A Virtual Machine Migration Algorithm Based on Group Selection in Cloud Data Center

Zhen Guo, Wenbin Yao, Dongbin Wang

▶ **To cite this version:**

Zhen Guo, Wenbin Yao, Dongbin Wang. A Virtual Machine Migration Algorithm Based on Group Selection in Cloud Data Center. 14th IFIP International Conference on Network and Parallel Computing (NPC), Oct 2017, Hefei, China. pp.24-36, 10.1007/978-3-319-68210-5_3 . hal-01705444

HAL Id: hal-01705444
https://hal.inria.fr/hal-01705444

Submitted on 9 Feb 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

A Virtual Machine Migration Algorithm Based on Group Selection in Cloud Data Center

Zhen Guo[1,3], Wenbin Yao[1,3], Dongbin Wang[2,3]

[1] Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia
[2] National Engineering Laboratory for Mobile Network Security
[3]Beijing University of Posts and Telecommunications, Beijing 100876, China
{guozhen402, yaowenbin_cdc}@163.com, dbwang@bupt.edu.cn

Abstract. Live migration of virtual machine (VM) is a promising technology that helps physical machines (PMs) adapt to load changes and guarantees Quality of Service (QoS) in cloud data center. Many individual-based VM migration studies ignore the association between VMs, resulting in high communication cost. Some research on multiple VMs migration migrates the VM group as a whole, which is likely to result in ineffective migration and increase the network burden. In this paper, a VM migration algorithm based on group selection (VMMAGS) is proposed, which takes into account the migration cost, communication cost, and VM heat to optimize migration performance. The appropriate VM groups are selected as migration options, and the optimal migration scheme is obtained according to the integration cost of partitions of selected VM groups. Extensive experiments show that our algorithm can effectively reduce the migration cost and communication cost, improve the system reliability compared with other related algorithms.

1 Introduction

Virtualization [1] is a rapidly evolving technology that enables flexible allocation of resources in cloud data centers [2]. VMs are created according to the amount of required resources and then run on a PM to host application to meet requirements of customers [3]. However, the application load changes constantly in the cloud computing environment, which is likely to cause SLAs violations and affect QoS. Therefore, some VMs on the overloaded PM need to be migrated, so as to ensure the stable operation of cloud data center.

In recent years, the VM migration problem has received much attention. Many individual-based VM migration studies [4] are presented to achieve optimal migration. Shrivastava [5] took the single VM as the migration object, and realized the remapping of VM individual and the PM according to the communication cost. The authors in [6] [7] proposed a multi-objective VM migration algorithm to optimize traffic between VMs, while minimizing the frequency of migration. But they ignored the overhead of the migration itself. More importantly, these individual-based migration strategies will result in higher communication cost due to the association between VMs.

Although some studies take into account the association between VMs, such as [8], which takes the entire associated VM group as the migration object. However, such

migration is likely to result in ineffective migration and increase the network burden. Sun [9] focused on the efficient online live migration of multiple correlated VMs to optimize system performance. However, the VM groups to be migrated were not obtained according to the resource states of the data center, but were given as known conditions. When VM migration is performed, the appropriate VM group should be selected as the migration object to ensure low communication cost and migration time.

An excellent migration strategy should also provide users with better service. Although the VM that is being migrated does not suspend execution during live migration, its execution may become slowed down somewhat due to the migration. Many studies [10], [11] do not take into account the operating state of the VM during the migration, so that the VM that needs to be migrated may be dealing with high-intensity tasks, which will not only result in a higher dirty page rate, but also greatly affect the response time of the PM. We use the VM heat to reflect the operating state of the VM and take it into consideration to guarantee the better service provided to users.

In this paper, a VM migration algorithm based on group selection (VMMAGS) is proposed. The association between VMs and the resource utilization of the VM are taken into account. According to the resource status of the overloaded PM and the degree of connectivity (DoC) of the remaining VMs, the algorithm selects the appropriate VM group as the migration object. The optimal migration scheme is obtained based on the integration cost of the partitions of selected VM groups.

The remainder of the paper is organized as follows. In Sect.2, we investigate the problem of the migration of VMs and present the definition of objective functions. Section 3 describes the proposed algorithm. An empirical evaluation is presented in Sect.4, and Sect.5 concludes the paper.

2 Problem and Objectives Description

2.1 Problem Description

When resources of the PM are tight, migrate some VMs on the overloaded PM to ensure that the remaining VMs and the migrated VMs can both work properly. Different migration strategies will produce different migration results, and the results directly affect the performance of the data center. Fig.1 shows two different migration solutions. PM_1 is overloaded, and some VMs on it need to be migrated. In Fig.1 (a), calculate the optimal migration scheme for the single VM. First, VM_3 is migrated to PM_2 which is closer to PM_1. Next, VM_4 is selected for migration. Since PM_2 does not have enough resources to place VM_4, PM_3 is selected as its target PM. As can be seen from Fig.1, this migration solution is likely to result in a higher communication cost between VM_3 and VM_4. In Fig.1 (b), VM_3 and VM_4 are both migrated to PM_3. This migration solution guarantees a lower communication cost. Based on the above analysis, we should take the VM group as the migration object, rather than the VM individual. The work of this paper is to select the best migration VM group for overloaded PM and find the target PM for each VM in the group, so as to reduce the migration cost, communication cost and VM heat.

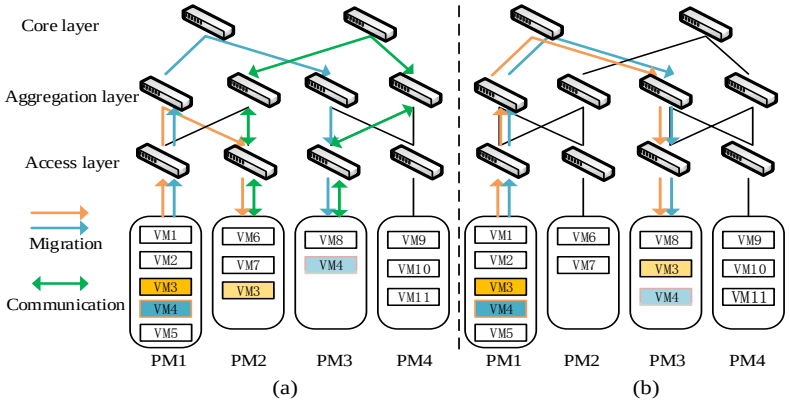


Fig. 1. Two different migration solutions.

2.2 The VM Model

In order to cooperate with each other in handling tasks, there may be frequent communication between VMs. Therefore, we model the associated VMs as an undirected graph *G(V,E)*, in which vertices represent VMs and the edge value represents traffic between VM pairs. The attributes of VM_i include its source PM, the requirements for CPU and RAM, denoted by $(PM_{i,src}, vc_i, vm_i)$. Without loss of generality, it is assumed that VMs on the same PM are connected and that VMs on different PMs may also be associated. So the VM model is shown in Fig.2.

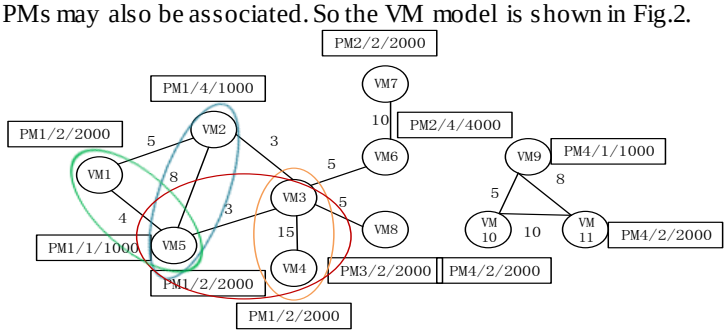


Fig. 2. The VM model.

2.3 Objective Functions Definition

The focus of the paper is to reduce the migration cost, communication cost and the VM heat during the migration process, so we first quantify these objective functions.

Migration Cost. We use the pre-copy strategy to migrate a VM. In the process of live migration, the dirty pages are transferred from the source PM to the target PM through continuous iterations. The longer the migration time, the more resources it occupied, and the greater the impact on network link communication. Therefore, we use the total migration time to reflect the migration cost.

Assume the pre-copy algorithm proceeds in n_i+1 rounds. The amount of data transmitted and transmission time for VM_i in the k-round is $v_{i,k}$ and $T_{i,k}$ ($0 \le k \le n_i$), respectively. The entire memory of the VM needs to be transferred to the target PM in the 0-round, so $v_0 = vm_i . v_{i,k} (k \neq 0)$ is determined by the dirty page generated by the previous round. So $v_{i,k} = r_i T_{i,k-1}$ ($1 \le k \le n_i$), where r_i is page dirty rate, and $T_{i,k} = v_{i,k} / B_i = (vm_i / B_i) \cdot (r_i / B_i)^k$. Therefore, the total time T_i is calculated as:

$$T_i = \sum_{k=0}^{n_i} T_{i,k} = \frac{vm_i}{B_i} \cdot \frac{1-(\frac{r_i}{B_i})^{n_i+1}}{1-\frac{r_i}{B_i}} \quad . \tag{1}$$

n_i is calculated according to the memory threshold vm_{th}. The iteration is stopped when the threshold is reached. So we can obtain:

$$n_i = \left\lceil \log_{\frac{r_i}{B_i}} (\frac{vm_{th}}{vm_i}) \right\rceil \quad . \tag{2}$$

The downtime of VM_i in the migration process is represented as $T_{down\_i} = T_{d\_i} + T_{resume\_i}$. T_{d_i} is the time of transferring the remaining dirty pages, and T_{resume_i} denotes the time spent on resuming the VM on the target PM. Therefore, the migration cost of VM_i is calculated as:

$$Cost\_mig(VM_i) = \frac{vm_i}{B_i} \cdot \frac{1-(\frac{r_i}{B_i})^{n_i+1}}{1-\frac{r_i}{B_i}} + T_{down\_i} \quad . \tag{3}$$

Communication Cost. The communication cost is mainly related to the distance and traffic between the migrated VM and other VMs. The communication cost of VM_i is represented by (4).

$$Cost\_com(VM_i) = \sum_{j \neq i} D(PM_{i,target}, PM_{j,src}) \cdot f(VM_i, VM_j) \quad . \tag{4}$$

VM Heat. The VM heat represents the strength of the VM to handle tasks. We use the resource utilization of the VM to reflect its heat. The resource utilization of the VM varies with the dynamic application load. When it is necessary to migrate VMs for an overloaded PM, some VMs may be dealing with high-intensity tasks, and their resource utilization is likely to be high. The calculation of the VM heat depends not only on the resource utilization at the migration moment, but also on historical data. Higher historical utilization represents that the VM is generally dealing with a lot of tasks, and it is also likely that the tasks will be intense in the future. Therefore, we use (7) to calculate the VM heat of the VM group.

$$H(VM_i) = (H(CPU)_i + H(RAM)_i)) / 2 \quad . \tag{5}$$

$$H(CPU)_i = \lambda \cdot AVG(\sum_{j \in T} U\_CPU_j) + (1-\lambda) \cdot U\_CPU_t, H(RAM)_i = \lambda \cdot AVG(\sum_{j \in T} U\_RAM_j) + (1-\lambda) \cdot U\_RAM_t \cdot \quad (6)$$

$$H(VMgroup) = AVG(\sum_{VM_i \in VMgroup} H(VM_i)) \quad . \quad (7)$$

In (6), U_CPU_j and U_RAM_j represent CPU utilization and RAM utilization at time j. t represents the migration moment. T is the total duration of historical data. We use the average of the utilization within T before t as the VM's historical resource utilization. Historical data is obtained by sampling. A sampling was conducted at each Δt interval during the T. The data at t should be given greater weight, so that we can calculate the VM heat more accurately. So we set $\lambda = 0.3$. In the experiments, we set T to one hour, and Δt is set to 30 seconds.

3 Algorithm

3.1 VM Group Selection

In order to avoid frequent migration, it is necessary to set the resource safe range (SR). When the resource occupied after the migration is in the SR, it represents the end of the migration on this PM. First, we will select all appropriate VM groups as migration options. The selected VM groups should include all possible scenarios to prevent the loss of the best solution, and the size of each group can't be large. So, traverse the VM associated graph on the overloaded PM, select all VM groups that make the occupied resources of the PM after the migration are in the SR and the DoC of the remaining VMs reaches a certain value. There can be a single VM or multiple VMs in the selected VM group. The 2 to 9 lines of Algorithm 1 show the selection of VM groups.

Algorithm 1. VMMAGS

1. Get $G_k(V_k, E_k)$ on PM_k from $G(V, E)$;
2. **for** *Binary_set* in *all_set*
3. *subG, remG is* the adjacency table of selected VMs, remaining VMs;
4. **if** (*checkAvailable* (G_k, *low, high*, PM_i_CPU, PM_i_RAM) && *checkConnectNum* (G_k, *remG*, θ))
5. All selected VMs *make up* $VMgroup_i$;
6. < $Cost\_norm\_VMgroup_i$, VM_{mig}, PM_{dist} >= CCMS($VMgroup_i$, *subG*, *PMList*);
7. **end for**;
8. **if** no VM group meets the conditions
9. Make each VM on PM_k as the selected VM group and calculate its <$Cost\_norm\_VMgroup_i$, VM_{mig}, PM_{dist} >;
10. *min* = *min* (*Cost_norm_VMgroup*);
11. Calculate σ of all *Cost_norm_VMgroup* ;
12. Get *VMgroupList* with *Cost_norm* in [*min*, *min*+ σ];
13. Calculate $H(VMgroup_j)$ of $VMgroup_j$ in *VMgroupList* ;
14. Get *min* (*Cost_ integrated*);
15. **return** VM migration scheme < VM_{mig}, PM_{dist} >;

A binary string *Binary_set* with the same length as the number of VMs on PM_k reflects the selected state of the VM. 1 indicates that the VM is selected, 0 is the opposite. *checkAvailable*() is used to determine whether the occupied resources of the PM after the migration are within the SR. $\theta(0 \le \theta \le 1)$ represents the value of the DoC that needs to be reached. *checkConnectNum*() is used to check whether the number of connected VMs has reached θ times the total number of remaining VMs. If both conditions are satisfied, the VM group consisting of the selected VMs is used for the next step. If no VM group meets the conditions, make each VM on PM_k as the selected VM group. Optional VM groups on PM_1 in the VM model are circled in Fig.2. The total resource of PM_1 is (16-core, 16000M). The SR is [0.5, 0.6], and θ is 0.5.

3.2 Objective Functions Integration

It is difficult to find the best migration scheme to meet these three goals. But if we integrate the three goals, the difficulty will be significantly reduced.

We define the *Cost_mig* and *Cost_com* weighted sum as the total cost. The simple weighted summation is susceptible to the larger value, so *Cost_mig* and *Cost_com* need to be normalized to eliminate the difference in magnitude. For VM_i on PM_k, we use the max-min method to normalize its cost.

$$Cost\_mig\_norm(VM_i) = \frac{Cost\_mig(VM_i) - \min(Cost\_mig)}{\max(Cost\_mig) - \min(Cost\_mig)}, \max(Cost\_mig) = \frac{\max(vm)}{\min(B)} \cdot \frac{1-(\frac{\max(r)}{\min(B)})^{n+1}}{1-\frac{\max(r)}{\min(B)}} \quad . \quad (8)$$

$$Cost\_com\_norm(VM_i) = \frac{Cost\_com(VM_i) - \min(Cost\_com)}{\max(Cost\_com) - \min(Cost\_com)}, \max(Cost\_com) = \max(degree)\max(D)\cdot\max(f) \quad . (9)$$

Cost_mig is normalized by (8) to obtain *Cost_mig_norm*. max(*vm*), max(*B*) and max(*r*) represent the maximum RAM of the VM on PM_k, the maximum bandwidth of the data center and the maximum dirty page rate. *Cost_com* is normalized in the same way, using (9) to obtain *Cost_com_norm*. max(*D*) represents the maximum distance between PMs. max(f) represents the maximum traffic between VMs on PM_k. max(*degree*) represents the maximum degree of VMs on PM_k. The calculation method of min is opposite to that of max.

Cost_norm is calculated using (10), where $\alpha + \beta = 1$, and we will determine their values through experiments.

$$Cost\_norm(VM_i) = \alpha \cdot Cost\_mig\_norm(VM_i) + \beta \cdot Cost\_com\_norm(VM_i) \; . \; (10)$$

Next we will integrate *Cost_norm* and the VM heat. The implementation of various migration schemes will result in different *Cost_norm*. The cost of many schemes may have only a small difference, but the heat of VM groups in these schemes may be quite different. It is unreasonable to sacrifice the service of VMs in exchange for the small *Cost_norm* difference. The 10 to 14 lines of Algorithm 1 show the specific steps to get the best migration scheme. σ represents the standard deviation of all VM groups and the integration cost *Cost_integrated* is calculated as:

$$Cost\_integrated(VMgroup_j) = \gamma \cdot Cost\_norm(VMgroup_j) + (1-\gamma) \cdot H(VMgroup_j) \quad (11)$$

γ controls the weight of *Cost_ norm*, $\gamma \in [0,1]$. Calculate the minimum value of *Cost_integrated*, and the corresponding migration scheme is the best solution.

3.3 VM Migration Algorithm

In this section, we use the greedy strategy to determine the optimal migration scheme based on selected VM groups.

For the selected VM group, we can't guarantee that the cost of migrating them to the same target PM is less than the cost of individual migration. Moreover, a VM group has multiple partitions. All partitions of $VMgroup_2(VM_3, VM_4, VM_5)$ on PM_1 in Fig.2 are as follows: $partition_1 = \{\{VM_3\},\{VM_4\},\{VM_5\}\}$, $partition_2 = \{\{VM_3, VM_4\},\{VM_5\}\}$ $partition_3 = \{\{VM_3, VM_5\},\{VM_4\}\}$, $partition_4 = \{\{VM_4, VM_5\},\{VM_3\}\}$, $partition_5 = \{\{VM_3, VM_4, VM_5\}\}$

Therefore, we should calculate all partitions of the VM group to get the best solution. Multiple VM collections will be generated in one partition. In order to guarantee a lower communication cost, it is necessary to require that the VMs in the same collection are connected, and they are migrated to the same PM. It means that $partition_4$ does not meet the condition. Algorithm 2 gives the specific steps to calculate the *Cost_ norm* value and the migration scheme of $VMgroup_i$. *availableResource*() is used to determine whether the resource exceeds the upper limit of the SR after the PM adds the migrated VM. *checkConnected*() is used to determine whether the VMs in the collections are connected. It should be noted that the placement conditions of the collection need to meet the resource requirements of all VMs in the collection. Calculate the minimum value of *Cost_ norm* for all partitions as the *Cost_ norm* value of this VM group.

Algorithm2. CCMS(Calculate Cost and Migration Scheme)
Input: $VMgroup_i$, $subG$, $PMList$
Output: $<Cost\_norm\_VMgroup_i, VM_{mig}, PM_{dist}>$

```
1.  for partition_k of VMgroup_i
2.    for( PM_j in PMList)
3.      if (availableResource ( partition_k, PM_j ) && checkConnected ( partition_k, subG ))
4.          Get the minimum Cost_ norm and its < VM_mig , PM_dist >;
5.      endfor;
6.  endfor;
7.  return <Cost_norm_VMgroup_i , VM_mig , PM_dist >;
```

The complete VM migration algorithm based on group selection (VMMAGS) is shown in Algorithm 1. For $VMgroup_i$ that satisfies the selection conditions, CCMS is used to calculate its $<Cost\_norm\_VMgroup_i, VM_{mig}, PM_{dist}>$. After obtaining *Cost_ norm* of all groups, calculate their *Cost_ integrated* according to the integration method mentioned in Sect.3.2. Finally, we get the minimum value of *Cost_ integrated* and the best VM migration scheme.

4 Experiments and Results

4.1 Experimental Setup

We use CloudSim [12] to carry out experimental tests in this section to verity the performance of VMMAGS. The performance of VMMAGS is evaluated by comparing with the algorithm AppAware [5] and TAVMS [8] in terms of migration cost, communication cost and response time. AppAware takes the single VM as the migration object, and uses the greedy strategy to find the migration scheme with the minimum communication cost. TAVMS solves the problem of multiple VMs migration and migrates the VM group as a whole. However, we find that the objectives of them are different from ours. For achieving fair comparison, we modify these two algorithms by replacing the objectives of them with *Cost_norm* defined in this paper.

In Fat-tree topology [13], the parameter k defines the data center size. We use three common structures in real cloud data centers for experiments. Structure1: k=12, there are 432 PMs and 156 switches; Structure2: k=14, there are 686 PMs and 210 switches; Structure3: k=16, there are 1024 PMs and 272 switches. The link capacities in Fat-tree are set ranging from 1GBps to 10GBps. The distance between PMs is computed as shown in [14]. In addition, we model four instances of PMs with different capacity in the simulations, as shown in Table 1. Each PM belongs to one of the four instances, with each instance having probability 1/4. Each VM has CPU requirement of 1, 2, 4 or 8 cores and memory requirement of 1 to 16GB, which is generated randomly from discrete uniformly distributions. We use FCFS algorithm for VM placement. Each VM runs a web-application with variable workload to generate different resource utilization, thus reflecting the different heat of the VM. The traffic between VMs is set according to what is suggested in [15]. If there is flow between VMs, a Gaussian distribution is used to generate the transmission rate. The mean is 10MBps. The standard deviation is 1MBps, and the probability is 0.75. In our experiments, the page dirty rate is set to 100MBps. vm_{th} is set to 100MB, which is a reasonable compromise based on other parameters, and T_{resume_i} is set to 20ms.

Table 1. Configuration information of PMs.

Configuration	CPU cores	RAM(GB)	MIPS
PM Instance 1	16	32	3000
PM Instance 2	16	16	2800
PM Instance 3	8	16	2500
PM Instance 4	4	8	2100

4.2 Parameters Analysis

VMMAGS involves some parameters, and different parameter settings will directly affect results. So we first experimentally analyze the best value of different parameters.

Two important parameters that affect VM group selection are the SR [*low*, *high*] and the DoC of the remaining VMs θ. Besides, these two parameters directly affect the total migration cost and communication cost of the data center. In order to control the

number of VMs that need to be migrated, we set the minimum value of *low* to 0.5. We compare the total migration cost of the different SRs and the communication cost corresponding to different θ values in Structure3 with 2400 VMs to get their best values. The results are shown in Fig.3 and Fig.4.

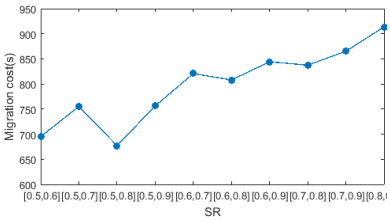


Fig. 3. The total migration cost of different SRs in Structure3

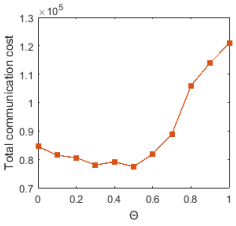


Fig. 4. The total communication cost of different θ in Structure3.

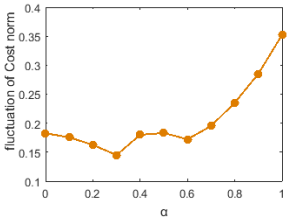


Fig. 5. The fluctuation of *Cost_norm* under different α .

It can be seen from Fig.3, when *high* becomes larger, the migration cost increases. With the expansion of the SR, that is, the gap between *high* and *low* becomes larger, the migration cost decreases. This is because with the expansion of the SR, the optional VM groups increased, so easier to get the best migration scheme. When the SR is [0.5, 0.8], the migration cost is minimal, so we set SR to [0.5, 0.8].

In Fig.4, when θ changes from 0 to 0.3, the communication cost is gradually reduced. This is because when the required DoC is low, it is likely to cause the selected VM group is not the best choice, producing more communication cost than migrating a single VM. When θ is in [0.3, 0.5], the corresponding communication cost is minimal and changes little. Then as θ becomes larger, the communication cost increases significantly. Taking into account the stability of the algorithm and the calculation time, we finally set θ to 0.4.

The calculation of *Cost_norm* involves the weight parameter α . A better weight parameter can guarantee the stability of the algorithm, so the effect on the system performance is reduced to the minimum. For all overloaded PMs in Structure1 with 800 VMs, we experimentally compared the average fluctuation of *Cost_norm* under different α settings. The fluctuation is the difference between the maximum and the minimum values of *Cost_norm*. As shown in Fig.5, the performance of the algorithm will fluctuate with the change of α . When α = 1, the fluctuation of *Cost_norm* reaches the maximum. When α =0.3, the performance of the algorithm is stable, and the value of *Cost_norm* floats in a small area. Therefore, the α value is set to 0.3 in the following experiments with considering the migration performance of the algorithm.

Next we determine the optimal value of γ in (11) to get *Cost_integrated*. We choose the overloaded PM that hosts the most VMs in Structure1 to carry out the experiment, denoted by PM_k. There are 148 selected VM groups. Fig.6 shows *Cost_norm* of all groups, *Cost_norm* in [min(*Cost_norm*), min(*Cost_norm*)+σ] and VM heat. There are four groups with *Cost_norm* in the range. We have experimentally proved that when the value of γ changes from 0.1 to 0.9, *Cost_ integrated* of $group_{95}$ in Fig.6 is always the minimum. Without loss of generality, we set γ to 0.5 in the following experiments.

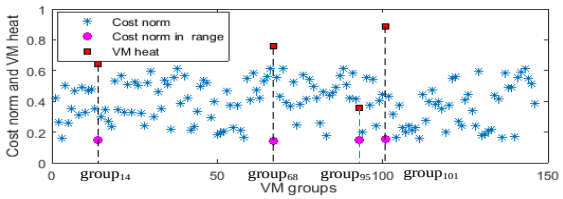
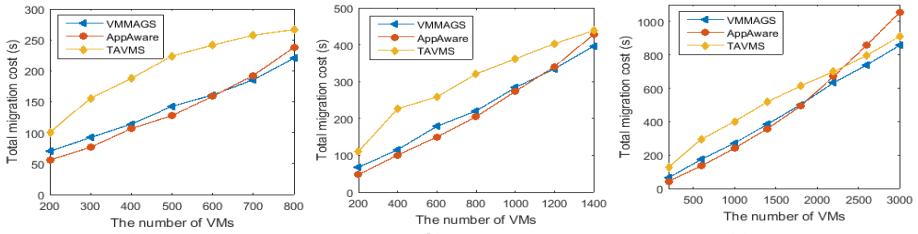


Fig. 6. *Cost_norm* of all selected VM groups on PM_k, *Cost_norm* in [min(*Cost_norm*), min(*Cost_norm*)+ σ] and the VM heat of the group.

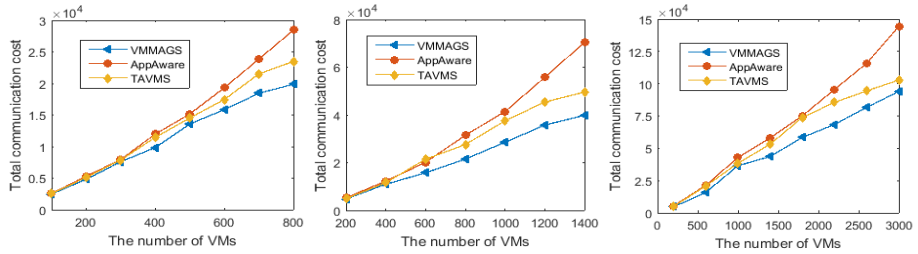
4.3 Results Analysis

Total migration cost. We compare the total migration cost of our proposed VMMAGS with that of the other two algorithms, with the variation of VMs in three structures. The results are shown in Fig.7.



(a) Structure1 (b) Structure2 (c) Structure3

Fig. 7. The total migration cost of all VMs in three structures



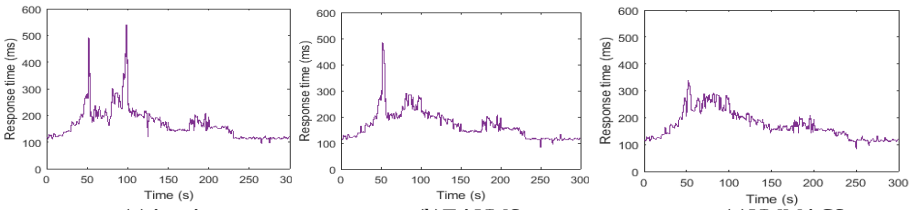
(a) Structure1 (b) Structure2 (c) Structure3

Fig. 8. The total communication cost of all VMs in three structures

It can be seen from Fig.7, our VMMAGS and AppAware performance is relatively similar, and TAVMS is the worst. That is because TAVMS migrates the entire VM group, resulting in a larger memory migration. When the number of VMs is small, the migration cost of AppAware is lower. But we find a rule from the results, that is, when the number of VMs in the data center increased to a certain extent, the migration cost of AppAware exceeds VMMAGS, even more than TAVMS. This is because when there is a large amount of overloaded PMs in the data center, individual-based migration is prone to ineffective migration, resulting in more frequent migration of

VMs, and the migration cost will exceed the group-based migration strategy. In these three structures, the total migration cost of VMMAGS is about 27.4% less than that of TAVMS. Besides, when the number of VMs is large, the total migration cost of VMMAGS is about 18.8% less than that of AppAware. Overall, our VMMAGS performance is more stable, and can effectively control the migration cost.

Total communication cost: The communication cost is another important metric to evaluate the performance of VM migration. So we compare the total communication cost of the three algorithms with the variation of VMs in three structures. In Fig.8, we can observe that our VMMAGS consumes less communication cost than other algorithms in all cases. With the increase of the number of VMs, the total communication cost of VMMAGS increases almost linearly, but the cost of AppAware increases significantly. That is because as VMs become more, individual-based strategy can't get the optimal solution, resulting in the associated VMs migrated to different PMs, so that the increase of the communication cost. When there are enough VMs in the data center, the total communication cost of VMMAGS is about 14.5% less than that of TAVMS, about 36.2% less than that of AppAware.



(a)AppAware (b)TAVMS (c)VMMAGS

Fig. 9. The response time of the PM in Structure1.

Response time: the VM heat directly affects the system response time, so we observe the changes in the response time of a PM using different algorithms in Structure1. Fig.9 depicts the results. When t=50s, the response time surged, indicating that the PM resources were tight. At this point, a migration occurred. As we can see from Fig.9 (a), the PM carried out two migrations, and the response time fluctuated significantly. In Fig.9 (b), the response time had been significantly reduced with TAVMS for migration. But the response time fluctuated greatly during migration. While using VMMAGS, the response time was relatively stable, and could be maintained within 300ms. These results show that VMMAGS can effectively guarantee the system service.

5 Conclusions and Future Work

In this paper, we propose a multi-object VM migration algorithm named VMMAGS, which takes into account the migration cost, communication cost and VM heat to optimize the performance of the data center. According to the SR and the DoC of the remaining VMs, the VM groups that satisfy the conditions are obtained as migration options. Get the optimal migration scheme based on the integration cost of all partitions of selected groups. We assess VMMAGS performance using simulation and compare it with AppAware and TAVMS. Experimental results show that the total

migration cost of VMMAGS is about 27.4% less than that of TAVMS, and the total communication cost of VMMAGS is about 36.2% less than that of AppAware. Besides, our algorithm can better control the response time. In the future, we consider the efficient migration of VMs across data centers.

Acknowledgments. This work was partly supported by the NSFC-Guangdong Joint Found(U1501254) and the Co-construction Program with the Beijing Municipal Commission of Education and the Ministry of Science and Technology of China(2012BAH45B01) and National key research and development program（2016YFB0800302） the Director's Project Fund of Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education (Grant No. 2017ZR01) and the Fundamental Research Funds for the Central Universities (BUPT2011RCZJ16, 2014ZD03-03) and China Information Security Special Fund (NDRC).

References

1. Goldberg, R.P.: Survey of virtual machine research. Computer. 7(6), (1974) 34–45.
2. Zhan, Z.H., et al.: Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version. In: IEEE Transactions on Parallel and Distributed Systems, (2016).
3. Chen, Z.G., Zhan, Z.H., et al.: Deadline constrained cloud computing resources scheduling through an ant colony system approach. In: 2015 ICCCRI, Singapore (2015) 112–119.
4. Zhang, J., Ren, F., Lin, C.: Delay guaranteed live migration of virtual machines. In: Proceedings of the IEEE INFOCOM, Toronto (2014) 574–582.
5. Shrivastava, V., Zerfos, P., Lee, K.W., et al.: Application-aware virtual machine migration in data centers. In: Proceedings of the IEEE INFOCOM, Shanghai (2011) 66–70.
6. Huang, D., Gao, Y., Song, F., et al.: Multi-objective virtual machine migration in virtualized data center environments. In: 2013 IEEE ICC, IEEE (2013) 3699–3704.
7. Zhang, X., Shae, Z.Y., Zheng, S., et al.: Virtual machine migration in an over-committed cloud. In: 2012 IEEE NOMS, Hawaii (2012) 196–203.
8. da Silva, R.A. C., da Fonseca, N.L.S.: Energy-aware migration of groups of virtual machines in distributed data centers. In: 2016 IEEE GLOBECOM, Washington (2016) 1–6.
9. Sun, G., Liao, D., Zhao, D., et al.: Live migration for multiple correlated virtual machines in cloud-based data centers. IEEE Transactions on Services Computing (2015).
10. Tao, F., Li, C., Liao, T.W., et al.: BGM-BLA: a new algorithm for dynamic migration of virtual machines in cloud computing. IEEE Transactions on Services Computing, 9(6), (2016) 910–925.
11. Yao, X., Wang, H., Gao, C., et al.: VM Migration Planning in Software-Defined Data Center Networks. In: 2016 IEEE HPCC, Sydney (2016) 765–772.
12. Calheiros, R.N., Ranjan, R., Beloglazov, A., et al.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience. 41(1), (2011) 23–50.
13. Li, Y., Wang, H., Dong, J., et al.: Application Utility-Based Bandwidth Allocation Scheme for Data Center Networks. In: PDCAT, Beijing (2012) 268–273.
14. Meng, X., et al.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: Proceedings of the IEEE INFOCOM, San Diego (2010) 1–9.
15. Biran, O., Corradi, A., Fanelli, M., et al.: A stable network-aware vm placement for cloud systems. In: 2012 IEEE CCGrid, Ottawa (2012) 498–506.