A Network-Aware Virtual Machine Allocation in Cloud Datacenter

Yan Yao, Jian Cao, Minglu Li

A Network-aware Virtual Machine Allocation in Cloud Datacenter

Yan Yao[a] Jian Cao[*b] Minglu Li[c]

[a, b, c] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[a]yaoyan@sjtu.edu.cn, [b]cao-jian@cs.sjtu.edu.cn, [c]ml-li@sjtu.edu.cn

Abstract. In a cloud computing environment, virtual machine allocation is an important task for providing infrastructure services. Generally, the datacenters, on which a cloud computing platform runs, are distributed over a wide area network. Therefore, communication cost should be taken into consideration when allocating VMs across servers of multiple datacenters. A network-aware VM allocation algorithm for cloud is developed. It tries to minimize the communication cost and latency between servers, with the number of VMs, VM configurations and communication bandwidths are satisfied to users. Specifically, a two-dimensional knapsack algorithm is applied to solve this problem. The algorithm is evaluated and compared with other ones through experiments, which shows satisfying results.

Keywords: VM allocation; cloud datacenter; Two dimensional knapsack algorithm.

1 Introduction

Cloud computing has emerged as a new paradigm for hosting and delivering services over the Internet[1]. There are three service models in cloud computing: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) which deliver the infrastructure, platform, and software (application) as services respectively. These services are made available to consumers in an on-demand way. In order to provide users with various cloud services, many cloud providers (e.g. Amazon, Google) have built their cloud datacenters around the world.

Resource allocation is a core process in a datacenter. Recently, an increasing number of cloud providers take advantage of virtualization technologies, such as VMware [3], Xen [4], KVM [5] and OpenVZ [6], to implement a cloud datacenter. Typically, a user submit his requests, including the number of Virtual Machines (VMs for short) and their configurations through a portal of the provider. The provider will allocate the VMs in the cloud datacenters to satisfy the requirements of the user. Thus VM allocation is becoming a new problem to be solved.

* Corresponding Author

There already exist many algorithms for VM allocations with different aims and assumptions [7]. Some of them try to allocate VMs in an energy efficient way[8-11]. Energy consumption is a big concern for cloud providers. But other factors should be considered in VM allocation as well, such as network. Some works have been done on VMs allocation in a network-aware way [12-15]. However, there still exists several open issues to be solved. Firstly, the existing allocation algorithms considering the consumptions of network resources are designed for traditional datacenters whose network architectures are often centralized, not distribute over a wide area. In a cloud data center, the distances between different sub-datacenters greatly affect the performance of applications. In addition, the VMs requested by users have various configurations, such as different number of processors or amount of memory. Thus the issue of the heterogeneity should be considered.

In this paper, we focus on the two open issues mentioned above. As it is well known that the resource allocation problem is NP-hard, here we developed a heuristic algorithm, a Network-aware VM allocation algorithm based on Maximum Clique (MCNVMA for short), with the goal of minimizing the maximum latency in communication between the sub-datacenters. MCNVMA considers constraints on local physical resources, such as CPU and memory, as well as the network. In order to make it more practical, the VMs and the datacenters both are heterogeneous in MCNVMA.

The rest of this paper is organized as follows: Section 2 discusses related work and Section 3 introduces the problem in details. Section 4 illustrates detail steps of the MCNVMA we developed. Experimental results are illustrated in Section 5 to show performance evaluation of MCNVMA. Section 6 concludes the paper.

2 Related Work

Existing work on the allocation of VMs can be categorized into three offering models: reservation model, spot markets model and on-demand access model [17]. In the reservation model, a user purchase a bundle of resources for a period (e.g., a whole year) , during which the specified VMs can get great discount for payment in advance. Spot market model is a one-side auction market, consuming resources at a lower and flexible cost. In on-demand access model, users simply requests a specified number of the VMs, and pays for it according to a fixed schedule of fees. Here we restrict our study to on-demand access model. There are two types of VM allocation decisions to be made: initial placement [18] and optimizing (or migration) of VMs allocations over time [19]. In the current research, initial placement and VM migration as considered as separate topics, though in some cases similar algorithms may be employed. We limit our study to initial VM placement. Cluster and node are two levels considered in initial VM placement. In general, we consider physical machines are put in an unstructured resource pool so that the default is a shared cluster.

Some constraints should be satisfied when VMs are allocated to physical machines,.. Usually, the constraints are put on some specific attributes such as CPU usage, memory usage, and network usage. These constraints can be summarized into a weighted criteria used to order physical machines. If the criteria is about energy, then the algorithm

tends to save energy [8-11]. If the criteria is about network, the algorithm tends to reduce the network traffic[12-15]. We briefly introduce some of them related to our work. In [12], the authors introduce a Traffic-aware VM Placement Problem (TVMPP), trying to reduce the aggregate traffic. As TVMPP is NP-hard , the authors introduce a heuristic approach to solve it. In [13], a Min Cut Ratio-aware VM Placement (MCRVMP) is proposed. It considers both constraints on local physical resources and network resources evolving from complex network topologies and dynamic routing schemas. They achieve them by exploiting the notion of network graph cuts. While both TVMPP and MCRVMP assume static and well-defined traffic demands, in [14] the authors focus on the equivalent capacity notion to consolidate VMs with uncorrelated traffic demands. Traffic demands are modeled as stochastic variables, and the optimization problem strives to place VMs while ensuring that the maximum network capacity is not violated with a particular user-defined probability. Hence, the final VM placement problem is a stochastic bin packing problem, and authors introduce a new heuristic approach to solve it. However, all the above network-aware VM allocation strategy ([12][13][14]) are designed for traditional datacenters whose network architectures are often centralized and do not distribute over a wide area. Mansoor Alicherry et. al in [15] proposed a network aware resource allocation algorithm based on distributed datacenter. They also take it make a compare with traditional datacenter and they regard the VMs as homogeny, which means all the VMs requested by users have the same number of processors and the same amount of memory. It is not applicable to real world. In this paper, we view datacenter are geographical distribution and VM requested by users are unique. The number of processors and the amount of memory of the VMs requested can be arbitrary realistic value.

3 VM Allocation Problem in Cloud Datacenters

What exactly is the VM allocation? In brief, a user requests for a service hosted in the cloud, requiring the allocation of VMs in the cloud datacenters, to meet the requested service's computational needs. The datacenter should identify the suitable physical resources for each requested VMs and allocate them.

A user's request can be specified in terms of the number of VMs, their configurations and the communication requirements. Sometimes a user may not have a priori knowledge of the communication requirements among the VMs. However we can get it by statistic analysis approach. In this paper, we assume that knowledge of the communication requirements among the VMs is already known.

We use a small dataset for ease of illustrating. In reality, it can be much larger sets of VMs and cloud datacenters. Suppose a user applies for ten VMs to run an application over a cloud infrastructure. Each VMs has fixed processor and memory (See Table 1) and the communication requirements(bandwidth) between VMs is collected (See Table 2). In Table 2, if the entry value is 0,it indicates there is no communication between two virtual machines, otherwise, it represents the necessary communication bandwidth needed. Suppose there are only five sub-datacenters to run these VMs. Each sub datacenter has some free CPU and memory capacities (see Table 3), and the distance

between them are known as well(Table 4). Now, we need to find an VM allocation meeting the following requirements:

- Processor and memory requirements of VMs allocated on a sub-datacenter should not exceed its available free capacity.
- The communication requirements among VMs can be satisfied.
- The communication between VMs which belonging to different sub-datacenters (or servers) is minimized.

In this paper, each VM can be placed on arbitrary datacenters and servers. We propose a heuristic approach which consists of two consequent steps: datacenter selection and server selection. Datacenter selection is to select sub datacenters , which are geographically distributed, to place the VMs. And then assign individual VMs to the selected sub datacenters. Server selection is to determine the servers in the selected sub-datacenters and place the individual VMs to the servers. The two steps are very similar in algorithm. And here we just provide an example of selecting sub datacenters.

Table 1. VM Requirements(processor number, memory amount)

VM	Processor Requirement	Memory Requirement
VM_0	4	2
VM_1	8	8
VM_2	2	4
VM_3	8	2
VM_4	4	4
VM_5	4	6
VM_6	2	1
VM_7	9	10
VM_8	8	2
VM_9	6	4

Table 2. Communication cost between VMs（Mbps）

	VM_0	VM_1	VM_2	VM_3	VM_4	VM_5	VM_6	VM_7	VM_8	VM_9
VM_0	0	1	0.1	0.3	0	0.5	0.2	0.3	0	0.4
VM_1	1	0	0.2	0.6	0.05	0.09	0	1	0.6	0.2
VM_2	0.1	0.2	0	0.1	0.1	0.2	0.25	0	0.3	0.1
VM_3	0.3	0.6	0.1	0	0.35	0.18	0.06	0.4	0.72	0
VM_4	0	0.05	0.1	0.35	0	0.1	0.72	0.1	0.1	0.18
VM_5	0.5	0.09	0.2	0.18	0.1	0	0.35	0	0.18	0.1
VM_6	0.2	0	0.25	0.06	0.72	0.35	0	0.06	0.72	0.35
VM_7	0.3	1	0	0.4	0.1	0	0.06	0	0.4	0.5
VM_8	0	0.6	0.3	0.72	0.1	0.18	0.72	0.4	0	0.2
VM_9	0.4	0.2	0.1	0	0.18	0.1	0.35	0.5	0.2	0

Table 3. Datacenters Free Capacities

DC	Processor Capacity	Memory Capacity

DC_0	60	15
DC_1	22	18
DC_2	40	12
DC_3	20	13
DC_4	80	21

Table 4. Distance between DCs (mile)

	DC_0	DC_1	DC_2	DC_3	DC_4
DC_0	0	2	7	11	10
DC_1	2	0	5	7	1
DC_2	7	5	0	9	20
DC_3	11	7	9	0	15
DC_4	10	1	20	15	0

4 VM Allocation Algorithm

4.1 Problem Formulation

Our problem statement can be briefly described as follows:

1. *m* cloud datacenters are available and their resource capacities given along processor and memory dimensions. Noted as the set $DC = \{dc_1, dc_2, \dots, dc_m\}$. For each $dc_i \in DC$, with the capacity $C_{dci} = (Pro_{dci}, Mem_{dci})$;
2. There are *n* VMs to be placed. The requirements of these VMs are given in terms of processors and memories needed, denoted as $VM = \{vm_1, vm_2, \dots, vm_n\}$ and for each $vm_i \in VM$, the capacity requirement is $C_{vmi} = (Pro_{vmi}, Mem_{vmi})$;
3. The communication cost between n VMs is denoted by a matrix $Cost = (cost_{ij})_{n\times n}$;
4. The communication distance between any two sub datacenters is given, denoted as $Dis = (dis_{ij})_{m\times m}$. We regard the network resource as sufficiently enough, however in server selection phase, the constrains on communication distance is changed to the network resource (like bandwidth).
5. We need to find a mapping between VMs and sub datacenters that satisfies the VMs' resource requirements while minimizing inter-datacenter traffic and intra-datacenter traffic between VMs.

While finding such a mapping, we have to take care that the total resource requirement of the VMs placed on the same sub datacenter should not exceed the datacenter's capacity.

4.2 MCNVMA Algorithm

The basic idea of our MCNVMA algorithm is: First of all, we identify a subset of the datacenters with minimizing length of the paths between the datacenters. Additionally

we need to determine the datacenter assignment for each individual VM. For this assignment, our objective is to minimize the inter datacenters traffic between the VMs.

Firstly, we select a set of datacenters to place the VMs. We view this problem as a sub-graph selection problem, which is finding a maximum sub-graph with a given diameter. For general graphs, it is an NP-hard problem and cannot be approximated within 2-ε for any ε>0)[15].

Given a complete graph *G = (V, E, c, l)*. The vertices V represent the datacenters, and weights $C_{dci} = (Pro_{dci}, Mem_{dci})$ on them denote the number of processors and the amount of memory of the datacenter respectively. The edges *E* represent the path between the datacenters and length *l* on them denotes the communication distance between the sub datacenters (in server selection phase denotes bandwidth). Let *s* be the number of VMs requested by the user. Then we should find a sub graph of *G*, denoted as G', with m' vertices. G' meeting the flowing conditions<1, 2>, with minimum maximal distance.

For $\forall i \in \{1,2,\ldots,m'\}$,

$$\sum_{j=1}^{n} Pro_{vmj}(i) \le Pro_{dci} \quad (1)$$

$$\sum_{j=1}^{n} Mem_{vmj}(i) \le Mem_{dci} \quad (2)$$

where $Pro_{vmj}(i)$ equals to p_j (represent the number of processors VM_j required) if VM_j is deployed on data center *i*, otherwise is zero; and $Mem_{vmj}(i)$ equals to m_j (represent the amount of memory VM_j required) if VM_j is deployed on datacenter $_i$, otherwise also is zero.

Since the original graph is a complete graph, the subgraph induced by the selected vertices is also complete. And all of the complete graphs are their self-clique. Hence, our goal is to find such a clique whose length of the longest edge is minimum.

```
Algorithm: MCNVMA(G, N, W[][],Value[])
Input: G =(V, E, w, l)
    N: the amount of VMs;
    Value[]: value of VMs;
Output: min_diameter.

for each vertex v∈V
   V' ← {v}, E' ← ∅, m ← Count(V), weight ← w(v);
   AllocatedVM←TDKnapsack(weight,W,Cost);
   ToAllocateVM←W-AllocatedVM;
   Sorted the vertices of G in increasing order of length to
v, noted as u1, u2, u3, ..., um-1.
   i ← 0
   while (i < m)
      weight ← w(vi);
      perform two dimensional knapsack algorithm.
```

```
      diameter ← max⟨diameter, l(v_j, u_i)⟩ ;
      V′ ← V′ ∪ {u_i};
      E′ ← E′ ∪ {(v′, u_i), v′ ∈ V′};
      compute to remain to allocated VMs.
     i ← i + +
  end while
  if min_diameter >diameter
     min_diameter ←diameter
  else if min_diameter > diameter
Compare the communication cost of clique
end while
end for
return min_diameter
```

Once a datacenter is selected, we will assign individual VMs to it. This assignment is done during the process of datacenter selection. This problem can be regarded as a variant of two dimensional knapsack problem. We regard each datacenters as a knapsack and each VM as an item. The capacity of knapsack consists of available processors and memory. As mentioned in the problem statement, each item has two different kinds of cost (processors and memory). The value of each VM is the sum of the bandwidth, which is needed while communicating with others. The value is dynamic change along with the implementation of the algorithm. Our goal is to find the allocation of VMs with the maximum value under the conditions of limited capacity and minimum communication.

The algorithm finds a sub-clique satisfying the constraints that include vertex v and places VMs by invoking two dimensional knapsack algorithm which is implemented by the dynamic programming method. After completing the two dimensional knapsack algorithm, the VMs with lager communication requirement are placed on the same datacenter, then the communication cost between different datacenter is least. The algorithm finds maximum clique including vertex v by adding nodes in an increasing order of length to v, until the weight of the clique meeting the constraints. The algorithm computes the diameter of the resulting sub-clique as well. This is done by maintaining the diameter as the nodes are added. When a node is added, the diameter of the sub-graph change only if the length of the edges induced by that node is greater than the current diameter.

The algorithm finds a sub-clique meeting the constraints through the while loop for each of the vertices, and selecting the one with the smallest diameter.

4.3 Example and Analysis

Now let's illustrate how to solve the problem introduced in Section 3.2 using MCNVMA algorithm. First of all, we need to map datacenters onto the nodes of a graph(Fig 3). The weights of nodes correspond to the capacity of datacenters (Table 3), and the weights of edges correspond to distances between sub-datacenters (Table 4).

Firstly, we select a arbitrarily node, such as n_0 as the start node. We use two dimensional knapsack algorithm to assign VMs set on it. The sum of each row of Table 2 is the VM's value (Table 5), which changing dynamically. Now we place {VM_3, VM_6, VM_7, VM_8} on n_0. As n_1 to shortest among n_0's neighbors. Next, we select and place VMs on n_1 from the remaining VMs list ({VM_0, VM_1, VM_2, VM_3, VM_4, VM_5, VM_9 }). This procedure repeats until all of the VMs have been allocated.

Table 5. VM value list

VM	Value(Mbps)
VM_0	1.9
VM_1	2.1
VM_2	1.35
VM_3	2.71
VM_4	1.6
VM_5	1.7
VM_6	2.61
VM_7	2.76
VM_8	3.22
VM_9	2.03

Table 6. Result

Start node	Clique	Diameter
n_0	{ n_0, n_1, n_2}	7
n_1	{ n_1, n_4, n_0}	10
n_2	{ n_2, n_1, n_0}	7
n_3	{ n_3, n_1, n_2}	9
n_4	{ n_4, n_1, n_0}	10

We record the nodes and the diameter of the sub graph, which is induced by the nodes (Table 6). Then we select the clique with minimum diameter C={n_0, n_1, n_2}, that is datacenter DC_0, DC_1, DC_2.

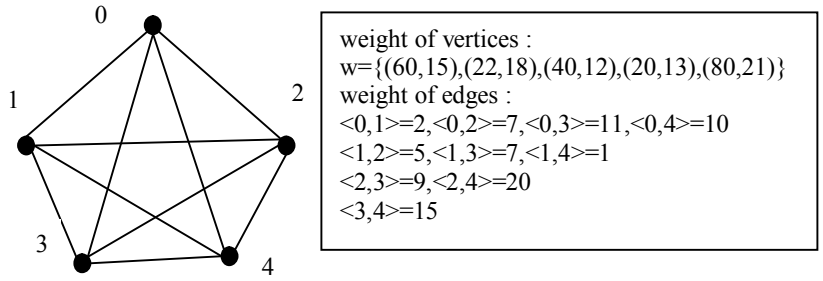

Fig. 1. Graph G=(V,E)

Analysis: In the while loop needs to sort the lengths of edges incident on a node, which takes *nlogn* time, where *n* is number of datacenters. While loop in the algorithm may be executed once per node. Computing diameter takes $O(n^2)$ as there are n^2edges. And the dynamic programming based two dimension knapsack algorithm takes O(*mn*) time as there are *m* VMs and *n* data centers. Hence, in the worst case, the running time of *while loop* is $O(n^2)$. Algorithm *MCNVMA* executes *the while loop* times, one for each node so that the worst case complexity is $O(n^3)$.

5 Evaluation

5.1 Experiment Settings

We compare MCNVMA algorithm with Random algorithm and Greedy algorithm. Random algorithm selects a random datacenter and places VMs as many as possible on it randomly. If there are more VMs in the request than available in the datacenter, then the algorithm chooses the next datacenter randomly to place the remaining VMs. This process is repeated until all the VMs are placed. Greedy algorithm selects the datacenter with maximum capacity. It places as many VMs from the request as possible on that datacenter. If there are remaining VMs in the request to be placed, then the algorithm selects the next datacenter with the largest free capacity. This process continues until all the VMs are placed.

To measure the performance of the algorithms, we create random topologies and user requests, and measure the maximum distance between any two VMs in the placement output by these algorithms. The locations of datacenters are randomly selected from 500x500 grid. In each of the experiments below, we report the results as average of 100 runs.

5.2 Experiment Results

Firstly, we measure the maximum distance of the placement for a request of 100 VMs. By varying the number of datacenters, we compare the maximum distance between any two VMs, see Figure 2 and 3. MCNVMA algorithm has much smaller distance than random algorithm. The communication cost between datacenters in MCNVMA is lower than other two.

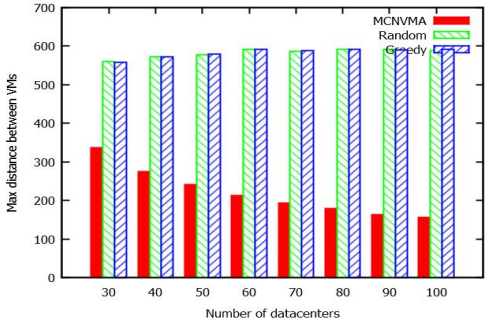

Fig. 2. Max distance for 100 VMs

Now, we set the number of datacenter to 50. We vary the number of VMs to evaluate the maximum distance and the stability of the algorithm. From figure 5 we can see that the maximum distance computed by MCNVMA algorithm is much lower than random algorithm and greedy algorithm. And the stability of MCNVMA is similar to the other two algorithms (Fig. 6).

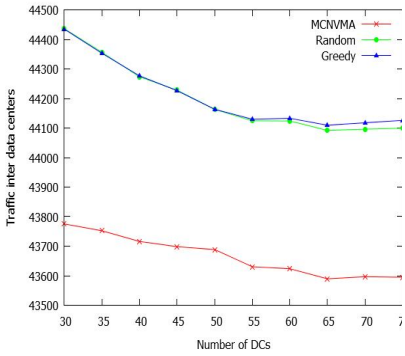

Fig. 3. Communication Cost for 100 VMs

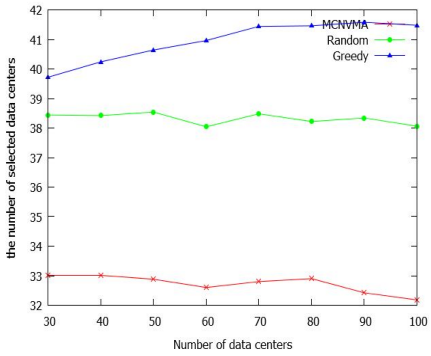

Fig. 4. the number of selected datacenters for

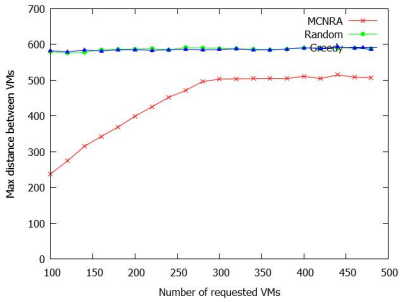

Fig. 5. Maximum distance for 50 DCs

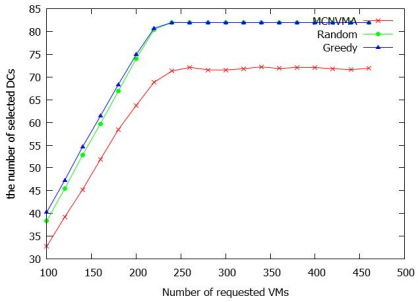

Fig. 6. Algorithm Stability for 50 DCs

6 Conclusions and Future Work

In this work we provide a allocation algorithm (MCNVMA) for VMs in cloud datacenter. Each VM can has its own configuration requirements in MCNVMA. The communication cost among VMs, which can be collected, are also defined as requirements as well. This algorithm try to find solutions with short communication path among VMs while user's requirements can be satisfied. From experimental results, the MNCRA algorithm reduces the communication cost between the VMs especially in large scale datacenters.

In this paper, only network factors are considered. However, energy saving, load balancing and other factors should also be considered in real applications. We are going to explore other market models, such as spot market model as well.

Acknowledgements

This work is partially supported by China National Science Foundation (Granted Number 61073021, 61272438), Research Funds of Science and Technology Commission of Shanghai Municipality (Granted Number 11511500102, 12511502704), Cross Research Fund of Biomedical Engineering of Shanghai Jiaotong University (YG2011MS38). The work described in this paper was supported by Morgan Stanley. Morgan Stanley and Shanghai Jiao Tong University Innovation Center of Computing in Financial Services have entered into Collaboration Agreement No. CIP-A20110324-2.

References

1. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zahari. : A View of Cloud Computin.*Communications of the ACM*, Vol. 53, No. 4, pp. 50-58, 2010.
2. Bhardwaj S, Jain L, Jain S.: Cloud computing: A study of infrastructure as a service (IAAS)[J]. International Journal of engineering and information Technology, 2010, 2(1): 60-63.
3. C. Waldspurger.:Memory resource management in VMware ESX server. ACM SIGOPS Operating Systems Review, vol. 36, no. SI, p. 194, 2002.
4. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield.:Xen and the art of virtualization. in Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003, p. 177.
5. A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori.: kvm: the Linux virtual machine monitor. inProceedings of the Linux Symposium, vol. 1, 2007, pp. 225–230.
6. "Openvz: Server virtualization open source project," http://openvz.org, 2010.
7. Ye K, Huang D, Jiang X, et al. Virtual machine based energy-efficient data center architecture for cloud computing: a performance perspective[C]//Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing. IEEE Computer Society, 2010: 171-178.
8. Cao J, Wu Y, Li M. Energy efficient allocation of virtual machines in cloud computing environments based on demand forecast[M]. Advances in Grid and Pervasive Computing. Springer Berlin Heidelberg, 2012: 137-151.
9. Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing[J]. Future Generation Computer Systems, 2012, 28(5): 755-768.
10. R. Buyya, A. Beloglazov, J. Abawajy, Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges, in: Proceedings of the 2010 International Conference on Parallel andDistributed Processing Techniques and Applications, PDPTA 2010, Las Vegas, USA, 2010.
11. Beloglazov, Anton, and Rajkumar Buyya. "Energy efficient resource management in virtualized cloud data centers." Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. IEEE Computer Society, 2010.
12. Meng X, Pappas V, Zhang L.: Improving the scalability of data center networks with traffic-aware virtual machine placement[C]. INFOCOM, 2010 Proceedings IEEE. IEEE, 2010: 1-9.

13. Wang M, Meng X, Zhang L.: Consolidating virtual machines with dynamic bandwidth demand in data centers[C]. INFOCOM, 2011 Proceedings IEEE. IEEE, 2011: 71-75.
14. Biran O, Corradi A, Fanelli M, et al.: A stable network-aware vm placement for cloud systems[C]. Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). IEEE Computer Society, 2012: 498-506.
15. Alicherry, Mansoor, and T. V. Lakshman. "Network aware resource allocation in distributed clouds." INFOCOM, 2012 Proceedings IEEE. IEEE, 2012.
16. Liu, Liang, et al. "GreenCloud: a new architecture for green data center." Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session. ACM, 2009.
17. S. Shang, Y. Wu, J. Jiang and W. Zheng, “An Intelligent Capacity Planning Model for Cloud Market”, Journal of Internet Services and Information Security, 1:1, 37-45.
18. F. Machida, M. Kawato and Y. Maeno, “Redundant Virtual Machine Placement for Fault tolerant Consolidated Server Clusters”, Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium, Osaka, Japan, 2010, 32-39.
19. S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubrahmanian, K. Talwar, L. Uyeda and U. Wieder, “Validating Heuristics for Virtual Machine Consolidation”, Microsoft Research, MSR-TR-2011-9, January 2011.