Task Based System Load Balancing Approach in Cloud Environments

Fahimeh Ramezani, Jie Lu and Farookh Hussain

Abstract Live virtual machine (VM) migration is a technique for transferring an active VM from one physical host to another without disrupting the VM. This technique has been proposed to reduce the downtime for migrated overload VMs. As VMs migration takes much more times and cost in comparison with tasks migration, this study develops a novel approach to confront with the problem of overload VM and achieving system load balancing, by assigning the arrival task to another similar VM in a cloud environment. In addition, we propose a multi-objective optimization model to migrate these tasks to a new VM host applying multi-objective genetic algorithm (MOGA). In the proposed approach, there is no need to pause VM during migration time. In addition, as contrast to tasks migration, VM live migration takes longer to complete and needs more idle capacity in host physical machine (PM), the proposed approach will significantly reduce time, downtime memory, and cost consumption.

Keywords Cloud computing · Multi-objective genetic algorithm · Virtual machine migration · Task based system load balancing algorithm

F. Ramezani (✉) · J. Lu · F. Hussain
Decision Systems and e-Service Intelligence Laboratory, Faculty of Engineering and IT, School of Software, Centre for QCIS, University of Technology, Sydney, Australia
e-mail: Fahimeh.Ramezani@students.uts.edu.au

J. Lu
e-mail: Jie.Lu@uts.edu.au

F. Hussain
e-mail: Farookh.Hussain@uts.edu.au

F. Sun et al. (eds.), *Knowledge Engineering and Management*,
Advances in Intelligent Systems and Computing 214,
DOI: 10.1007/978-3-642-37832-4_4,

1 Introduction

Cloud computing provides new business opportunities for both service providers and requestors clients (e.g., organizations, enterprises, and end users), by means of a platform and delivery model for delivering Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). A cloud encloses the IaaS, PaaS, and/or SaaS inside its own virtualized infrastructure, in order to carry out an abstraction from its underlying physical assets. Typically, the virtualization of a service implies the aggregation of several proprietary processes collected in a virtual environment, called Virtual Machine (VM) [1, 2].

Often, clouds are also spread over distributed virtualization infrastructure covering larger geographical areas (example, let us think about Amazon ('Amazon Elastic Compute Cloud [24]'), Azure [25], and RESERVOIR (an European project facing the cloud computing IaaS topic [3]). In addition, the perspective of cloud federation [4, 5], where cloud providers use virtualized infrastructures of other federated clouds, opens toward new scenarios in which more and more types of new services can be supplied. In fact, clouds exploiting distributed virtualization infrastructures are able to provide new types of "Distributed IaaS, PaaS, and SaaS" [2].

Cloud computing platform using virtualization technology for resource management, achieves dynamic balance between the servers. Using online VM migration technology [6] can online achieve the remapping of VMs and physical resources, and dynamic achieve the whole system load balancing [7]. In modern data center (DC) or cloud environment, virtualization is a critical element since using virtualization the resources can be easily consolidated, partitioned, and isolated. In particular, VM migration has been applied for flexible resource allocation or reallocation, by moving VM from one physical machine to another for stronger computation power, larger memory, fast communication capability, or energy savings [8].

Although a significant amount of research has been done to achieve the whole system load balancing ([6–9], etc.), more improvement is still needed as most of these approaches tried to migrate VMs, when they became overloaded. As VMs migration takes much more times and cost in comparison with tasks migration, we believe migrating tasks from overloaded VMs instead of migrating overloaded VMs, will significantly reduce transfer time and total cost. In addition, to migrate VMs, we have to find a new PM which can accommodate the VM being migrated, and we can rarely avoid choosing an idle PM to optimize power consumption. But in task migration, we just need to find another VM which is located on an active PM and has the same features and number of CPU and more capacity just for executing a task.

Considering these facts and lack of resources in this area, we developed a novel Task Based System Load Balancing (TBSLB) approach to achieve system load balancing and confront with the lack of capacity for executing new task in one VM, by assigning the task to another homogeneous VM in cloud environment. In

addition, we proposed an algorithm to solve the problem of migrating these tasks to new VM host which is a multi-objective problem subject to minimizing cost, minimizing execution time, and transferring time. To solve this problem, we applied multi-objective genetic algorithm (MOGA).

The rest of this paper is organized as follows. In Sect. 2 related works about VM migration are described. In Sect. 3 we propose a conceptual model and the algorithm of TBSLB approach for solving the problem of overloaded VMS by optimal tasks migration from overloaded VMs. The MOGA algorithm is described in Sect. 4. Our developed algorithm for solving multi-objective tasks scheduling problem and completing TBSLB algorithm, is described in Sect. 5. The proposed approach is evaluated in Sect. 6. Finally, we present our conclusion and future work in Sect. 7.

2 Related Works for VM Migration

Virtualization has delivered significant benefits for cloud computing by enabling VM migration to improve utilization, balance load, and alleviate hotspots [10]. Several mechanisms have been proposed to migrate a running instance of a VM (a guest operating system) from one physical host to another to optimize cloud utilization.

VM migration is a hot topic of computing system virtualization. Primary migration relies on process suspend and resume. Many systems [11–13] just pause the VM and copy the state data, then resume the VM on the destination host. This forces the migrated application to stop until all the memory states have been transferred to the migration destination where it is resumed. These methods cause the application to become unavailable during the migration process. ZAP [14] could achieve lower downtime of the service by just transferring a process group, but it still uses stop-and-copy strategy. To reduce the migration downtime and move the VM between hosts in local area network without disrupting it, VMotion [15] and Xen [6] utilize precopy migration technique to perform live migration and support seamless process transfer. Based on their works, [16] tried migrating running VM on a wide area network [8].

In precopy migration technique, VMs migrate by precopying the generated run-time memory state files from the original host to the migration destination host. If the rate for such a dirty memory generation is high, it may take a long time to accomplish live migration because a large amount of data needs to be transferred. In extreme cases, when dirty memory generation rate is faster than precopy speed, live migration will fail. Considering this fact, [8] presented the basic precopy model of VM live migration and proposed an optimized algorithm to improve the performance of live migration by limiting the speed of changing memory through controlling the CPU scheduler of the VM monitor [8].

[7] designed an IPv6 live migration framework for VM based on IPv6 network environment [7]. The framework has been used IPv6 VM live migration in

different IPv6 network. They designed a global control engine as a core complete IPv6 live migration for VM, and provided IPv6 cloud computing service for IPv4/IPv6 client. In their approach, during VM migration process, the source VM would continue to offer services, and the source VM is still not stopped, but no longer provides new services until the old service completion then stop the source VM.

Since power is one of the major limiting factors for a DC or for large cluster growth, [9] proposed a runtime VM mapping framework in a cluster or DC to save energy [9]. Their placement module focused on reducing the power consumption. The main point of their approach is how to map VMs onto a small set of PMs without significant system performance degradation. Actually, they tried to turn off the redundant nodes or PMs to save energy, while the remaining active nodes guarantee the system performance. In their GreenMap framework, one probabilistic, heuristic algorithm is designed employing the idea from simulated annealing (SA) optimization, for the optimization problem: mapping VMs onto a set of PMs under the constraint of multi-dimensional resource consumptions.

[17] believe that most of the proposed methods for on-demand resource provisioning and allocation, focused on the optimization of allocating physical resources to their associated virtual resources, and migrating VMs to achieve load balance and increase resource utilization. Unfortunately, these methods require the suspension of the executing cloud computing applications due to the mandatory shutdown of the associated VMs [17]. To overcome this drawback, they proposed a threshold-based dynamic resource allocation scheme for cloud computing that dynamically allocates the VMs among the cloud computing applications based on their load changes. In their proposed method, they determined when migration should be done but they did not specify the details of how the reallocation will occur.

A fundamental shortcoming of the most existing research is that they consider complete VM migration to overcome overload VM and achieve system load balance. To improve previous approaches and reduce time and cost consumption in such situation, we proposed a new TBSLB approach which migrate tasks from overloads VMs instead of whole VM migration, and to decrease power consumption, a set of VMs on active PMs will be chosen as a new tasks' host. In addition, our approach not only eliminates the process suspend and resume which will happen in VM migration, but also omits precopy mechanism and producing dirty memory in live VM migration.

3 A Conceptual Model and Main Algorithm for Task Based System Load Balancing

In this section, we describe proposed TBSLB approach. This approach contains a conceptual model and TBSLB algorithm which are designed to achieve whole system load balancing by migrating tasks from overloaded VMs. In this approach to

decrease energy consumption and costs, we avoid choosing idle PMs or Computer Nodes (CNs) as a new PM host, because if we transfer tasks to an idle PM, we have to turn it on and this action will increase energy consumption and costs [9].

As cloud computing has the advantages of delivering a flexible, high-performance, pay-as-you-go, on-demand offering service over the Internet, common users and scientists can use cloud computing to solve computationally complex problems (complex applications). The complex applications can be divided into two classes. The one is computing intensive, the other is data intensive. For transferring data intensive applications, the scheduling strategy should decrease the data movement which means decreases the transferring time; but for transferring computing intensive tasks, the scheduling strategy should schedule the data to the high-performance computer [18]. In this paper, we consider bandwidth as a variable to minimize the tasks transferring time for data intensive applications. In addition to enhance performance utilization for computing intensive applications, we consider new host PM's properties (memory, hard disk, etc.).

In cloud environment, there are some tasks schedulers that consider task types, priorities, and their dependencies to schedule tasks in optimal way considering their specific VM's resources. In our proposed system, we design a schedulers' blackboard, where all cloud schedulers (which manage VMs on clouds (see Fig. 1)) share their information about VMs, their features, and their tasks. We apply the information of this blackboard to find an appropriate host VM for the task. Furthermore, the criteria of QoS as SLA information are mentioned in this blackboard.

Every VM has already some tasks to execute and they have limited workload. To determine the time of tasks migration from an overloaded VM, we have to determine online remained workload capacity of a VM (VMs workload information), we defined it as:

$$\mathrm{VM}_{\mathrm{rw}} = \mathrm{VM}_{w} - \mathrm{VM}_{\mathrm{et}} \tag{1}$$

where VM_{w} is VM workload, and $\mathrm{VM}_{\mathrm{et}}$ is the number of executing tasks in VM. The VM will be overload and arrival tasks should be migrated to another similar VM to execute, when:

$$\mathrm{VM}_{\mathrm{rw}} \leq 1 \tag{2}$$

Considering all these facts, we propose a novel TBSLB algorithm which prepares another scheduler to transfer tasks from an overhead VM to a new similar and appropriate VM according to following steps:

Step 1 Gathering data and information about VMMs, VMs, PMs, and SLA information, in the global blackboard as ***inputs*** of TBSLB algorithm as follow:

1. VMs tasks information:

   1.1 The number of executing tasks
   1.2 Tasks' execution time
   1.3 Tasks' performance model

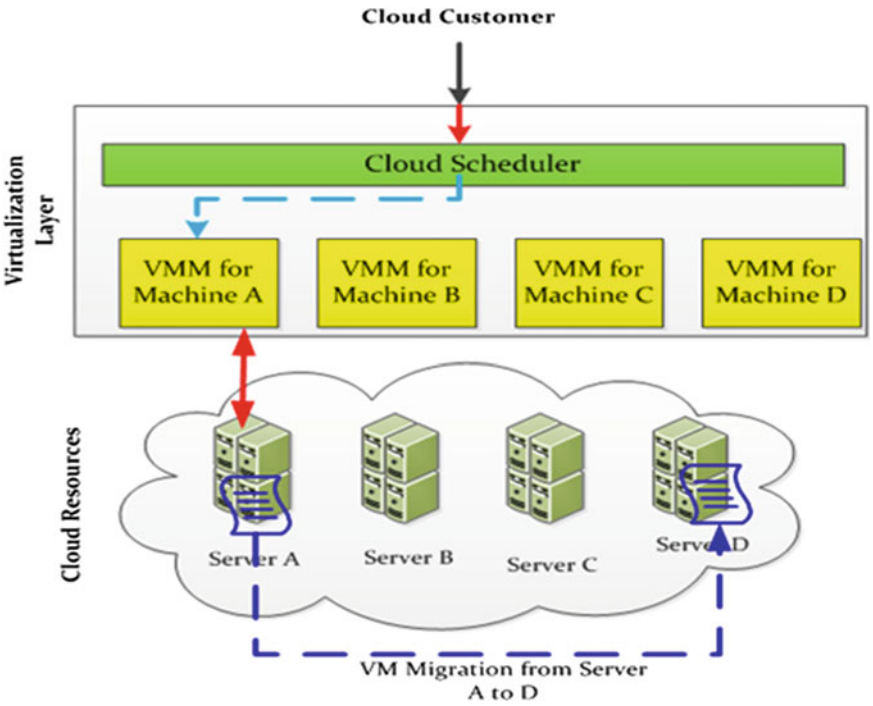

Fig. 1 Cloud architecture

1.4 Tasks' locations
1.5 Tasks' required resources (number of required processors)

2. PMs' Criteria (total/current)

2.1 CPU (number and speed of the processors)
2.2 Free Memory and Hard disk
2.3 Bandwidth
2.4 Idle or active
2.5 Its host VMM

3. SLA information
4. The objectives of the tasks migration optimization model and their information:

4.1 Minimizing cost

4.1.1 Cost information

4.2 Minimizing execution time and transferring time

4.2.1 Execution information
4.2.2 Bandwidth information

Step 2 Monitoring data and information to determine VMs' workflow situation and determining:

1 VMs workload information
2 Overloaded VMs
3 The tasks which should be migrated from overloaded VM's
4 Migration time

Step 3 Finding optimal homogeneous VMs as a new host for executing the tasks of the overloaded VMs, which is a multi-objective task migration problem, applying MOGA (this step will be described in Sect. 5).

Step 4 Considering obtained optimal tasks migration schema, determining following information as the *outputs* of TBSLB algorithm:

1 New optimal cost
2 New optimal execution time
3 Current VMs properties (Executing tasks, CPU, etc.)

Step 5 Transferring tasks and their corresponding data to the optimal host VMs

Step 6 Updating blackboards and schedulers' information according to the outputs of Step 4.

Step 7 End.

The conceptual model of the proposed approach is summarized in Fig. 2.

4 A Multi Objective Genetic Algorithm

In Step 3 of the algorithm, a multi-objective problem which is described in Sect. 5, should be solved to optimize tasks migration from overload VM and find the best VMs as new tasks hosts. In multi-objective optimization problems, each objective function interacts on each other; they almost cannot be optimal at the same time. In other words, one objective function optimization often means a bad developing direction of other objective functions. Therefore, a compromise strategy can be used among the objective functions so as to make them reach optimization at the same time. Now the most popular MOGAs abroad are Corne's PESA2 [19] and PAES [20], SPEA2 [21] proposed by Ziltler, Deb's NSGAII, and so on. Among them, Deb's NSGAII not only has good convergence and distribution but also has higher convergence speed, and solve the shortcomings that shared parameters are difficult to determine [22].

According to MOGA, first initial population whose scale is N is generated randomly. The first generation child population is gained through non-dominated sorting [23] and basic operations such as selection, crossover and mutation. Then, from the second generation on, the parent population and the child population will be merged and sort them based on fast non-dominated. Calculate crowding distance among individuals on each non-dominated layer. According to non-dominant relationship and crowding distance among individuals, select the appropriate

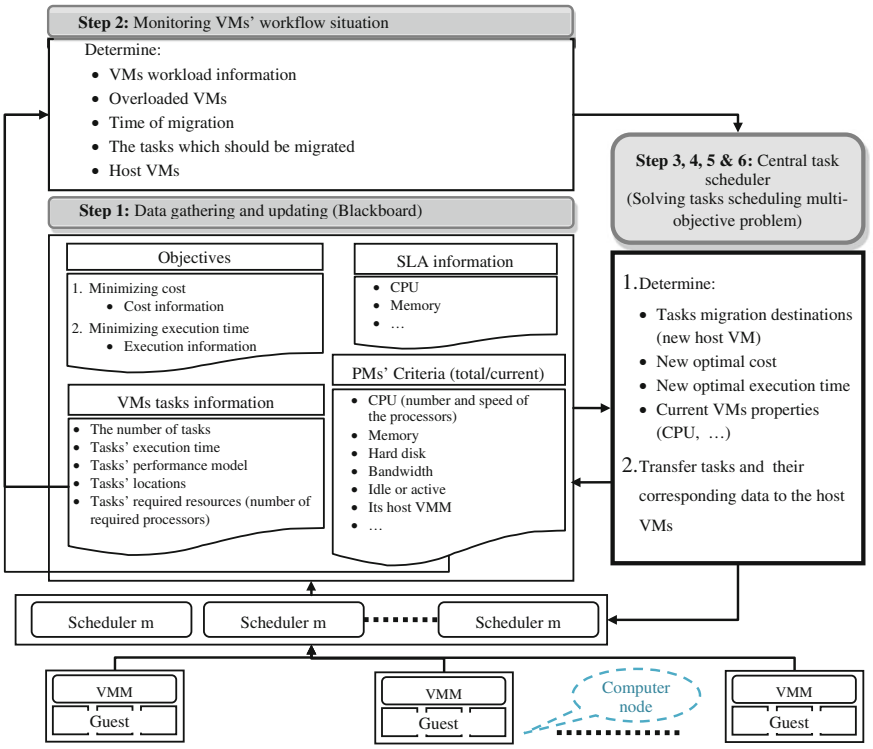

Fig. 2 The conceptual model of TBSLB approach

individuals to form a new parent population. Finally, new child population is generated through basic operations of genetic algorithm. And so on, until the conditions of the process end can be met [22].

5 An Algorithm for Solving Multi-Objective Tasks Migration Problem Using MOGA

In this section, we will describe the sub-TBSLB algorithm which is developed to complete the Step 3 of TBSLB algorithm and solve Multi-objective tasks migration problem. This sub-algorithm will determine the most appropriate VMs to assign the tasks of the overloaded VMs and find optimal tasks scheduling model applying MOGA.

This sub-algorithm applies data and information which are determined in Step 1 and Step 2 of the TBSLB algorithm as its ***inputs***. In this algorithm, we first eliminate those VMs which do not satisfy all constraints to reduce the population

of the candidate solution set. Then we apply MOGA method to find the optimal solution.

According to sub-TBSLB algorithm, to find optimal host VMs to assign new overloaded VMs' tasks, following steps should be conducted:

Step 3.1 Determining candidate host VMs set by choosing the set of VMs which satisfy the constraints about host VMs' properties as $\mathrm{VM}_{\mathrm{set}} = \{\mathrm{vm}_1, \ldots, \mathrm{vm}_m\}$

Step 3.2 Determining overloaded VMs applying Eq. 2, and eliminating them from candidate host VMs set.

Step 3.3 Determining the set of tasks which should migrate from overloaded VMs as immigrating tasks set: $T_{\mathrm{set}} = \{t_1, \ldots, t_n\}$

Step 3.4 Applying MOGA to solve multi-objective problem and assign the immigrating tasks to the optimal host VMs minimizing execution time, transferring time and processing cost. To achieve this goal, following steps should be conducted:

Step 3.4.1 Initializing population P_0 which is generated randomly

Step 3.4.2 Assigning rank to each individual based on non-dominated sort

Step 3.4.3 Implementing binary tournament selection, crossover and mutation on the initial population and creating a new population Q_0 and set $t = 0$

Step 3.4.4 Merging the parent P_t and the child Q_t to form a new population $R_t = P_t \cup Q_t$

Step 3.4.5 Adopting non-dominated relationship to sort population and calculate the crowding distance among population on each layer

Step 3.4.6 Selecting the former N individuals as the parent population, namely $P_{t+1} = P_{t+1}$ [1: N] (Elite strategy)

Step 3.4.7 Implementing reproduction, crossover and mutation on population P_{t+1} to form population Q_{t+1}

Step 3.4.8 If the termination conditions are met, *output* results as optimal tasks migration schema; otherwise, update the evolutionary algebra counter $t = t+1$ and go to step 3.4.4

Step 3.5 End.

6 Evaluation

We determine two parameters to evaluate our proposed TBSLB approach and compare it with traditional whole VM migration methods. The first parameter is related to "power consumption". As, the less number of active PM means the less power consumption [9], we applied following ratio to compare power consumption after load balancing:

$$R_{\mathrm{pc}} = \frac{\text{Number of active PM}}{\text{Number of overloaded VMs}} \tag{3}$$

In proposed approach, to execute some tasks of overloaded VM, we need to find a new similar VM on an active PM as a new host and there will be no need to turn a new PM on. In contrast, for whole VM migration, more hardware capacity will be needed and it is impossible for every case to avoid choosing idle PM. So, in our approach, R_{pc} should be less than whole VM migration attitude for load balancing. Therefore, we will have less "power consumption" after load balancing and:

$$R_{\mathrm{pc}_{\mathrm{Offline\ VM}}} \geq R_{\mathrm{pc}_{\mathrm{Online\ VM}}} > R_{\mathrm{pc}_{\mathrm{NewApproach}}}$$

To compare the efficiency of TBSLB approach, we applied "downtime VM pause time)" as second parameter and estimate the amount of "idle memory" which is prepared during the time of solving the problem of overloaded VM as:

$$M_{\mathrm{im}}(t) = \mathrm{OriginalVM_m}(t) + \mathrm{HostVM_m}(t) \tag{4}$$

where $OriginalVM_m$ and $HostVM_m$ are the amount of original VM memory and host VM, respectively.

In offline VMs migration method, during VM migration time, the original VM should be suspend and its memory and the amount of memory in new host PM which is determined for host VM will be idle. In online VMs migration method, although VM will not be suspended during migration process, the amount of memory in new host PM will be idle in this time. Meanwhile, in TBSLB approach, we eliminate process of suspend and resume in primary VM migration which is mentioned in [8] and there will be no downtime for VMs and no idle memory. As the results:

$$M_{\mathrm{im}}(t)_{\mathrm{Offline\ VM}} > M_{\mathrm{im}}(t)_{\mathrm{Online\ VM}} > M_{\mathrm{im}}(t)_{\mathrm{NewApproach}}$$

7 Conclusion and Future Work

VM migration has been applied for flexible resource allocation or reallocation, by moving overload VM from one PM to another to achieve stronger computation power, larger memory, fast communication capability, or energy savings.

This paper proposed a new TBSLB approach to confront with the problem of overload VM by migrating arrival tasks to another homogeneous VM. This algorithm contains multi-objective tasks migration model subject to minimizing cost, execution time, and transferring time. In proposed approach, there is no need to pause VM during migration time. In addition, as contrast to tasks migration, VM live migration takes longer to complete and needs more idle capacity in host PM, the proposed approach will significantly reduce time, downtime memory, and cost consumption. Furthermore, proposed approach will decrease energy consumption

by avoiding choosing idle PMs or CNs as a new host PM. In our future work, we will propose a method to predict the time of task migration from an overload VM to accelerate load balancing process in our proposed approach.

References

1. Buyya R, Broberg J, Goscinski A (eds) (2011) Cloud computing: principles and paradigms
2. Celesti A, Fazio M, Villari M, Puliafito A (2012) (VM) provisioning through satellite communications in federated cloud environments. Future Gener Comput Syst 28(1):85–93
3. Rochwerger B, Breitgand D, Epstein A, Hadas D, Loy I, Nagin K, Tordsson J, Ragusa C, Villari M, Clayman S (2011) Reservoir-when one cloud is not enough. Comput 44(3):44–51
4. Goiri I, Guitart J, Torres J (2010) Characterizing cloud federation for enhancing providers' profit. In: IEEE 3rd international conference on cloud computing (CLOUD), pp 123–130
5. Ranjan R, Buyya R (2008): Decentralized overlay for federation of enterprise clouds, Arxiv preprint arXiv:0811.2563
6. Clark C, Fraser K, Hand S, Jacob GH (2005) Live migration of (VM)s. In: Proceedings of 2nd ACM/USENIX symposium on network systems, design and implementation (NSDI)
7. Jun C, xiaowei C(2011): IPv6 (VM) live migration framework for cloud computing, Energy procedia, vol 13(0):5753–5757
8. Jin H, Gao W, Wu S, Shi X, Wu X, Zhou F (2011) Optimizing the live migration of (VM) by CPU scheduling'. J of Netw and Comput Appl 34(4):1088–1096
9. Liao X, Jin H, Liu H (2012) Towards a green cluster through dynamic remapping of (VM)s. Future Gener Comput Syst 28(2):469–477
10. Jain N, Menache I, Naor J,.Shepherd F(2012) Topology-aware VM migration in bandwidth oversubscribed datacenter networks, automata, languages, and programming, pp 586–597
11. Kozuch M, Satyanarayanan M (2002) Internet suspend/resume, Mobile computing systems and applications. Proceedings fourth IEEE workshop on, pp 40–46
12. Sapuntzakis CP, Chandra R, Pfaff B, Chow J, Lam MS, Rosenblum M (2002) Optimizing the migration of virtual computers. ACM SIGOPS operating systems review, vol 36, no. SI, pp 377–390
13. Whitaker A, Cox RS, Shaw M, Gribble SD(2004) Constructing services with interposable virtual hardware. In: Proceedings of the 1st symposium on networked systems design and implementation (NSDI), pp 169–182
14. Osman S, Subhraveti D, Su G, Nieh J (2002): The design and implementation of Zap: A system for migrating computing environments, ACM SIGOPS Operating systems review, vol 36, no. SI, pp 361–376
15. Nelson M, Lim BH, Hutchins G (2005) Fast transparent migration for (VM)s, pp 25–25
16. Travostino F, Daspit P, Gommans L, Jog C, De Laat C, Mambretti J, Monga I, Van Oudenaarde B, Raghunath S, Yonghui Wang P (2006) Seamless live migration of (VM)s over the MAN/WAN. Future Gener Comput Syst 22(8):901–907
17. Lin W, Wang JZ, Liang C, Qi D (2011) A threshold-based dynamic resource allocation scheme for cloud computing. Proced Eng 23:695–703
18. Guo L, Zhao S, Shen S, Jiang C (2012) Task scheduling optimization in cloud computing based on heuristic algorithm. J Netw 7(3):547–553
19. Corne DW, Jerram NR, Knowles JD, Oates MJ (2001) PESA-II: Region-based selection in evolutionary multiobjective optimization, Citeseer
20. Knowles JD, Corne DW (2000) Approximating the nondominated front using the Pareto archived evolution strategy. Evolut compu 8(2):149–172
21. Zitzler E, Laumanns M, Thiele L(2001) SPEA2: Improving the strength pareto evolutionary algorithm

22. Zhang Y, Lu C, Zhang H, Han J(2011) Active vibration isolation system integrated optimization based on multi-objective genetic algorithm, computing, control and industrial engineering (CCIE), IEEE 2nd international conference on, vol 1, pp. 258–261
23. Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2(3):221–248
24. Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/
25. Azure: Microsoft's service Cloud platform, http://www.microsoft.com/windowsazure