Fair VM Placement in Geo-Distributed Cloud Systems

Wenlai Yang
Sun Yat-Sen University
School of Information Science and Technology
Guangzhou,China
Email: Wen_lai [AT] hotmail.com

Hong Shen
Sun Yat-Sen University, University of Adelaide
School of Information Science and Technology, School of Computer Science
Guangzhou,China Adelaide,Australia

Abstract—**We consider the resource placement problem, which appropriately places virtual machines (VMs) for a batch of user requests in a distributed cloud where network nodes are far away from each other. In such an environment, the computation nodes need to communicate with each other to adapt to the request partitioning, and a low communication cost will benefit the total run time of user requests. Moreover, we introduced a new measure as the utility computing billing method "Pay-As-You-Go" that has become important in cloud computing. Users will pay bills for their computing service, thus how to guarantee service fairness among their users is an important challenge, although few works consider this as a optimization objective in VMs placement. In this paper, we extend the VMs placement to the multitask situation and show the poor performance of existing methodologies in this environment, which cannot ensure service fairness. We present a novel heuristic algorithm by clustering the nodes to minimize the communication cost, ensure service fairness and conduct simulations to evaluate the performance of our algorithm.**

Keywords-component; VMs placement; cloud; fairness;

I. INTRODUCTION

Virtualization technology enables cloud systems [1] to share computing resources efficiently, steadily and easily. However, efficiency and applicability of service provisions are still the main challenges for cloud management. One of the key tasks of cloud system management is VM placement, which places the VMs requested by the users on appropriate network nodes or physical machines (PMs) and retains the systems in a state of high efficiency, flexibility and availability. Amazon EC2 [2] and Microsoft Windows Azure [15] are two typical cloud computing systems.

In this paper, we focus on the most basic cloud-service model, cloud provision of Infrastructure-as-a-Service (IaaS), that offers VMs executing tasks submitted by users, just like physical machines. As a VM placement example shown in Fig.1, when three user requests submitted by user A and B arrived in a cloud system, each request will be split into a number of subtasks and every subtask will be performed by one VM. Every VM needs to communicate with VMs belonging to the same request during computing, and the system will return the computing results after all the subtasks are accomplished. As we can see, the VMs placed on the same PM should be constrained within the available resources of this PM to ensure a high stability and availability of the system. Also, the datacenters, on which a cloud computing platform runs, are distributed over a wide area network, and considering some computing tasks may span over a set of PMs, such as Map-Reduce tasks and multi-tier web services. This means a poor placement will lead to terrible communication costs and delay the completion time. Therefore, communication costs and traffic demands become two of the key factors which should be taken into consideration when placing VMs.

On the other hand, a cloud provides on-demand computing instances or capacities with a "Pay-As-You-Go" economic model [3] in most cases. This means users will submit their computing requests and pay for the essential computing resources such as CPU, storage, data, etc. Users then occupy these system resources provided by cloud service providers (CSPs) in a proper way and release them after finishing the computing tasks. Hence, the pricing scheme becomes an important bridge between users and providers. The current practice among major cloud providers is to price computing based on virtual-machine hours. For example, Amazon charges $0.095 per virtual-machine hour.

Moreover, the ultimate aim of CSPs is to provide a quality service which is optimized under both resource-centric (utilization, availability, reliability, incentive) and user-centric (response time, budget spent, fairness) constraints, where fairness [4] is an important metric to evaluate the designed cloud systems which is hardly used to estimate the allocation performance. In this paper, the definition of service fairness is in the situation of multi-user placement, that is to say, the money paid by every user and the service they received are aligned. As the pricing model "Pay-As-You-Go" mentioned above, the simple requests with inappropriate placement might have huge communication costs and these users will have to pay a lot of money, even more than the large request users. Obviously, few users are willing to be treated unfairly like that, because the services they get will not match the bills they have to pay. Therefore, it is also significant to investigate a VM placement strategy with a desirable performance in service fairness.

There are several methods reported in the literature for VM placement. Those methods might have good performance in

single request situations and single optimization objectives. However, little work considered the interaction among users in placement that might cause an inappropriate placement This problem is more complex because it must consider the factors of fairness among the users and dynamic network states among the network nodes, such as network topologies and routing schemes. Hence, to solve these problems mentioned above in a reasonable time, we need to investigate an effective algorithm.

II. Related Works

In the studies addressed the network-aware VM placement issue, a lot of works used graph to represent network topology that vertexes represent the network nodes, which VMs can be placed on. The goal of VMs placement is to select the appropriate nodes to place the VMs which can be transformed to sub-graph problem which intends to obtain the sub-graph satisfied all kinds of optimization objectives to place the VMs. [7] considered single request VMs placement for distributed cloud systems as a sub-graph selection problem and the goal is to reduce the inter-datacenter and intra-datacenter traffic as well as to minimize the maximal path length of the sub-graph to improve the application performance, as a long diameter will cause terrible latency. This paper proposed a top-down systematic resource allocation strategy for distributed cloud systems to reduce the communication latency. They extended a greedy algorithm and presented several data-center selection algorithms. They also showed that this problem is NP-hard and developed a 2-approximation algorithm to solve it. On this basis, [8] considered the same scenario mentioned above, and extended the sub-graph selection algorithm to the situation that every user requests with two kinds of resource requirement. Besides, the communication costs and the distance between datacenters were taken into account. [9] considered the situation that given the fixed location of the data sets, the paper considered minimizing the total access time, maximum access time and a combination of both in two situations that with and without the inter-VM constraint. The threshold technique has been applied which effectively guarantees that the maximum of the solution is limited to an appropriate range. To propose the approximation algorithm for the scenario mentioned above. [10]solved the placement problem of computation nodes to minimize the maximum access latency among all pairs of computation nodes and their responsible data nodes by initially proposing a 3-approximation algorithm which formulates the placement problem as a mixed integer program and relax the constraints and traverse all the edges in a nondecreasing order in the network topology and uses an LP solver to get and LP solution. Li et al. [12] formulated the problem as a variant of the bin packing problem with traffics between each VM, investigated the problem from two aspects and presented an effective binary-search-based algorithm to determine the number of PMs that should be used under a desirable trade-off between communication cost and energy cost.

Although, many improvements are proposed for [7] mentioned above, but little work focused on the scenario that more than one request arrives simultaneously in the cloud. Certainly, we can call these existing methods continuously until all requests get placed. However, some drawbacks can be uncovered when we analyzed deeper: 1) it is obvious that a local optimum cannot ensure the global optimum if we process the requests one by one. 2) this may cause fairness problem since the first few requests which get assigned will take up the high-quality resources and the later requests may need longer time to accomplish. In other words, these later requests will charge their users more since the pricing strategy "pay-as-you-go" mentioned above, even though the sizes of these later requests are smaller than the first few requests. This strategy is unacceptable. It is noteworthy that if we process the requests by increasing order of task size, the simple tasks will get the high-quality resources and finish quickly and more complicated tasks will take longer time relatively. That strategy might look fine at a glance, because the total computing time is proportional to their task sizes. That is, the simpler task must takes less time to accomplish. However, the complex tasks which always cause high communication costs will need much more time to process that may lead to higher total communication costs and degrade the performance of clouds, as they take up low-quality resources. The focus of this paper is extending the VMs placement scheme mentioned in [7] for multi-user placement to minimize the total communication costs without break the fairness among users. We describe the problem in detail in the next section.

III. Problem Description

We consider the scenario of VM placement in a geographically distributed cloud system in which a large number of cloud computing resources are located in distinct sites and each node with different amounts of VMs. As the placement example showed in the Fig.1.

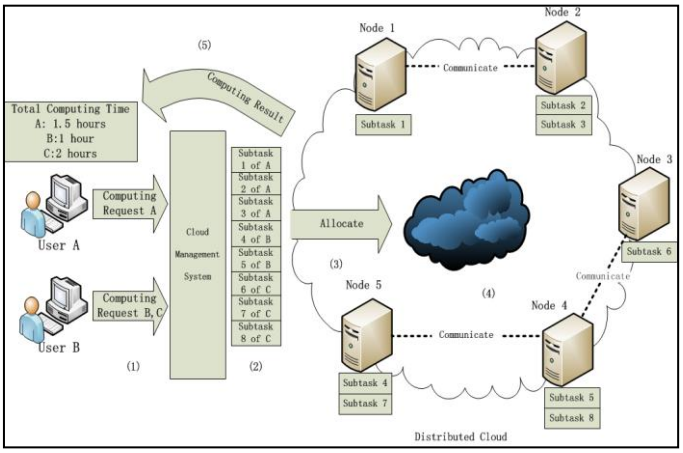

Figure 1. Example of Vms placement

1) Users send their requests to cloud system.

2) Cloud management system split the requests into small subtasks to fit the VM. Since the size of a user request can be very large and we cannot place all the VMs in the same node due to resource constraint of each node. Therefore, a large request will need to be split into many small subtasks and placed on different network nodes. In doing so, the system load

will need to be balanced to achieve a better robustness that would avoid complete collapse of the whole task due to a single node crash.

3) Determine the network nodes that each subtask will be placed and finish assignment. It is important to study whether the placement is fair for all requests. In this paper, we consider the placement is fair if the average consumptions which are the ratios of expenditure of the user to task size are close to each other for every request and we measure system fairness by using Jain's fairness index [13].

4) Computing and communication. The VMs of same request process and exchange their intermediate results. When each VM specified by the same request is placed in distinct nodes, the intra-task data communication is changed to inter-node communication among network nodes which will takes up network resources. That means the diameter of network nodes allocated to the same request should be as small as possible to cut down the communication cost.

5) Return computing results. Cloud systems accomplish the calculation and return the computing results to their users.

In a cloud system, users may not have a priori knowledge of the resource requirements and communication requirements among the VMs. In addition, they are not fixed during the execution. Thus the cloud system need to obtain an approximate estimation based on historical data. As the dynamic characteristics of resource requirement of requests, we apply the existing method [11] in resource-level to achieve fault-tolerant and all the VM can be look as homogeneous in VM-level. In this paper we assume that all the VMs are homogeneous and all the knowledge of the VM requirements are already known. Hence, our goal is to obtain a VMs placement scheme with low communication cost and high system fairness in VM-level.

A. System Model

TABLE I. SYMBOLS

Symbols	Meaning
Input Parameters	
U	The set of user requests
V	The set of nodes in the cloud
E	The set of communication cost in the cloud
u_i	The number of undistributed VMs of request i in U
v_j	The amount of the available VMs in node
$L_{i,j}$	The communication cost between V_i and V_j
VM Allocation	
s_i	Allocation of VMs for request u_i
S	Allocation for all requests
x_i	Average consumption for request U_i
Objective function	
$F_F(S)$	Fairness function
$F_C(S)$	Communication cost function
$F(S)$	Optimization function

Refer to the symbol definitions in Table 1, we consider the graph representation of a cloud system $G(V,E)$. where V enotes the set of network nodes in the cloud system and E represents the edges between all pairs of nodes. We let U denote the set of user requests, and request U_i has the VM resource requirement expressed as u_i, $i \in \{1, \dots, m\}$, which represent the number of VMs are needed to fulfill the tasks. Correspondingly, each network nodes V_j ,$j \in \{1, \dots, n\}$ with weight v_j, $j \in \{1, \dots, n\}$ denote the amount of the VMs it can support. We use edge $e_{i,j}$ between vertices V_i and V_j to denote the network connection between them and the weight on the edge $l_{i,j}$ $i,j \in \{1, \dots, n\}$ represents the communication cost on this edge. For each U_i, our algorithm will give a placement s_i and calculate the average consumption x_i. It is noteworthy that we will not remove the vertices from the graph even if there is no available resource in this node because the diameter of the graph may increase if the vertices in the optimal path are removed.

B. Jain's fairness index

Jain's fairness index is the well-known quantitative fairness measure of how equally a network resource is being shared and we use this index to measure the system fairness. Jain's index is defined as follows x_i:

$$\varnothing = (x_1, x_2, \dots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

Jain's fairness index estimates the discrete degree of x_i. In this paper we borrow the two metrics distance and CPU load mentioned in [14] and refine on two metrics to one metric that each x_i represents the average consumption of request U_i. For the sake of convenience, we use total computing time to represent user expenditure and VM computing time and we character task size using the number of VMs the request took in this paper. As we can see from the formula that the value of this index ranges is within [0, 1]. A value that is close to zero means a low fairness between the users, while a value approaching 1 designates a high fairness.

In the example of Fig.1, request A, B and C are submitted and split into 3,2 and 3 subtasks respectively, every subtask will take up a VM. The the expenditures are 1.5ω , 1ω and 2ω for request A, B and C where ω means the hourly fee of cloud systems. Hence, the average consumption of them is 0.5,0.5 and 0.66 and the system fairness is 98.04%, which means the

placement for request A and B is relatively fair but not absolutely fair for all request because not all ratios are equal.

C. Problem Formulation

Fair Network-Aware VM placement Problem

Given: $G(V,E,w,l)$, vm_i, $i \in \{1,\ldots,n\}$

Compute: $G_i'(V', E', w, l) \subseteq G, i \in \{1, \cdots, n\}$

Obj: G min sum D(G') and Fairness(G')

s.t: $\sum_{w \in G_i'} w \geq wm_i, \forall i \in \{1, \cdots, n\}$

We formulate the Fair Network-Aware VM Placement (FNAVP) problem as follows:

$$\arg\min F(S) = \frac{F_C(S)}{F_F(S)} . \quad (1)$$

$$F_F(S) = \phi(F_T(S)) . \quad (2)$$

$$F_C(S) = \sum_{j=1}^{m} d(s_j) \quad (3)$$

s.t.

$$s_j \subset V, \forall j \in \{1, 2, \cdots, m\}$$
$$\sum_{v_i \in s_j} v_i' \leq v_i, j \in \{1, 2, \cdots, m\} \quad (4)$$

As we purpose to reduce the communication cost $F_C(S)$ and keep a high system fairness $F_F(S)$, our goal is to find a placement S that minimize the objective function $F(S)$ and each element s_i , $i \in \{1, \ldots, m\}$ is a set which includes all vertices that work cooperatively for request U_i. $F_F(S)$ and $F_C(S)$ estimate the fairness and communication cost respectively where $\phi(F_T(S))$ calculate the Jain's fairness for placement where $F_T(S)$ calculate the computing cost and $d(s_j)$ denote the diameter for placement s_j. Equation (4) shows that the sum of resource requirement of all VMs in each node should not break the resource constraint.

IV. ALGORITHM

In this section we consider a set of network nodes to place the VMs where all the VMs resource requirements and communication requirements are already known. FNAVP can be reduced to MINDIAMETER problem mentioned in [5] when the amount of user requests is reduced to 1, that means FNAVP is more complex and also NP-hard and even hard to approximate. Therefore, we view this problem as a variant of MINDIAMETER problem which have been proved to be a NP-hard problem and cannot to be approximated within 2- ε; for any ε; >0. Our goal is to investigate an algorithm which can efficiently reduce the sum of communication cost among network nodes and guarantee the resource allocation fairness concurrently. To guarantee a high fairness, we have to treat each request equally but not place them one by one since the imbalance of network resource. Moreover, we select the nodes that close to each other to place the same request for reducing communication cost. Therefore, we borrow the idea of clustering algorithm and design algorithm: First of all, we identify a subset of the nodes as the cluster centers for every request. Additionally we add the edges and corresponding vertexes to the appropriate cluster one by one and make sure that every VM that has been allocated generate the least communication increase until all requests placed.

Algorithm 1 FNAVMP(G,U)

Input: $G(V,E)$: Graph representation of cloud service

U: the set of resource requirement of current request

Output: S: Initial state

1: Computing the distance matrix by Floyd-Warshall algorithm

2: $S \leftarrow$ *GETSTART*(G,U)

3: **while** ∃ **u≠ 0 do**

4: $V_j' \leftarrow \arg\min_{V_j \in V} \frac{diameter(V_j, S_i)}{\min(v_j, u_i)}$

5: Add V_j' to group S_i

6: **if** ∀ **v=0 then**

7: No available groups exist.Return NULL

8: **end if**

9: **end while**

10: **return** S **and** mindiameter

Algorithm 2 FindInitialCenter(G,U)

Input: G(V,E): Graph representation of cloud service

U: the set of resource requirement

Output: $S_1, S_2, \cdots, S_m$: Initial state

1: Sort in U decreasing order of u

2: **for** $U_i : U_m$ **do**

3: $V_j' \leftarrow \arg\min_{V_j \in V, v_j \neq 0} \sum_{z=1}^{\left\lceil \frac{\min\{(u'-v_j),0\}}{average(v)} \right\rceil} I(V_j)_z$

4: Add V_j' to group S_i

5: **end for**

6: **return** $S_1, S_2, \cdots, S_m$

The Algorithm 1 first call classical Floyd-Warshall algorithm to get the distances between each two nodes and find *m* initial vertexes for *m* requests as cluster centers by invoking FindInitialCenter. Subsequently, we travel all the clusters and vertexes continually to find a vertex with the least ratio of increment of diameter (if we added this vertex to the cluster) to the number of VMs it can supply. That is, each time we add one vertex to one cluster by ensuring the least diameter increasement until all the requirements are met. Since all requests are treated fairly and each placing operation cases the least diameter increasement, the placement schemes will have great performance at fairness and communication cost here.

In order to get an appropriate cluster center set, we select the set with probable minimal diameter by rough estimation of diameter for each placement. Algorithm *FindInitialCenter(G,U)* finds an Initial centers of each request in descending order of request size. For each request, we simply estimate the final diameter for every vertex if we selected this vertex as cluster center and pick up the vertex with the least value. Our estimation approach is summing the smallest edge weights that the number of edges counted is obtained by dividing the amount of unallocated VMs of current request by average number of VMs in all nodes. Function $l(V_i)_j$ returns the jth-smallest edge weight from node V_i.

V. Performance Evaluation

In this section, we evaluate our algorithm through simulations in our two goals: Fairness and communication cost. We conduct three experiments to respectively evaluate the performance of our algorithm in different network scale, resource load rate and request granularity. As it is unknown to us to achieve the optimum, we compare the performance of our algorithm with already existed algorithm mentioned in [7] which achieves 2-approximation to every single request and random placement algorithm. This random algorithm is to select nodes for random request randomly, therefore, it is intuitive that random algorithm can provides a high system fairness since every request is placed equally. We generate random mesh-based topology that every node is at least connect with the other different nodes and the distance is chosen randomly between 5 to 10. To simplify the calculation course, we adapt distance between any two nodes to measure the communication cost, that means the communication cost of one request is the diameter among the nodes it runs on. The final results of all the experiments below are average of 100 runs. In each run, and the total VM requirement make up 80% of the system VM resource.

In the first experiment, we compare the performance of three algorithms mentioned above on different network scale. We design five different cloud network scenarios, each contains 10, 25, 50, 75 and 100 nodes respectively. Number of VMs per node is generated uniformly random between 50 to 100. The VM requests set for each cloud is the same and the sizes are chosen uniformly random between 20 to 100. The simulation results are depicted in Fig. 2(a).

In second experiment, we fix network scale at 50 nodes and change the request number to check the algorithm performance at different resource load rate, each request is generated randomly between 1 to 100. The results are depicted in Fig. 2(b) while other parameters were not changed.

In the third experiment, we focus the different request composition. We assume our heterogeneous system with 50 nodes, the numbers of virtual machines held by each node are generated uniformly random between 1 and 100. Since the average VMs in each node is 50, We conduct experiments average requirement 25, 50, 75, 100, 500 VMs for five sets of requests that are 0.5, 1, 1.5, 2, 5 times of average VMs in each node. Since all the request will occupy 80% system resource, that means there are most requests in the first set and least requests in the fifth set. Fig. 2(c) showed the fairness and total communication cost.

As the simulation results, we have the following observations.

When the number of nodes becomes lager, the communication cost also grows. Our algorithm works well on larger network and simple network and the average performance of our algorithm at communication cost is better than approximate algorithm by 30% and outperforms Random by 70%. The fairness of our algorithm keeps excellent performance with average 0.978, where approximate algorithm is 0.836 and Random 0.951.

When system in a low load state, our algorithm performs as good as approximate algorithm in reducing the communication cost, and shows the better at high load state. This is because the high-quality resources are not fully occupied in low load situation, our algorithm cannot exploit the advantages to the full.

We can see that our algorithm showed excellent performance in all experiment to guarantee the system fairness, even better than random algorithm. In these three experiments, heuristic algorithm is better than approximate algorithm and Random by 11.2% and 3.0% respectively. Because allocation fairness of each operation. It is noteworthy that when request size became lager, the system fairness will be lower for three algorithms. This is because when the request granularity become lager and request number reduction, it is hard to reach completely fair due to the complexity and irregularity of network constraint.

In summary, our algorithm is an efficient strategy for placing VMs in cloud systems, since the high system fairness and excellent performance. Moreover, our algorithm has better performance in handling middle and small requests and also works well in high load state.

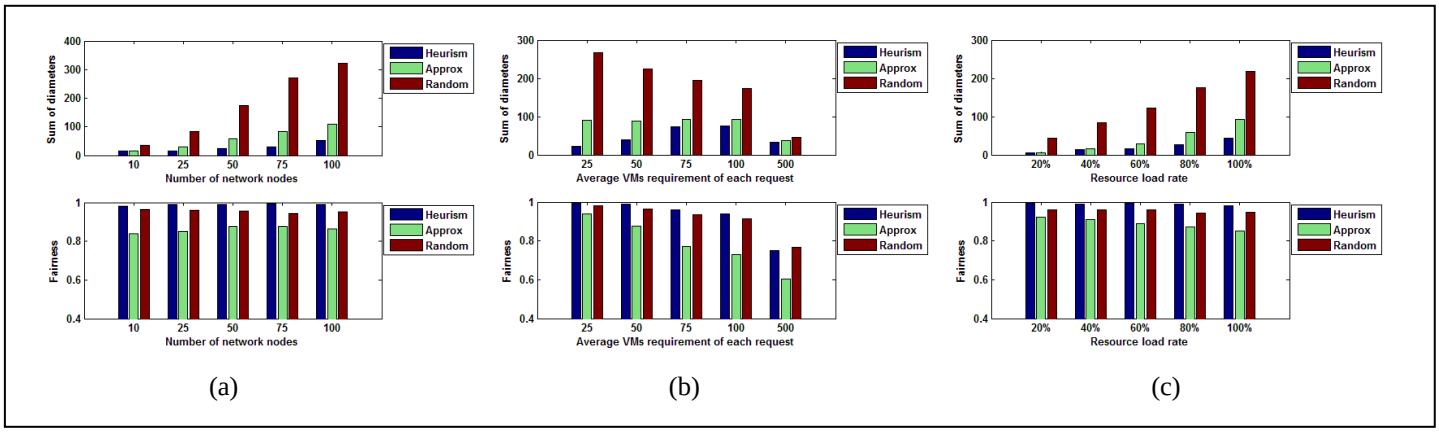

Figure 2. The comparison between the performance of algorithms by 100 times experiment

VI. CONCLUSION

In this paper, we extended the existing VMs placement problem to the multitasking environment and introduced service fairness as an optimization objective. As is the nature of "Pay-As-You-Go", ensuring resource allocation fairness among all user requests is significant for a ripe cloud management system. Optimal placement not only guarantees system fairness but also reduces the total communication cost. Since the problem is NP-hard and even difficult to approximate, an exhaustive search is not feasible. We show the poor performance of existing methodologies and present a heuristic algorithm for placing VMs to network nodes by clustering them in a cost-effective way In the end, we evaluate the performance of our strategy and compare it with the other two strategies.

ACKNOWLEDGMENT

This work is supported by The 985 Project funding of Sun Yat-Sen University, Australian Research Council (ARC) Discovery Project DP150104871 and National Science Foundation of China General Projects funding 6117023. The corresponding author is Hong Shen.

REFERENCES

[1] T OGRAPH B, MORGENS Y R. Cloud computing[J]. Communications of the ACM, 2008, 51(7).

[2] Amazon EC2. http://aws.amazon.com/ec2/.

[3] Dikaiakos M D, Katsaros D, Mehra P, et al. Cloud computing: distributed internet computing for IT and scientific research[J]. Internet Computing, IEEE, 2009, 13(5): 10-13.

[4] Wang H, Jing Q, Chen R, et al. Distributed systems meet economics: pricing in the cloud[C]//Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. USENIX Association, 2010: 6-6.

[5] Alicherry M, Lakshman T V. Network-aware resource allocation in distributed clouds[C]//INFOCOM, 2012 Proceedings IEEE. IEEE, 2012: 963-971.

[6] Yao Y, Cao J, Li M. A Network-Aware Virtual Machine Allocation in Cloud Datacenter[M]//Network and Parallel Computing. Springer Berlin Heidelberg, 2013: 71-82.

[7] Alicherry M, Lakshman T V. Optimizing data access latencies in cloud systems by intelligent virtual machine placement[C]//INFOCOM, 2013 Proceedings IEEE. IEEE, 2013: 647-655.

[8] Kuo J J, Yang H H, Tsai M J. Optimal Approximation Algorithm of Virtual Machine Placement for Data Latency Minimization in Cloud Systems[J].

[9] Li X, Wu J, Tang S, et al. Let's Stay Together: Towards Traffic Aware Virtual Machine Placement in Data Centers[C]//Proc. of the 33rd IEEE International Conference on Computer Communications (INFOCOM). 2014.

[10] Jain R, Chiu D M, Hawe W R. A quantitative measure of fairness and discrimination for resource allocation in shared computer system[M]. Hudson, MA: Eastern Research Laboratory, Digital Equipment Corporation, 1984.

[11] Venzano D, Michiardi P. A Measurement Study of Data-Intensive Network Traffic Patterns in a Private Cloud[C]//UCC. 2013: 476-481.

[12] Microsoft. Windows Azure Platform. http://www.microsoft.com/azure/default.mspx

[13] Ben F, Wang Y. Virtual machine resource allocation strategies based on fault-tolerant QoS in cloud computing[J]. Microelectron. Comput, 2013, 30(3): 136-139.