that the communication cost can reduce as much as possible. Therefore, mCaM2 has the minimum communication cost.

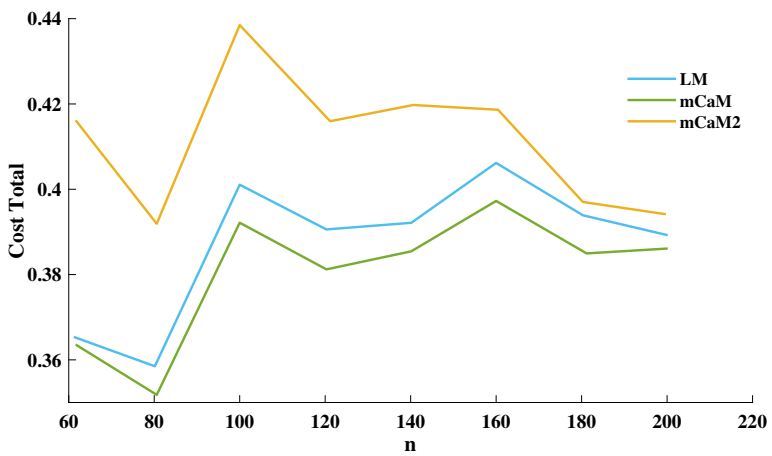

Fig. 2 Comparison of objective function values

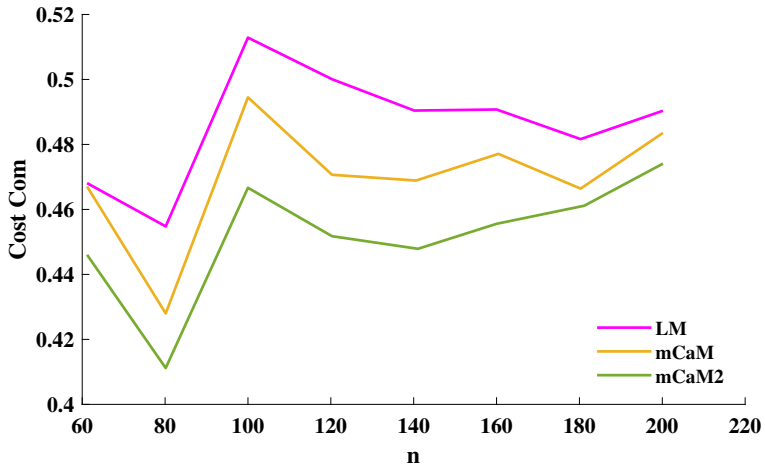

Fig. 3 Comparison of communication costs

As shown in Fig. 4, LM has the smallest migration cost, followed by mCaM, but the difference is small. Given that the destination PM selection strategy ignores migration cost, mCaM2 has the largest migration cost. In conclusion, mCaM has the highest network optimization performance in terms of both communication cost and migration cost among the three algorithms. mCaM2 optimizes communication costs at most, but the migration cost of it is higher than that of mCaM. LM has the lowest migration cost, but the optimization in communication costs is the highest among the three algorithms.

Comparison with AppAware in terms of performance

We also compared the total cost of mCaM with that of AppAware. We list the statistical results in total cost in Table 4 and Fig. 5.

Figure 5 shows that mCaM's total cost reduces sharply compared with that of AppAware, and Table 4 shows that mCaM has a 20% lower total cost than AppAware on average. Given that AppAware only optimizes the communication cost and ignores the migration cost, the result satisfies our expectations.

Figure 6 shows the experimental results for AppAware and mCaM in terms of communication cost. As shown in this figure, given that AppAware only aims to optimize the communication cost, AppAware performs better than mCaM.

Next, we analysed performance in terms of the migration cost. Figure 7 presents the statistical results of these two algorithms in the simulation experiment.

The bar graph in Fig. 7 represents the number of migrations made by the two algorithms. The curves in Fig. 7 represent the corresponding migration costs. Evidently, mCaM has a much lower migration cost than AppAware, and its number of migrations is also smaller than that of AppAware by approximately 50%.

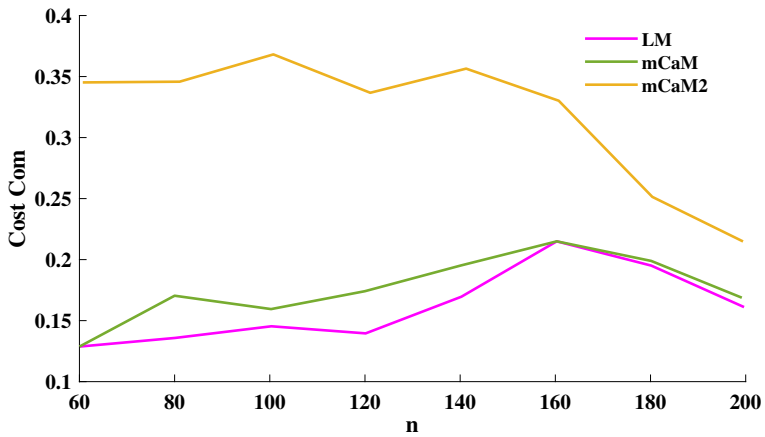

Fig. 4 Comparison of migration costs

Table 4 Comparison of total costs

Number of VMs Algorithm	60	80	100	120	140	160	200
AppAware	0.482	0.456	0.494	0.476	0.490	0.480	0.431
mCaM	0.364	0.352	0.395	0.383	0.388	0.401	0.389
Improvement over AppAware	24.48%	22.81%	20.04%	19.54%	20.82%	16.46%	9.74%

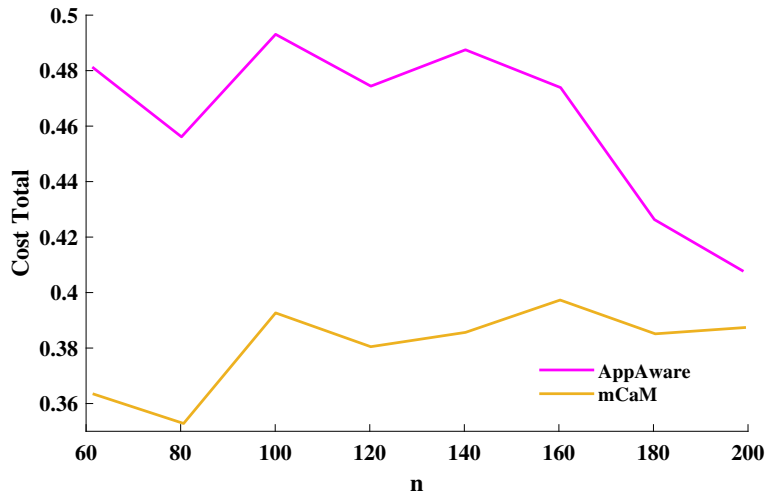

Fig. 5 Comparison of total network costs

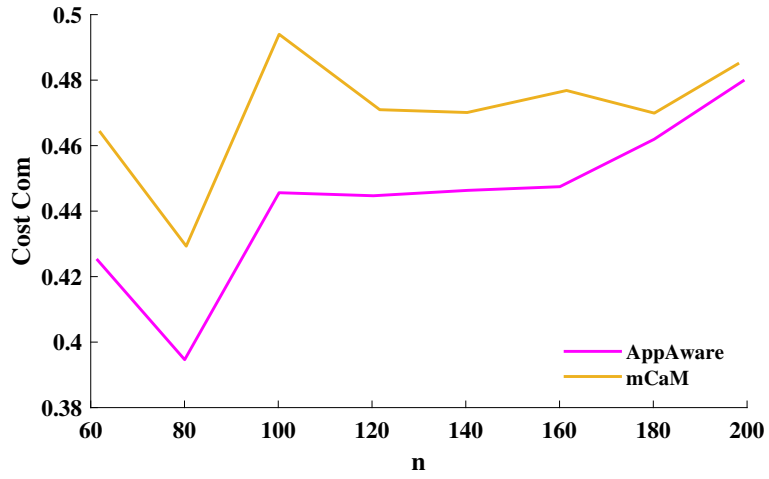

Fig. 6 Comparison of communication costs

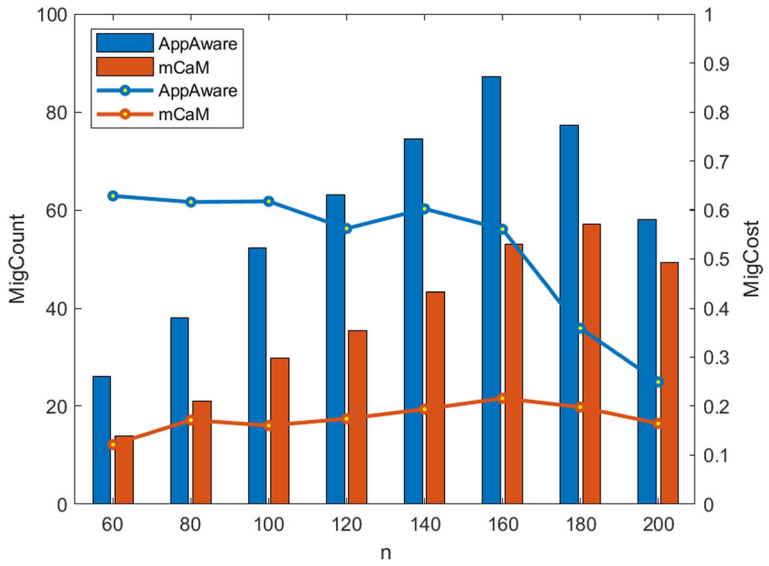

Fig. 7 Comparison of migration costs and times

mCaM fully considers the influence of migration cost and uses the lowest migration cost to obtain the largest network benefit. AppAware only optimizes the communication traffic and directly migrates all VMs on the overloaded PM, which easily generates frequent VM migration. The curve of the number of migrations proves this point. Finally, we compare the operation times of these two algorithms. LM, mCaM and mCaM2 have the same time complexity. Therefore, LM, mCaM and mCaM2 has the similar operation times, and we only compared the operation times between mCaM and AppAware. The detailed data are shown in Fig. 8.

As shown in Fig. 8, mCaM exhibits a clear improvement in operation time compared with AppAware by 70% on average. AppAware migrates all the VMs on the overloaded PM. However, mCaM only migrates some of the overloaded VMs in order to just make the PM underloaded. Therefore, mCaM has a much smaller operation time. Furthermore, mCaM considers the balance between migration cost and network benefit. Therefore, mCaM migrates the VM with the highest benefit, which also decreases the operation time. In conclusion, the advantages of mCaM in operation time increase with the number of VMs and the scale of the problem.

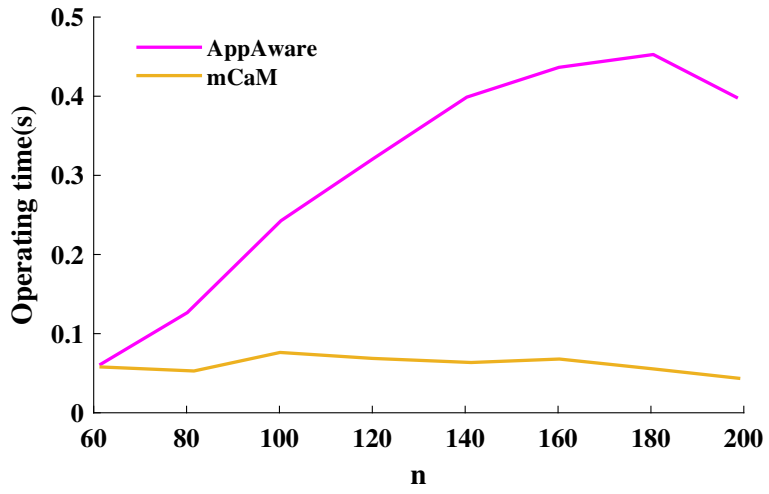

Fig. 8 Comparison of algorithm operation times

Conclusion

In this paper, we focus on the network traffic created by the VM migrations, which profoundly limit the performance of the datacenters. To solve this problem, we propose three heuristics VM migration algorithms, LM, mCaM and mCaM2, to balance the optimization effect of communication cost and migration cost. The simulation experiments demonstrate that it efficiently decreases network traffic of the datacenters by 20% comparing with AppAware. And our algorithms also outperform AppAware in terms of operation time by 70% on average. Our simulation results show that the proposed algorithms can effectively decrease the energy consumption in the cloud-edge fusion computing environment. One promising future direction is to implement and evaluate our algorithms on a real cloud-edge fusion computing environment.

Abbreviations

PSO: Particle Swarm Optimization; VMIG: Virtual Machine Migration; *V*: The set of virtual machines; *S*: The set of physical machines; V_i: A virtual machine; S_i: A physical machine; *G*: Network topological dependence graph of virtual machines; *E*: Communication correlations between vitual machines; *Load*: Resource demand of virtual machines; *Capacity*: Resource capacity of physical machines; *Cost*: Communication cost of a pair of virtual machines; *Distance*, *D*: Delay of hops between physical machines; *W*: Communication demand of virtual machines; *X*: Decision variables of virtual machines distribution; *Cost_Com*: Communication cost of virtual machines; *Cost_Com_std*: Normalized communication cost of virtual machines; *Size*: Size of virtual machines; *Cost_Mig*: Migration cost of virtual machine migrations; *Cost_Mig_std*: Normalized migration cost of virtual machine migrations; *Cost_Total*: Total cost of virtual machine migrations, which is weighted sum of the communication cost and the migration cost.; VM: virtual machine; PM: physical machine; VMIG: Virtual Machine Migration; LM: LocalSearch for minimum cost; mCaM: Min-Communication andMigration Cost; PSO: Particle Swarm Optimization; AppAware: Application-aware; BRITE: Boston universityRepresentative Internet Topology gEnerator

Acknowledgements

We are thankful to Dr. Weizhe Zhang School of Computer Science and Technology, Harbin Institute of Technology for his valuable contribution towards the proposed work.

Authors' contributions

All authors have participated in conception and design, or analysis and interpretation of the data, drafting the article or revising it critically for important intellectual content, approval of the final version.

Funding

This work was supported in part by the Key Research and Development Program for Guangdong Province (2019B010136001), Shenzhen Science and Technology Research and Development Foundation (No. JCYJ20190806143418198), the National Key Research and Development Program of China under Grant 2017YFB0801801, and the National Natural Science Foundation of China (NSFC) under Grant 61672186 and Grant 61872110. Professor Zhang is the corresponding author.

Availability of data and materials

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Competing interests

The author declares that he has no competing interests.

Author details

[1]School of Computer Science and Technology, Harbin Institute of Technology, Xidazhi Street, Harbin, China. [2]Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China. [3]School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane QLD, Australia.

Received: 4 November 2019 Accepted: 23 July 2020
Published online: 03 August 2020

References

1. Dasgupta G, Sharma A, Verma A, Neogi A, Kothari R (2011) Workload management for power efficiency in virtualized data centers. Commun ACM 54(7):131–141
2. Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Futur Gener Comput Syst 28(5):755–768
3. Shrivastava V, Zerfos P, Lee K.-w., Liu Y, Jamjoom H, Liu Y-H, Banerjee S (2011) Application-aware virtual machine migration in datacenters. In: Ni L, Zhang W (eds). 2011 Proceedings IEEE INFOCOM. IEEE, Toronto. pp 66–70
4. Zhang X, Shae Z-Y, Zheng S, Jamjoom H (2012) Virtual machine migration in an over-committed cloud. In: TurcK FD, Gaspary LP, Medhi D (eds). IEEE Network Operations and Management Symposium. IEEE, Maui. pp 196–203
5. Wen X, Chen K, Chen Y, Liu Y, Xia Y, Hu C (2012) Virtualknotter: Online virtual machine shuffling congestion resolving in virtualized datacenter. In: Zhao W, Lai TH (eds). IEEE 32nd International Conference on Distributed Computing Systems. IEEE, Macau. pp 12–21
6. Heller B, Seetharaman S, Mahadevan P, Yiakoumis Y, Sharma P, Banerjee S, McKeown N (2010) Elastictree: saving energy in data center networks. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation. ACM, Berkeley. pp 17–17
7. Choudhary A, Govil MC, Singh G, Awasthi LK, Pilli ES, Kapil D (2017) A critical survey of live virtual machine migration techniques. J Cloud Comput 6:23
8. Kumar M, Sharma SC, Goel A, Singh SP (2019) A comprehensive survey for scheduling techniques in cloud computing. J Netw Comput Appl 143:1–33
9. Verma A, Ahuja P, Neogi A (2008) pmapper: power and migration cost aware application placement in virtualized systems. In: Middleware 2008. Springer, Berlin Heidelberg. pp 243–264
10. Goudarzi H, Ghasemazar M, Pedram M (2012) Sla-based optimization of power and migration cost in cloud computing. In: Majumdar S (ed). 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing(CCGrid). IEEE, Ottawa. pp 172–179
11. Lefèvre L, Orgerie A-C (2010) Designing and evaluating an energy efficient cloud. J Supercomput 51(3):352–373
12. Moges FF, Abebe SL (2019) Energy-aware vm placement algorithms for the openstack neat consolidation framework. J Cloud Comput 8:2
13. Greenberg A, Hamilton JR, Jain N, Kandula S, Kim C, Lahiri P, Maltz D, Patel P, Senguptag S (2009) Vl2: a scalable and flexible data center network. In: Rodriguez P, Biersack E (eds). ACM SIGCOMM Computer Communication Review. ACM, New York. pp 51–62
14. Zhani MF, Zhang Q, Simona G, Boutaba R (2013) Vdc planner: Dynamic migration-aware virtual data center embedding for clouds. IEEE, Ghent
15. Singh S, Chana I, Singh M, Buyya R (2016) SOCCER: Self-optimization of energy-efficient cloud resources. Clust Comput 19:1787–1800
16. Mishra SK, Mishra S, Bharti SK, Sahoo B, Puthal D, Kumar M (2018) VM selection using DVFS technique to minimize energy consumption in cloud system. In: 2018 International Conference on Information Technology (ICIT). IEEE, Bhubaneswar. pp 284–289
17. Kumar M, Sharma SC (2018) PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint. Sustain Comput Inform Syst 19:147–164
18. Gawali MB, Shinde SK (2018) Task scheduling and resource allocation in cloud computing using a heuristic approach. J Cloud Comput 7:4
19. Kliazovich D, Bouvry P, Khan SU (2010) Dens: data centerenergy-efficient network-aware scheduling. In: IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing. IEEE, Hangzhou. pp 69–75
20. Kumar M, Sharma SC (2018) Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. Comput Electr Eng 69:395–411

21. Kumar M, Sharma SC (2019) PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing. Neural Comput Appl 32(16):12103–12126
22. Meng X, Pappas V, Zhang L (2010) Improving the scalability of data center networks with traffic-aware virtual machine placement. In: Mandyam G, Westphal C (eds). 2010 Proceedings IEEE INFOCOM. IEEE, San Diego. pp 1–9
23. Jayasinghe D, Pu C, Eilam T, Steinder M, Whally I, Snible E (2011) Improving performance and availability of services hosted on iaas clouds with structural constraint-aware virtual machine placement. IEEE, Washington, DC
24. Wood T, Shenoy P, Venkataraman A, Yousif M (2009) Sandpiper: Black-box and gray-box resource management for virtual machines. Comput Netw 53(2009):2923–2938
25. Tiwari PK, Joshi S (2016) Dynamic weighted virtual machine live migration mechanism to manages load balancing in cloud computing. In: 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). IEEE. pp 1–5
26. Zhao D, Yang T, Jin Y, Xu Y (2017) A service migration strategy based on multiple attribute decision in mobile edge computing. In: 2017 IEEE 17th International Conference on Communication Technology (ICCT). IEEE, Chengdu. pp 986–990
27. Medina A, Lakhina A, Matta I, Byers J (2001) Brite: An approach to universal topology generation. IEEE, Cincinnati

Publisher's Note