J Supercomput (2018) 74:5846–5864
https://doi.org/10.1007/s11227-017-2104-9

Improving performance by network-aware virtual machine clustering and consolidation

Gangyi Luo[1] · Zhuzhong Qian[1] ·
Mianxiong Dong[2] · Kaoru Ota[2] · Sanglu Lu[1]

Published online: 6 July 2017

Abstract Modern data center consists of thousands of servers, racks and switches. Complicated structure means it requires well-designed algorithms to utilize resources of data centers efficiently. Current virtual machine scheduling algorithms mainly focus on the initial allocation of virtual machines based on the CPU, memory and network bandwidth requirements. However, when tasks finished or lease expired, related virtual machines would be deleted from the system which would generate resource fragments. Such fragments lead to unbalanced resource utilization and decline of communication performance. This paper investigates the network influence on typical applications in data centers and proposed a self-adaptive network-aware virtual machine clustering and consolidation algorithm to maintain an optimal system-wide status. Our consolidation algorithm periodically checks whether consolidation is necessary and then clusters and consolidates virtual machines to lower communication cost with an online heuristic. We used two benchmarks in a real environment to examine network

✉ Zhuzhong Qian
qzz@nju.edu.cn

Gangyi Luo
luogangyi@dislab.nju.edu.cn

Mianxiong Dong
mx.dong@csse.muroran-it.ac.jp

Kaoru Ota
ota@csse.muroran-it.ac.jp

Sanglu Lu
sanglu@nju.edu.cn

1 State Key Laboratory for Novel Software Technology, Nanjing University, 163 Xianlin Avenue, Nanjing 210023, China

2 Muroran Institute of Technology, 27-1 Mizumoto-cho, Muroran, Hokkaido 050-8585, Japan

influence on different tasks. To evaluate the advantages of the proposed algorithm, we also built a cloud computing testbed. Real workload trace-driven simulations and testbed-based experiments showed that, our algorithm greatly shortened the average finish time of map-reduce tasks and reduced time delay of web applications. Simulation results showed that our algorithm considerably reduced the amount of high-delay jobs, lowered the average traffic passed through aggregate switches and improved the communication ability among virtual machines.

Keywords Data center · Network aware · Virtual machine · Clustering · Consolidation

1 Introduction

Cloud computing is changing the way in which people use computing resources. Recent research shows more than 93% organizations are using or preparing to use Infrastructure as a Service (IaaS). With the help of Cloud, people no longer need to buy and maintain their own physical devices. Instead, they can lease computing resources from Cloud providers to serve their needs. More and more people choose to put their job into Cloud which leads to the expansion of data centers' scale. Therefore, the amount of virtual machines in data center rises quickly. Increasing demand for cloud computing resources and the expanding of data centers' scale pose a challenge to efficiently managing users' request and physical resources.

Traditional researches usually focused on improving resources such as CPU and memory utilization by introducing intelligent virtual machine placement algorithms [12,20] and periodically virtual machine consolidation [5]. These studies often turned the problem into multi-dimensional classical bin packing problem which is known to be NP-hard and solved by heuristic algorithms. Recently, more studies paid their attention to improving communication ability of virtual machines since numerous of distributed computing tasks like map-reduce have been deployed into data centers. Such distributed computing tasks involve massive data transfer among virtual machines which require high network bandwidth guarantee.

To guarantee the performance of distributed tasks wit high bandwidth requirements, cloud managers should assign virtual machines with large mutual bandwidth usage to host machines in close proximity. However, though a variety of network-aware virtual machine placement algorithms have been proposed, at the best of our knowledge, none of these researches considered a specific network-aware virtual machine consolidation algorithm. When tasks finished, related virtual machines would be shut down and deleted from the system which generate plenty of resource fragments. Such fragments lead to low resource utilization as well as increase network delay. Furthermore, it is hard to predict communication patterns. Thus, during the running time, some heavy communication traffic may across a long distance. To solve this problem, we designed a self-adaptive network-aware virtual machine clustering and consolidation algorithm which automatically detects resource fragments and high-cost network communication virtual machines and then clusters and consolidates them through appropriate live migrations. The target of our algorithm is to maximize communication ability among virtual machines to improve their performance with relatively low cost. The algorithm has linear time complexity and is suitable for using in large data center. And experiments showed that our algorithm considerably reduced the amount of high-delay tasks, lowered the average traffic passed through aggregate switches and improved the communication ability among virtual machines.

The rest of this paper is organized as follows. Section 2 presents related works. Section 3 introduces the data centers' architecture and analyzes the network influence on typical applications in data centers. Section 4 formulates the virtual machine placement and consolidation problem and proves the complexity. Section 5 explains the proposed self-adaptive network-aware virtual machine clustering and consolidation algorithm. After that, Sect. 6 presents the experimental results in simulation and in our cloud computing testbed, and Sect. 7 gives the conclusion of this paper. In Sect. 8, we introduce the next step of our work.

2 Related works

Meng Wang et al. [17] formulated the virtual machine placement problem into a stochastic bin packing problem and proposed an online packing algorithm. Zhu Jiang et al. [21] studied the problem of virtual machine allocation under the consideration of providing bandwidth guarantees and proposed an online allocation algorithm for tenants with homogeneous bandwidth demand, which aimed to improve the accuracy of existing algorithms. Meng et al. [10] proposed a traffic-aware virtual machine placement algorithm to improve the network scalability which tried to allocate VMs with large mutual bandwidth usage to host machines in close proximity. Alicherry et al. [1] extend the network-aware virtual machine placement problem into distributed Cloud. They divided the problem into two stage, firstly choosing data center and then choosing rack and server, and proposed a two-approximation algorithm. And Steiner et al. [15] considered the same situation as *Alicherry*, but they focused on service level. Jiang et al. [8] studied a joint tenant (e.g., server or virtual machine) placement and routing problem and gave an approximation to minimize traffic costs. Ofer Biran et al. [3] proposed a heuristics network-aware VM placement algorithm which is trying to allocate a placement that not only satisfied the predicted communication demand but was also resilient to demand time variations. Wilson et al. [18] studied influence of bandwidth on web applications in data center and proposed a deadline-aware control protocol to lower flow latency and improve burst tolerance. Kliazovich et al. [9] considered the network-aware virtual machine placement problem in perspective of energy saving. Ming Xia et al. [19] studied the traffic features of virtualized NFs (vNF), and formulated the problem of optimal vNF placement into binary integer programming (BIP), and proposed an alternative efficient heuristic algorithm to solve this problem. Beloglazov et al. [2] studied the virtual machine consolidation problem in OpenStack platform and implemented a program called NEAT to do consolidation. Ferdaus et al. [6] surveyed network-aware virtual machine placement problem and identified the benefits and limitations of the existing techniques. All these works paid their attention to real-time network-aware virtual machine placement when user's requests arrived but ignored the benefit of consolidation.

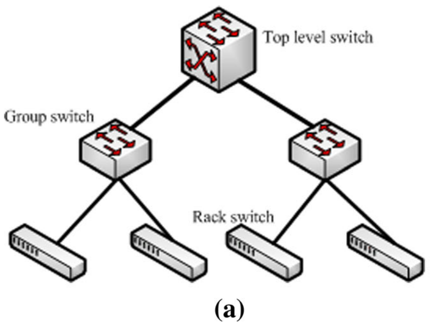

Fig. 1 Network architecture. **a** Abstract network architecture, **b** experimental cloud computing

Breitgand et al. [4] studied the cost of reconfiguring virtual machines in response to workload variations. They observed that live migration requires a significant amount of spare CPU on the source server and if spare CPU is not available, and it impacts both the duration of migration and the performance of the application being migrated. Alexander et al. [14] introduced network topology-aware scheduling models which took workload characteristics, network bandwidth requirements of migrations and network topologies into account. Shrivastava et al. [13] studied the inherent dependencies between VMs and the complex load interactions between the underlying physical server. They introduced an application-aware virtual machine migration algorithm which incorporated inter-VM dependencies and the underlying network topology into virtual machine migration decisions. These works carefully studied the process and cost of live migration of virtual machine which were the basis of our study.

3 Data center architecture and typical application

The most typical network structure in data center is a tree or tree-like topology which is described in Fig. 1a. Figure 1b shows our cloud computing testbed. Commonly in modern data center, each physical server has four or six NICs, and each two NICs are made a bond. These two or three bonds are used for management, virtual machine traffic and storage traffic, respectively. Each bond directly connects to one rack switch, and each rack switch connects to a group switch (aggregate switch). Finally, each group switch connects to top-level switch. Since these three networks have the same structure, we simplified them into a single tree model. In the tree structure, the bandwidth of higher-level switch is shared by lower-level switch which leads to decrease in bandwidth in inter-group communications. Commonly, the bandwidth of two nodes in a same TOR is up to 10Gb/s, while the bandwidth of two nodes in different aggregates switch is lower than 1Gb/s. Unfortunately, two typical types of applications in data centers, distributed computing and three-tier web application, both involve large data transfers which apparently requires high network bandwidth guarantee. Therefore, in this section, we researched on the communication influence on these two applications. We use phrase *tree level* as the meaning of the level of network tree which traffic passed through.

Table 1 MapReduce performance in different network condition

Finish time(s)	Tree level		
	1	2	3
Normal	1145	1262	1457
Overload	1174	1337	1807

Table 2 RUBiS performance in different network condition

Response time(ms)	Tree level		
	1	2	3
Normal	45	46	48
Overload	53	64	92

3.1 Distributed computing scenario

Leasing data center's resources to run temporary distributed computing task is one of most popular applications in cloud computing, and map-reduce is a typical type of distributed computing. Therefore, we choose PUMA,[1] a map-reduce benchmark, to test the network influence on the performance of MapReduce jobs. We did our experiment under two situations, network traffic normal and overload. We compared job's finish time in different ways of virtual machine placement under these two situations.

As Table 1 shows, MapReduce job's average finish time prolongs with tree-level rises. And when traffic is overloaded, the negative influence on the performance of MapReduce jobs becomes more conspicuous.

3.2 Web application scenario

Current web applications normally use three-tier architecture. Considering the performance, the manageability and the robustness, developers would choose to deploy each tier into different physical machines or virtual machines. Therefore, the communication ability between VMs affects the overall performance of web application. RUBiS [2] is an auction site prototype modeled after eBay.com which is used by us to evaluate the performance of application servers (virtual machines).

As Table 2 shows, in normal situation, average response time almost stays same when tree level rises. However, when traffic is overloaded, average response time increases quickly with tree-level rises. The response time in worst case nearly doubled compared with the best case.

[1] Purdue MapReduce Benchmarks Suite http://web.ics.purdue.edu/fahmad/benchmarks.htm.

[2] Rice University Bidding System, http://rubis.ow2.org/.

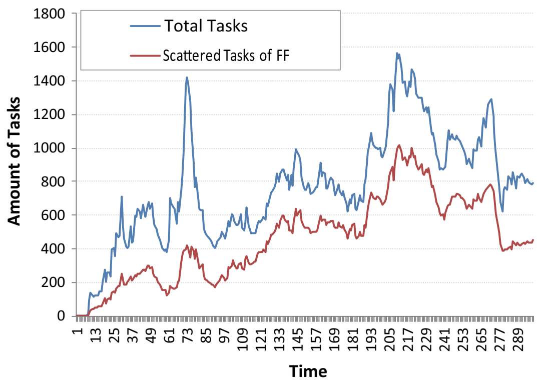

Fig. 2 Rising of *scattered tasks*

4 Virtual machine placement and consolidation

In the previous section, we found that the tree level where inter-virtual machine traffic passed through has a strong impact on the performance of distributed computing and web application tasks. In addition, after a long-time running, some virtual machines would be shut down or deactivated because the job on them has finished which may lead to resource fragments. These fragments will lead to *scattered task* phenomenon (e.g., a task's virtual machines have been allocated into different racks even different rack groups). Simulation of continuously submit and run distributed computing tasks in Fig. 2 shows, as time goes on, the amount of *scattered task* rises.

Therefore, even assumed that we know the traffic model at first, we cannot easily allocate them into same rack due to the fragments problem. Besides, not all users could predict their tasks' traffic model precisely or they just not submit all resource requirement in single time.

4.1 Virtual machine placement

To decrease the amount of *scattered tasks*, firstly, we give the formal definition of the virtual machine placement problem.

Suppose a data center has k_{RG} rack groups, each rack group has k_{R_i} servers. The total amount of physical servers is k_P. We denote each server as p_i and the physical resource which belongs it as a vector $\mathbf{H}_i$. If p_i is running, then $S(i) = 1$, otherwise $S(i) = 0$. $\mathrm{RG}(i) = i_{\mathrm{RG}}$ and $R(i) = i_R$ means that server p_i belongs to i_{RG}th rack group and i_Rth rack in this group. h_{ij} is the min tree level between p_i and p_j which could be computed by $\mathrm{RG}(i)$ and $R(i)$, and we define $h_{ii} = 0$.

Consider the situation that a user u_k initiates a task and this task requests for a group of VMs G_k and the amount of G_k is w_k. Each virtual machine v_i^k in the G_k has its specific requirement for CPU cycles and memory sizes which denotes as vector $\mathbf{R}_i^k$, and $\mathbf{R}_i^k$ could be extended to include other resources such as I/O operations and outlet bandwidth according to specific application. T_k is the $w_k \times w_k$ traffic matric of G_k, and each item t_{ij}^k in T_k is the traffic between v_i^k and v_j^k during time Δt. The traffic matric could either be given by user or be calculated by service provider through the collected data. Notice that our traffic model is quite generic and clearly it could cover both three-tier web application and distributed computing situations. We denote $Z_i^k(m) = 1$ if virtual machine v_i^k is hosted on a physical server p_m, otherwise $Z_i^k(m) = 0$. Therefore, a feasible decision space for virtual machine placement is characterized by

$$\mathcal{Z} = \left\{ \left\{ Z_i^k(m) | Z_i^k(m) \in \{0, 1\}, \sum_{m=1} Z_i^k(m) = 1, \ \forall (i, k); \right. \right.$$
$$\left. \left. \sum_{k=1} \sum_{i=1}^{w_k} Z_i^k(m) \cdot \mathbf{R}_i^k \le \mathbf{H}_i \cdot S(m), \ \forall m \right\} \right\}, \quad (1)$$

where the first equation guarantees that each virtual machine is placed on exactly one host, and the second equation ensures that the total resource consumptions of virtual machines on a physical server should not exceed their host's physical capacity in the condition that the machine is turned on. $HV_Z(v_i^k, v_j^k)$ is the min tree level between v_i^k and v_j^k in the placement Z.

$$HV_Z(v_i^k, v_j^k) \ = \sum_{m=1} \sum_{n=1} \left[h_{mn} \cdot Z_i^k(m) \cdot Z_j^k(n) \right]. \quad (2)$$

4.2 Traffic pattern modeling

Since we have the specific position of each virtual machine and the traffic matrix between virtual machines, we could calculate the total traffics of each switch and then judge whether it is overloaded. We mark all switches in the data center as $\{s_0, s_1 \ldots\}$. $L(m, n) = (b_0, b_1, b_2 \ldots)$, $b_i \in \{0, 1\}$ is a route function between p_m and p_n, $b_i = 1$ means traffic between p_m and p_n would pass through switch s_i, otherwise traffic would not pass through switch s_i. After we defined $L(m, n)$ between p_m and p_n, we could define LM $\left(v_i^k, v_j^k\right)$ as the route function between v_i^k and v_j^k.

$$\mathrm{LM}(v_i^k, v_j^k) \ = \sum_{m=1} \sum_{n=1} \left[L(m, n) \cdot Z_i^k(m) \cdot Z_j^k(n) \right]. \quad (3)$$

Therefore, the total traffic of a task on all switches could be calculated as follows.

$$D(k) \ = \sum_{i=1}^{w_k} \sum_{j=1}^{w_k} \left[\mathrm{LM}(v_i^k, v_j^k) \cdot t_{ij}^k \right]. \quad (4)$$

4.3 Measurement of network benefit

As shown in previous sections, an overloaded switch has a strong negative effect on the performance of virtual machines which have large traffic passing through it. Since we have already got all tasks' traffic and their route, we could get the total traffic of switch s_i,

$$\mathrm{DS}(i) = \sum_{k=1} D(k) \cdot \mathbf{1}_i, \tag{5}$$

where $\mathbf{1}_i$ means a vector where only ith bit equals 1, other bits equal 0. If $\mathrm{DS}(i)/\Delta\ t \geq C(i)$, $C(i)$ is capacity of switch s_i, s_i is overloaded. However, it is obviously too late to reschedule while a switch is overloaded. Therefore, we introduce parameter α, if $\mathrm{DS}(i)/\Delta\ t \geq \alpha \cdot C(i), \alpha \in (0, 1]$ and s_i is not the rack switch, it will trigger consolidation.

Assume at time t_0, the placement of virtual machines was $\left\{X_i^k(m)\right\}$, $\left\{X_i^k(m)\right\} \in \mathcal{Z}$; and after consolidation, the new placement of virtual machines is $\left\{Y_i^k(m)\right\}$,$\left\{Y_i^k(m)\right\} \in \mathcal{Z}$.

Our target is to improve the communication capability among VMs in each group G_k; thus, a practical and effective way is to keep all VMs of a group in a close position (e.g., schedule them into the same server or the same rack). It has three advantages. Firstly, the lower tree-level traffic has to pass through, the higher bandwidth it may shared. Secondly, if traffic only pass through low tree nodes (switches), it could save the network capacity of high tree nodes which means more bandwidth could be shared by lower nodes. Thirdly, as Table 1 shows, overload occurred in lower-level switches has less influence on tasks compared with overload occurs in higher-level switches.

Therefore, we define a $Benefit$ function to estimate network influence quantitatively.

$$\mathrm{Benefit}_Z(G_k) = \sum_{i=1}^{w_k}\sum_{j=1}^{w_k} \frac{t_{ij}^k}{B\left(HV_Z\left(v_i^k, v_j^k\right)\right)}, \tag{6}$$

where $B(HV_Z(v_i^k, v_j^k))$ is a bandwidth function of min tree level. To simplify the computation, we define $B(x) = (N)^x$, N is the bandwidth declining rate. According to Cisco's Data center Infrastructure Design Guide,[3] a typical network design of data centers has a bandwidth declining rate of 2.5:1 to 8:1. Here we use 4:1 as our bandwidth declining rate in experiment, and it should be modified to real bandwidth declining rate on the basis of specific data center architecture.

4.4 Consolidation optimization

Suppose at time t_0, the placement of virtual machines was $\left\{X_i^k(m)\right\}$, $\left\{X_i^k(m)\right\} \in \mathcal{Z}$; and after consolidation, the new placement of virtual machines is $\left\{Y_i^k(m)\right\}$, $\left\{Y_i^k(m)\right\} \in \mathcal{Z}$. We can formulate our network-aware consolidation (NAC) problem as follows:

[3] http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCI_SRND_2_5a_book.html.

$$
\begin{aligned}
&\textbf{NAC(Y)}\\
&\text{Maximize} \quad \textstyle\sum_{i=1}^{K} \; Benefit_Y(G_k)\\
&\text{Subject to} \quad \left\{Y_i^k(m)\right\} \in \mathcal{Z}, \; \forall i, \; k, \; m\\
&\qquad\qquad \text{Benefit}_X(G_k) \leq \text{Benefit}_Y(G_k), \; \forall k
\end{aligned}
$$

The first equation is to optimize overall communication ability of the data center. The following equations guarantee that the new placement would be valid and for each group of virtual machines, it would benefit from consolidation.

5 Consolidation algorithms

Since *Benefit* function is nonlinear and virtual machine placement is a integer programming problem, **NAC** is a nonlinear integer programming problem. Murty et al. [11] proved that nonlinear programming is NP-Hard. Therefore, nonlinear integer programming, as a subproblem of nonlinear programming, is NP-Hard either. Since current data centers usually have more than 10,000 physical servers and 100,000 virtual machines running on them, to get the exact solution of the above NP-Hard problem in such a large scale is extremely time-consuming which is unacceptable in real applications. To solve the above optimization in a feasible time, we design a heuristic algorithm.

Before introducing our algorithm, we give three important observations first.

– Since live migration is a costly operation which will generate large burst traffic, moving all virtual machines of a group to new servers or racks is too expensive.
– Traffic between virtual machines is irregular; therefore, virtual machine pair which generates large traffic should have priority to be consolidated.
– Communication through high-level switch has higher delay and lower bandwidth. Therefore, virtual machine pair which communicates through high-level switches should be rescheduled in priority.

Due to these observations, we defined a priority function,

$$\text{Priority}(G_k) = \sum_{i=1}^{w_k}\sum_{j=1}^{w_k} t_{ij}^k \cdot \left[B\left(HV_X\left(v_i^k, v_j^k\right)\right)\right]. \tag{7}$$

The value of Priority(G_k) reflects the order of urgency. Thus, our algorithm firstly calculates *Priority* value of each virtual machine group and sort them in descending order. Then we select virtual machine group to do our consolidation in order. While processing each virtual machine group, we are trying to move virtual machine pair which has large mutual traffic into the same rack with minimum operations. Since consolidating all virtual machines of a group into the same rack is not always possible, we cluster virtual machines according to their traffic pattern in need .Therefore, we

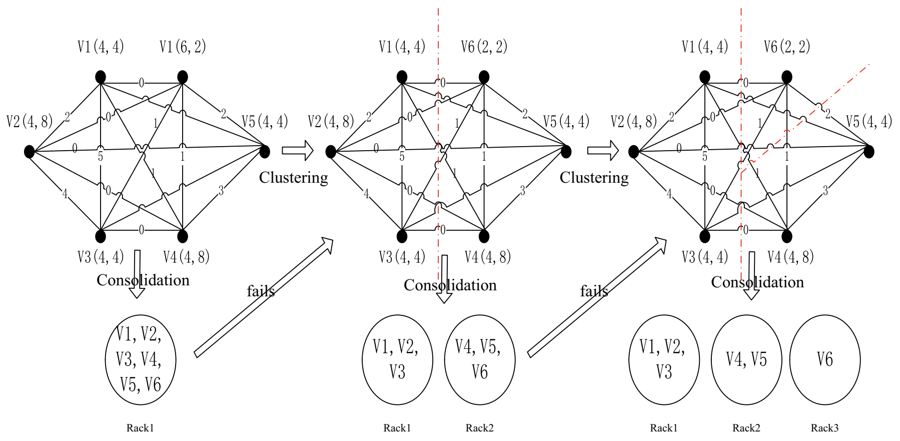

Fig. 3 Process of virtual machine clustering

iteratively cluster and consolidate virtual machines until all the virtual machines in the same clustered group were consolidated into same rack.

5.1 Traffic-based VM clustering

In the condition that we cannot consolidate all virtual machines of a group into the same rack, we cluster virtual machines according to their traffic pattern. The goal of clustering is making the traffic between clustered parts to be minimum. Since a traffic matrix could be mapped to a weighted graph equivalently, clustering a traffic matrix could be converted to cut a weighted graph. The weighted minimum K-cut problem is NP-compete [7] when K is uncertain. When $k = 2$, using *Stoer–Wagner* [16] algorithm could find the optimum solution in polynomial time. Therefore, we perform *Stoer–Wagner* algorithm to split virtual machines of a group into two small groups and then we consolidate the virtual machines of each small group. If consolidation fails, we further split the failed group into smaller group by performing *Stoer–Wagner* algorithm again. The process of the clustering is demonstrated in Fig. 3. The vertexes in the graph represent virtual machines, and the weight on edges represents the traffic between virtual machines. After first time of clustering, the vertexes were split into two sets $\{v1, v2, v3\}$ and $\{v4, v5, v6\}$. The traffic between $\{v1, v2, v3\}$ and $\{v4, v5, v6\}$ is 2 which is minimum traffic in any clustering method. In second time of clustering, $\{v4, v5, v6\}$ was clustered further into $\{v4, v5\}$ and $\{v6\}$.

5.2 Network-aware VM consolidation

For each group of virtual machines, we initiate a queue and push the group into the queue. While the queue is not empty, popping head element of the queue, a set of virtual machines, and trying to consolidate them into the same rack. If consolidation fails, clustering that set of virtual machines into two smaller sets and push them into the queue. In each round of consolidation, we keep the virtual machines in the rack which holds most virtual machines of this group unchanged and consolidate other virtual machine into this rack. Iterating this process until the queue is empty. The processes of clustering and consolidation are demonstrated in Fig. 3. The detailed algorithm is described as follows.

Algorithm 1 Network-Aware Greedy Consolidation

```
Require: $\{\mathbf{X_i^k(m)}\}$
1: Calculate each VM group's $priority$ value, $\{(G_0, pri_0), (G_1, pri_1), \cdots (G_k, pri_k)\}$
2: Sort by $priority$ value in descending order, process VM group from beginning.
3: for Each VM group $G_i$ do
4:   Initiate $Queue$ and push $G_i$ into it.
5:   while $Queue$ is not empty do
6:     Get $G_{head}$ from $Queue$
7:     Partition $G_{head}$ into small parts $\{SG_0, SG_1, \cdots\}$ according to which rack they belong to.
8:     Calculate the amount of VM in $\{SG_0, SG_1, \cdots\}$ and sort them in descending order.
9:     Get the rack $Rack_k$ which holds the largest $SG_k$
10:    if $Rack_k$ have enough space to
       accommodate VMs in $\{SG_0, SG_1, \cdots\} - \{SG_k\}$ then
11:      Migrate $\{SG_0, SG_1, \cdots\} - \{SG_k\}$ into $Rack_k$
12:    else
13:      Cluster $G_{head}$ into $G_{head}^1, G_{head}^2$ and push them into $Queue$
14:    end if
15:  end while
16: end for
17: return $\{\mathbf{Y_i^k(m)}\}$;
```

To avoid burst traffic from live migration which may congest the management network, we keep the migration speed under v_i which satisfies $\text{DS}(i)/\Delta\ t + v_i \cdot L(j,k) \leq C(i)$.

5.3 Self-adaptive mechanism

We adopt two different ways to execute our clustering and consolidation algorithm, regularly and self-adaptively. Regular way means we do our consolidation algorithm periodically. Short period achieves better performance but brings more cost. Large period is more efficient but may miss the best time to do consolidation. Self-adaptive method resorts to a more intelligent way. It checks switches' current status and traffic history and then judges whether consolidation is necessary. Self-adaptive consolidation algorithm is described as follows.

As *Self-Adaptive Trigger* described, we calculated the total traffic of each switch and compared it with their capacity. If the total traffic approaches the capacity, it denotes that the switch is overloaded or will be overloaded in near future. When switch is overloaded, we compare the current traffic of each virtual machine group with their history data. If current traffic conspicuously declines, this virtual machine group may have a strong possibility of suffering insufficient bandwidth. In such condition, consolidation is necessary.

Algorithm 2 Self-Adaptive Trigger

```
Require: Each Switch's Capacity $\{\mathbf{C_0}, \mathbf{C_1}..\}$
1: Initiate each VM group's traffic to $T_k(t_0) = \mathbf{0}$,
2: for Each $\Delta t$ do
3:    Calculate average traffic of each switch $DS(i)/\Delta t$
4:    Calculate each VM group's traffic $T_k(t_j)$.
5:    if $DS(i)/\Delta t \geq \alpha \cdot C_i$ then
6:       Calculate average traffic of each VM group relates to this Switch $i$ in history
7:       if Average traffic in history $\leq$ current traffic multiply $\beta$ then
8:          Trigger Consolidation
9:       end if
10:   end if
11: end for
```

Notice that in practice, we choose at least 600s as the value of Δt due to the irregularly fluctuated traffic. And we ignored the migration traffic since such traffic goes on management network which is isolated from virtual machine traffic network.

6 Experiments

To evaluate the effectiveness of the proposed approach, we firstly conducted experiments on our experimental data center with small data set. After proving the effectiveness on small data set, we conducted a group of simulation experiments with large data set. Data set was based on the log of real parallel workload.[4] Small data set was derived from a small part of above data set and was modified to fit the scale of our experimental data center. Since this log only includes start time, end time, amount of machines, CPU requirement, memory requirement, we have to generate a random traffic matrix for each task to fit our simulation. And since all the traffic is randomly generated which could describe various scenario, there is no need to distinguish three-tier web application or map-reduce application as discussed before.

6.1 Experiments in a cloud computing testbed

Our experimental cloud computing testbed, see Fig. 1b, contains two rack groups, and each rack group has two five-server racks. All the switches used in experiments are 1Gb/s in specification. All the servers are connected by tree-like network.

As Fig. 4a shows, four lines represent the performance of map-reduce tasks of barely using First-Fit (FF), network-aware [1] and using them along with our consolidation algorithm, respectively. Performance deteriorated quickly when barely using FF algorithm. Using our consolidation algorithm with FF could significantly promote the performance of FF algorithm. Due to constraint of the scale of our experimental environment, our consolidation algorithm didn't bring much improvement to network-aware allocation algorithm. However, in long-time simulation with large data set which

[4] http://www.cs.huji.ac.il/labs/parallel/workload/logs.html.

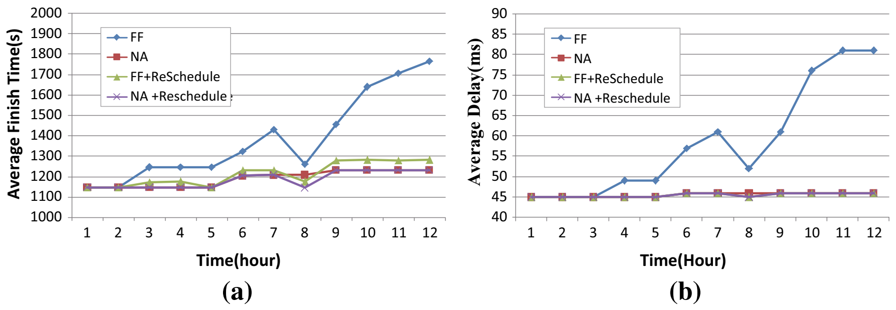

Fig. 4 Experiment results in cloud computing testbed. **a** Effects on distributed computing tasks, **b** effects on web applications

would be shown in next subsection, our clustering and consolidation algorithm brought much improvement to network-aware allocation algorithm either.

Figure 4b shows the result of comparing the performance of web applications in four different algorithms. Same as map-reduce scene, delay increased quickly when barely using FF algorithm. And using our clustering and consolidation algorithm with FF also could achieve good performance. Due to the reason mentioned in previous subsection, our clustering and consolidation algorithm didn't bring much help to network-aware allocation algorithm in this situation.

6.2 Simulation experiments

In this part, we use large data set to conduct our simulation experiments. Since it is hard and inaccurate to compute a task's finish time in simulation, we use the amount of *scattered tasks* and the sum of *benefit* to indicate the performance of tasks. *Scattered task* means that virtual machines which belong to this task has been allocated to different racks. If a task is scattered, its communication ability would be weaken which may lead to decline in performance. *Benefit* function defined in Eq. 6, which indicates the communication ability more accurately. According to the document of Thunder Linux Cluster,[5] we construct a virtual data center which contains 21 rack groups, 269 racks and 3224 physical servers. Workload includes 120, 000 tasks in a month.

Figure 5a shows the comparison of barely using First-Fit algorithm and First-Fit algorithm with clustering and consolidation. We can see that the amount of scattered tasks increase quickly with First-Fit algorithm. When we add our clustering and consolidation algorithm to First-Fit, the amount of scattered tasks decrease significantly. Figure 5d uses *benefit* value as an indicator of network performance of tasks which had the same tendency of Fig. 5a.

Figure 5b shows the comparison of barely using network-aware algorithm[1] and network-aware algorithm with clustering and consolidation. We can see that, although barely using network-aware VM allocation algorithm could achieve good performance

[5] https://computing.llnl.gov/?set=resources&page=index.

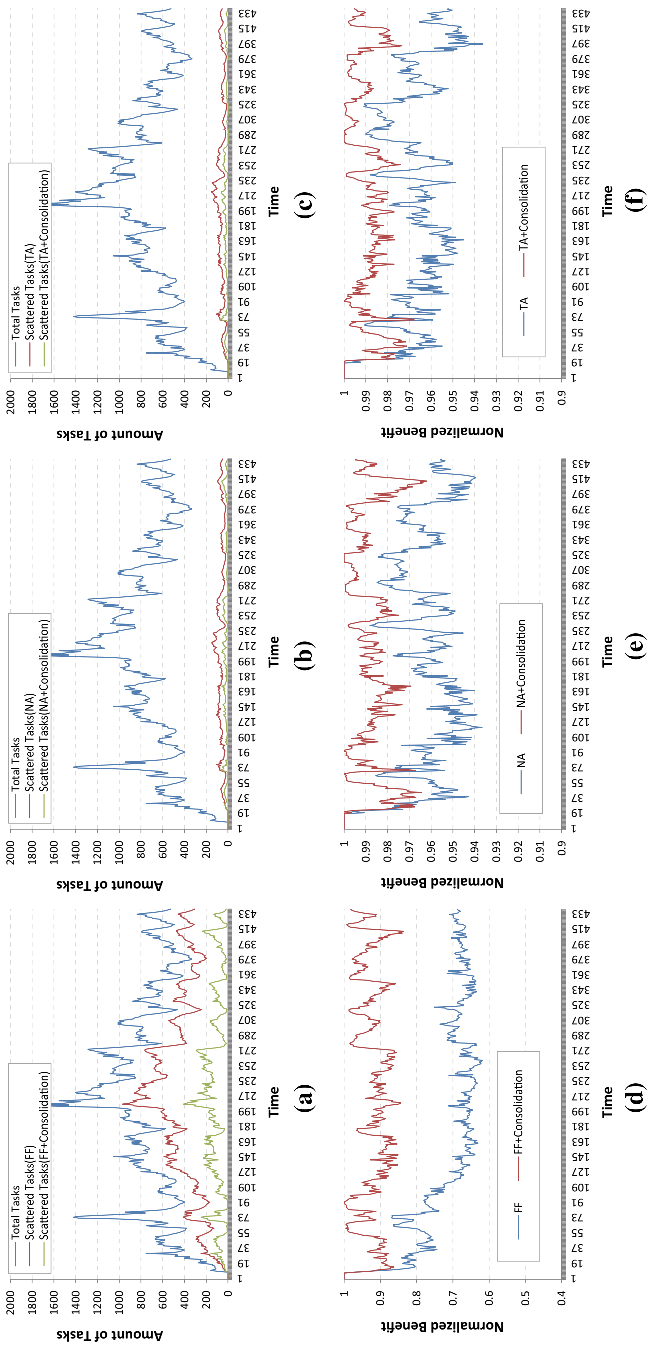

Fig. 5 Results of simulation experiments. **a** Scattered tasks: First-Fit, **b** scattered tasks: network aware, **c** scattered tasks: traffic aware, **d** normalized $Benefit$: First-Fit, **e** normalized $Benefit$: network aware, **f** normalized $Benefit$: traffic aware

in scattered tasks, it could do better when it coordinated with our clustering and consolidation algorithm. Figure 5e uses *benefit* value as an indicator of network performance of tasks. It also shows that clustering and consolidation algorithm could promote performance of original network-aware algorithm.

Figure 5c shows the comparison of barely using traffic-aware algorithm[10] and traffic-aware algorithm with clustering and consolidation. We can see that, traffic-aware algorithm achieved good performance as network-aware algorithm. However, it also achieved better performance when it coordinated with our clustering and consolidation algorithm. Similar as previous results, Fig. 5f uses *benefit* value as an indicator of network performance of tasks and shows that our clustering and consolidation algorithm could promote performance of original traffic-aware algorithm.

Figure 6 shows the average traffic passed through top-level switches. We attached our clustering and consolidation algorithm with First-Fit allocation, network-aware allocation algorithm and traffic-aware allocation algorithm. Results shows that in all situations, using allocation algorithm with our clustering and consolidation algorithm could reduce top-level switch traffic significantly. Compared with First-Fit, network-aware allocation and traffic-aware allocation had much less top-level switch traffic. However, network-aware and traffic-aware allocation in company with our clustering and consolidation algorithm can make top-level switch traffic decreased further more.

Figure 7 shows the average traffic passed through middle-level switches. Due to space constraints, we only displayed the traffic of first four groups. And since the differentiation of traffic between traffic-aware virtual machine algorithm and network-aware algorithm is not significant, we only displayed the result of network-aware algorithm. Similar with top-level switch traffic, results show that no matter we use First-Fit or network-aware allocation algorithm, when we use our clustering and consolidation algorithm to assist them, the middle-level switch traffic would be reduced tremendously. And since live migration renders large traffic in short time, there are some leaps in Fig. 7e–h. And because the total traffic was low, these traffic leaps will not cause side effects.

7 Conclusion

Virtual machine allocation and scheduling are key issues of management in data centers. In this paper, we analyzed the network influence on distributed computing tasks and web applications and proposed a network-aware virtual machine clustering and consolidation algorithm with self-adaptive mechanism. Our algorithm focused on lowering communication cost among virtual machines through conditional and automatic virtual machine live migrations. We used two benchmarks, RUBiS and PUMA, in our experimental cloud computing testbed to examine network influence on tasks. And by using real workload data in both simulation and our experimental testbed, we compared the result of with and without using our algorithm with different virtual machine allocation algorithms. Results in real experiments showed that our algorithm reduced the average finish time of map-reduced tasks and reduce time delay of web applications. Simulation results showed that our algorithm considerably reduced the

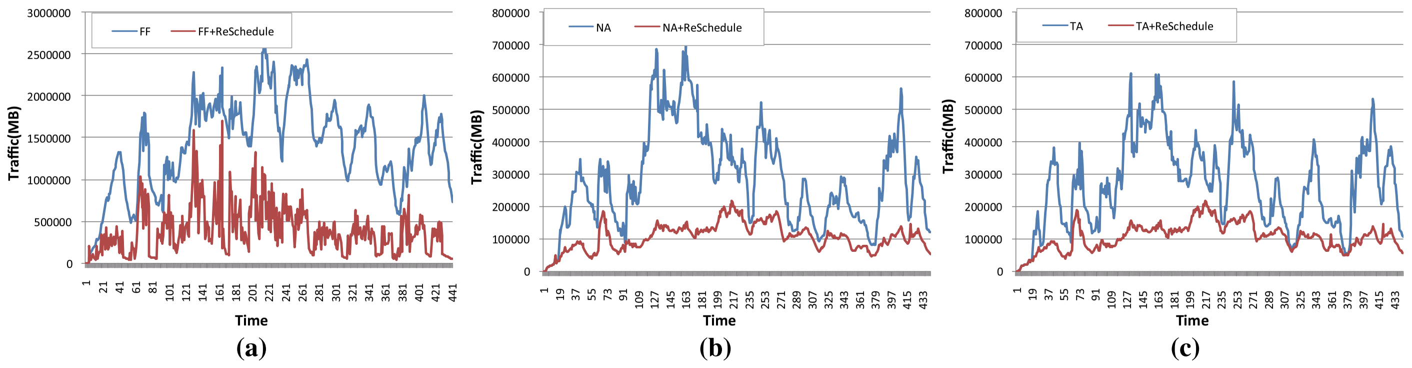

Fig. 6 Top-level switch traffic. **a** First-Fit, **b** network aware, **c** traffic aware

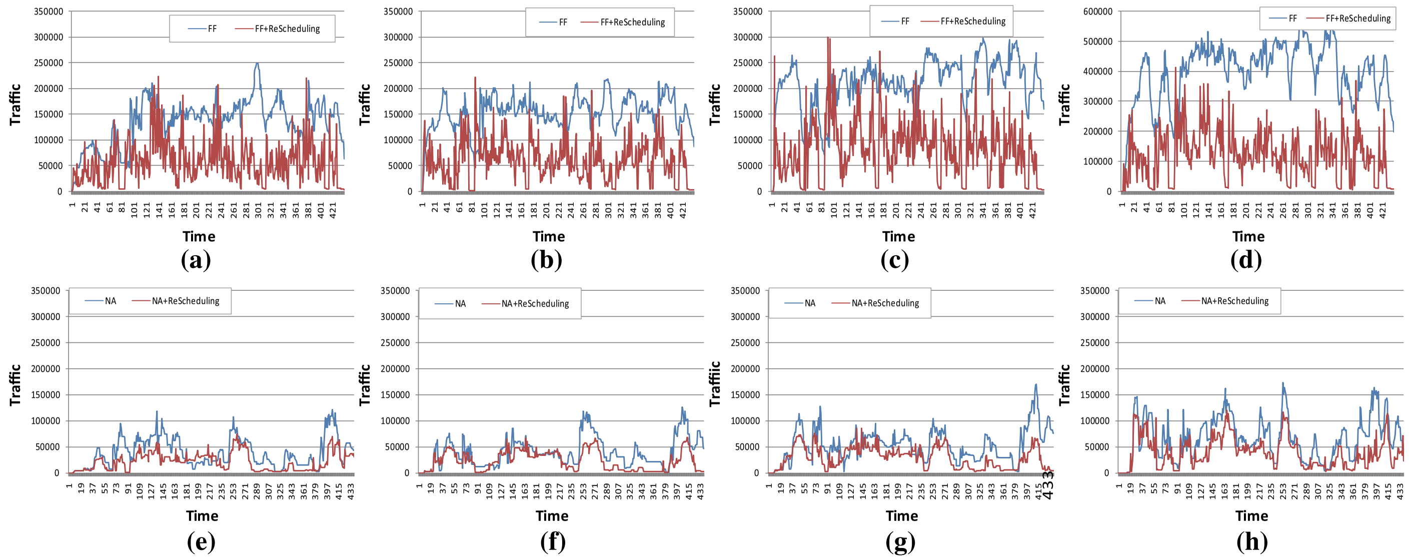

Fig. 7 Middle-level switch traffic. **a** Group0, **b** Group1, **c** Group2, **d** Group3, **e** Group0, **f** Group1, **g** Group2, **h** Group3

amount of high-delay jobs, decreased the average traffic of high-level switches and improved the communication ability among virtual machines.

8 Future work

Software defined network (SDN) is gradually introduced into data centers which makes the network structure becoming more and more complicated. Virtual routers and virtual switches created hundreds of virtual networks on the physical network and change the communication path. Therefore, the next step of our work would focus on how to extend our network-aware consolidation algorithm into a SDN environment.

Acknowledgements The authors want to thank Weicheng Huai, Zhigang Jiang, Kaiyuan Wen and Shen Zhang for their novel suggestions and kind assistance. This work is partially supported by the National Natural Science Foundation of China under Grant No. 61472181, 61321491; Jiangsu Natural Science Foundation under Grant No. BK20151392; JSPS KAKENHI Grant Number JP16K00117, JP15K15976, KDDI Foundation. And this work is also partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

1. Alicherry M, Lakshman TV (2012) Network aware resource allocation in distributed clouds. In: Proceedings of International Conference on Computer Communications, IEEE
2. Beloglazov A, Buyya R (2015) OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds. Concurr Comput Pract Exp 27(5):1310–1333
3. Biran O et al. (2012) A stable network-aware vm placement for cloud systems. In: Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE Computer Society
4. Breitgand D, Kutiel G, Raz D (2010) Cost-aware live migration of services in the cloud. In: Proceedings of the 3rd Annual Haifa Experimental Systems Conference, ACM
5. Dutta S, Verma A (2011) Service deactivation aware placement and defragmentation in enterprise clouds. In: Proceedings of the 7th International Conference on Network and Services Management, International Federation for Information Processing
6. Ferdaus MH, Murshed M, Calheiros RN, et al. (2015) Network-aware virtual machine placement and migration in cloud data centers. In: Emerging research in cloud distributed computing systems, Chap 2, pp 42–91. doi:10.4018/978-1-4666-8213-9.ch002
7. Iqbal W, Dailey MN, Carrera D (2010) Sla-driven dynamic resource management for multi-tier web applications in a cloud. In: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), IEEE
8. Jiang JW et al (2012) Joint VM placement and routing for data center traffic engineering. In: Proceedings of International Conference on Computer Communications, IEEE
9. Kliazovich D, Bouvry P, Khan SU (2013) DENS: data center energy-efficient network-aware scheduling. J Cluster Comput 16(1):65–75
10. Meng X, Pappas V, Zhang L (2010) Improving the scalability of data center networks with traffic-aware virtual machine placement. In: Proceedings of International Conference on Computer Communications, IEEE
11. Murty KG, Kabadi SN (1987) Some NP-complete problems in quadratic and nonlinear programming. Math Program 39(2):117–129
12. Nguyen Van H, Dang Tran F, Menaud JM (2009) Autonomic virtual resource management for service hosting platforms. In: Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing, IEEE
13. Shrivastava V et al. (2011) Application-aware virtual machine migration in data centers. In: Proceedings of International Conference on Computer Communications, IEEE

14. Stage A, Setzer T (2009) Network-aware migration control and scheduling of differentiated virtual machine workloads. In: Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing. IEEE Computer Society
15. Steiner M et al (2012) Network-aware service placement in a distributed cloud environment. J ACM SIGCOMM Comput Commun Rev 42(4):73–74
16. Stoer M, Wagner F (1997) A simple min-cut algorithm. J ACM (JACM) 44(4):585–591
17. Wang M, Meng X, Zhang L (2011) Consolidating virtual machines with dynamic bandwidth demand in data centers. In: Proceedings of International Conference on Computer Communications, IEEE
18. Wilson C et al (2011) Better never than late: meeting deadlines in data center networks. J ACM SIGCOMM Comput Commun Rev 41(4):50–61
19. Xia M, Shirazipour M, Zhang Y, Green H, Takacs A (2015) Network function placement for NFV chaining in packet/optical data centers. J Lightwave Technol 33(8):1565–1570
20. Xu J, Fortes JAB (2010) Multi-objective virtual machine placement in virtualized data center environments. In: IEEE/ACM Int'l Conference on Green Computing and Communications, & Int'l Conference on Cyber, Physical and Social Computing, IEEE
21. Zhu J et al (2012) Towards bandwidth guarantee in multi-tenancy cloud computing networks. In: Proceedings of 20th IEEE International Conference on Network Protocols, IEEE