A Heuristic Clustering-Based Task Deployment Approach for Load Balancing Using Bayes Theorem in Cloud Environment

Jia Zhao, Kun Yang, *Senior Member, IEEE*, Xiaohui Wei, *Member, IEEE*, Yan Ding, Liang Hu, and Gaochao Xu

Abstract—Aiming at the current problems that most physical hosts in the cloud data center are so overloaded that it makes the whole cloud data center' load imbalanced and that existing load balancing approaches have relatively high complexity, this paper has focused on the selection problem of physical hosts for deploying requested tasks and proposed a novel heuristic approach called Load Balancing based on Bayes and Clustering (LB-BC). Most previous works, generally, utilize a series of algorithms through optimizing the candidate target hosts within an algorithm cycle and then picking out the optimal target hosts to achieve the immediate load balancing effect. However, the immediate effect doesn't guarantee high execution efficiency for the next task although it has abilities in achieving high resource utilization. Based on this argument, LB-BC introduces the concept of achieving the overall load balancing in a long-term process in contrast to the immediate load balancing approaches in the current literature. LB-BC makes a limited constraint about all physical hosts aiming to achieve a task deployment approach with global search capability in terms of the performance function of computing resource. The Bayes theorem is combined with the clustering process to obtain the optimal clustering set of physical hosts finally. Simulation results show that compared with the existing works, the proposed approach has reduced the failure number of task deployment events obviously, improved the throughput, and optimized the external services performance of cloud data centers.

Index Terms—Task deployment, load balancing, Bayes theorem, clustering, cloud computing

1 Introduction

Cloud computing [1], [2] is one of the most promising and valuable research directions following utility computing, grid computing and distributed computing [3] currently. It provides services of infrastructure, platform and software for users, and supplies the on-demand services to users through Internet [4]. Infrastructure as a Service (IaaS) is the basis of cloud computing. The cloud data center deploys a large number of physical hosts to supply user services. Since the remaining resource amount of every physical host changes constantly, the task cannot be placed surely to the physical host with the largest remaining resource amount every time. We assume that the tasks requested by users are deployed every time to a physical host which is chosen at random. If the resource amount of the submitted task is greater than the remaining resource amount of the chosen physical host, this physical host cannot handle this task. It will cause a deployment failure of this task. And when the requested resource amount of a task is similar to the remaining resource amount of the physical host in which this task will be executed, the time used for dealing with this task will relatively be longer. When a cloud data center receives task requests constantly, it will make the cloud data center load imbalanced and thus to result in that computing results cannot be responded to users timely and effectively. What causes make the cloud data center' load so imbalanced that it cannot provide efficient external services for users? In fact, in order to ensure the service performance, cloud computing data centers generally deploy tasks according to the highest load demands to the corresponding hosts. Therefore, most physical hosts remain relatively idle in most of the time and it is a waste of computing resource.

At present, the field of load balancing in cloud data centers is a hot issue. In order to achieve the load balancing of cloud data centers, it is necessary to choose the optimal physical host efficiently in the process of deploying tasks. Most existing work has focused on the problem of how to achieve the immediate load balancing within an algorithm cycle in which the proposed approaches are able to obtain the optimal task deployment policies for the current [5], [6]. There exists a limitation of overly concentrating on the optimal load balancing policy for the current deployment problem and thus to make the efficiency decreased and users waiting time prolonged unnecessarily. As is well-known,

- *J. Zhao is with the College of Computer Science and Technology, the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, No. 2699 Qianjin Street, Changchun 130012, China and the College of Computer Science and Engineering, ChangChun University of Technology, No. 2055 Yan'an Street, Changchun 130012, China. E-mail: zhaiyj049@sina.com.*
- *K. Yang is with the School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, United Kingdom. E-mail: kunyang@essex.ac.uk.*
- *X. Wei, Y. Ding, L. Hu, and G. Xu are is with the College of Computer Science and Technology and the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, No. 2699 Qianjin Street, Changchun 130012, China. E-mail: {xugc, hul, weixh}@jlu.edu.cn, dingyan11@mails.jlu.edu.cn.*

Manuscript received 7 Aug. 2014; revised 6 Feb. 2015; accepted 6 Feb. 2015. Date of publication 10 Feb. 2015; date of current version 20 Jan. 2016. Recommended for acceptance by S. Guo. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPDS.2015.2402655

load balancing is recognized as a means of providing satisfactory service performance for users while maximizing the availability and utilization of the whole cloud system rather than the final purpose. Further, the satisfactory service performance is easy to achieve immediately since the available amount of computing resource in cloud date centers is always larger than the requested amount though this kind of requirements is real-time and strict. Also, as the total resource amount requested by all users accumulated within a processing cycle has rarely possibility to approach the current total amount of available computing resource in a cloud data center, the availability and performance potential of the cloud data center don't need to keep being maximized at all times. Based on the points above, it is unnecessary to ensure that the optimal load balancing effect is reached through after each algorithm cycle in real time as long as the whole system has been always tending to a long-term optimal load balancing effect.

Based on this motivation, this paper has proposed a heuristic approach to finding the optimal physical hosts for tasks deployment by achieving a load balancing strategy through a long-term algorithm process and thus to obtain optimal performance. A constraint value is determined in the light of the resource amount of requested tasks. That is, each requested task has a constraint value. Then the physical hosts, whose constraint values of computing resource are greater than the constraint value of the tasks in the cloud data center, are clustered. And the physical hosts whose similarities are within a certain threshold value constitute a set through clustering. And the physical host set obtained by clustering methods is the physical host set whose physical hosts are optimal for deploying the task. Then, the tasks will be put into physical hosts in the set to deploy. The process of the clustering of physical hosts in the cloud data center is to find the optimal physical hosts for deploying tasks. Thus, through the task deployment strategy, not only the load balancing of the cloud data center can be achieved, but also efficient performance of external services can be provided for users.

This paper aims to achieve long-term load balancing of cloud data centers and provide efficient performance of external services. It can be achieved only by deploying requested tasks into the resource pool of the cloud data center efficiently and properly, which can improve computing efficiency of cloud data centers, and provide users with cloud data centers good performance of external service. The proposed load balancing strategy Load Balancing based on Bayes and Clustering (LB-BC) can find optimal physical hosts of task deployment effectively, and achieve the whole networks load balancing of cloud data centers chronically.

The rest of this paper are organized as follows: in the second section, the related work of the current approaches achieving load balancing of cloud data centers will be introduced briefly. And in the third section, first of all, a premise of the proposed research problem is pointed out, and then the proposed problem is formalized in detail. In the fourth section, the system architecture is designed first in cloud computing environment, and then the proposed solution is put forward for addressing the formalized research problem. In the end of this section, the design and implementation process of the proposed algorithm are introduced in detail. The fifth section shows simulation experiments and results, proving that the proposed approach has high efficiency. And the conclusion of the whole paper is made in the sixth section.

2 Related Work

Load balancing has always been a hot research topic of cloud data centers [7], and its goal is to ensure that every computing resource can handle tasks effectively and quickly. Ultimately, the utilization of resource is improved. Researchers have proposed a series of static, dynamic and hybrid scheduling strategies. In addition, there are also some studies using live migration technology of virtual machine to meet the cloud data centers' task requests which include performance requirements and load limitation.

Existing load balancing strategies are generally divided into two categories: static load balancing and dynamic load balancing (DLB). Static load balancing scheduling algorithms [8], [9], [10], [11] are commonly including round robin, weighted round robin, least connection method, weighted least connection method and so on. These static algorithms only use some static information which cannot reflect dynamic load changes in the cluster of hosts effectively and they have poor adaptive ability. Currently, most of open-source IaaS platforms have utilized static algorithms to conduct the resource scheduling. For example, Eucalyptus [12] platform uses round robin to assign virtual machines to different physical hosts in sequence to achieve load balancing. In [8], Wei et al. have employed the weighted minimum link algorithm, which means that different weights indicate the performance of the physical host. Then, the virtual machine will be allocated to the physical host which has the smallest ratio of the current number and the weight. The advantage of static scheduling algorithms is that it is simple to use. But in the large-scale cloud data centers whose resource heterogeneity is very strong and user demand isn't consistent, the load balancing effect is not ideal. The proposed LB-BC approach is derived the idea of reducing unnecessary computation complexity just like the existing static approaches. However, LB-BC is an efficient approach which has ability in dynamically achieving the long-term optimal load balancing effect.

Dynamic load balancing [13], [14], [15] is a NP-complete problem and a classic combinatorial optimization problem. It is mainly used in the field of distributed parallel computing, and its main objective is how to distribute the load more rationally among multiple hosts to avoid some phenomenon that some computing nodes are overloaded and some nodes are light-loaded and thus to improve the whole performance of systems. Additional communication overhead produced in the process of DLB will degrade the system performance of the dynamic load balancing. And with the increase of network latency between nodes, the impact of the additional communication overhead on the performance of DLB will further increase. Therefore, how to reduce communication overhead furthest between nodes in the process of DLB becomes an important problem which will influence the performance of DLB. At present, aiming at the problem of reducing additional communication overhead in the process of DLB, the solution is mainly using greedy algorithms to

deal with it. Load Receiver Strategy (LRS) algorithm which is put forward in [16] has used the way that the light load is received and allocated preferentially. Xu [17] et al. have proposed an efficient double-sided combinatorial auction model to dynamically achieve optimal goals.

In [18], Lau et al. have integrated the two strategies of heavy load priority and light load priority. They have proposed an adaptive load distribution algorithm to effectively reduce communication overhead of the load balancing process. Utilizing the greedy algorithms can solve the problem of load distribution. However, several algorithms above cannot meet greedy choice performance and the nature of optimal sub-structure at the same time. So these load distribution approaches often obtain the local optimal solutions. And the effect of solving the problem of load distribution under certain special circumstances is not ideal. Cloud data center cannot reach load balancing of the entire network.

Virtual machine migration placement strategy is a widely used strategy to achieve load balancing of cloud computing data centers currently [19], [20]. VMware load balancing solution is distributed resource scheduling (DRS) [21]. When DRS select the physical host for the virtual machine, it will check the load status of each physical host and choose the placement solution which can improve the overall load balance degree. And in the process of running a virtual machine, DRS will continuously monitor the load status of the cluster and use VMware VMotion technology to perform live migration of virtual machines between different physical servers. Thus, it can ensure load balancing and efficient utilization of physical resource of the entire cluster. In [22], Piao and Yan have proposed a network-aware virtual machine placement and migration approach to minimizing the data transfer time consumption and improving the overall application performance of the cloud data center. However, this approach probably results in a relatively lower utilization of resource of physical hosts and raises the operation cost of the cloud data center. Sonneck et al. in [23] have presented a decentralized affinity-aware migration technique that incorporates heterogeneity and dynamism in network topology and job communication patterns to allocate virtual machines on the available physical resources. The technique monitors network affinity between pairs of virtual machines and uses a distributed bartering algorithm, coupled with migration, to dynamically adjust virtual machine placement such that communication overhead is minimized and load balancing is achieved.

In [24], Shrivastava et al. have presented an approach to place the virtual machines with strong correlation of applications intensively. However, the load balancing problem and cost overheads of cloud data centers weren't considered. It only focused on virtual machines management to strengthen the management of cloud data centers and improve the performing efficiency of cloud data centers. Rahman and Graham [25] have proposed a hybrid approach that combines static and dynamic provisioning. Its algorithm is to adapt a good initial static placement of virtual machines in response to evolving load characteristics using live migration to improve the computing efficiency of cloud data centers. Dupont et al. [26] have proposed a flexible framework for the (re)allocation of virtual machines in a cloud data center as well as thus to calculate and work out the best placement of virtual machines, achieving the overall load balancing of cloud data centers. In addition, Zhao et al. [27] have proposed an approach MOGA-LS, which is a heuristic and self-adaptive multi-objective optimization algorithm based on the improved genetic algorithm (GA) and the theory of Pareto optimal solutions. It aims to achieve dynamic load balancing.

In paper, the proposed LB-BC approach presents a solution to dynamically achieve the process-based optimal load balancing with low computation complexity and thus obtain the efficient service performance and computing efficiency on two sides. The main contributions of this paper are as follows:

1) Introduce the concept of process-based load balancing in cloud computing environment, which is in contrast to the immediate load balancing strategies in the current literatures. The benefits are many fold: to reduce unnecessary computation complexity and to increase deployment efficiency while fulfilling the service performance for users. What is crucial is that LB-BC has long-term potential benefits of optimizing external service capabilities and resource utilization from the cloud ends perspective.

2) Introduce the model of Bayes theorem on optimal deployment of tasks to work out the probabilities of optimal physical hosts.

3) Propose the cluster-based task deployment algorithm by combining with Bayes probabilities to obtain the optimal set of candidate physical hosts.

4) Propose a data structure of Matrix to determine the final solution vector in each algorithm cycle.

3 The Proposed Problem and Its Formulation

3.1 Proposed Problem

In IaaS cloud data centers, the system will deploy tasks on the physical hosts in the resource pool of the cloud data center when users submit task requests. In general, the cloud data center will choose the physical hosts at random to deploy tasks. When the resource amount requested by a task is greater than the remaining resource amount of the physical host, the physical host cannot deploy the task. And when the requested resource amount is close to the remaining amount of the physical host, the physical host will have a heavy workload, causing the decline of its service capacity and computing power. The decline will result in load imbalance of the cloud data center and make service efficiency fall. Obviously, different task deployment strategies may lead to different load allocation of the whole system and may cause different execution efficiency and external service capability. Fig. 1 gives an example of task deployment in the IaaS cloud data center. There is no doubt that the optimal task deployment strategy should have abilities to make the whole cloud computing system have the better load balancing effect. Therefore, it is necessary to design and implement an efficient and load-balancing task deployment strategy in cloud data center.

3.2 Problem Formulation

The problem of task deployment is formalized as follows: there are n task requests accumulated in the time window

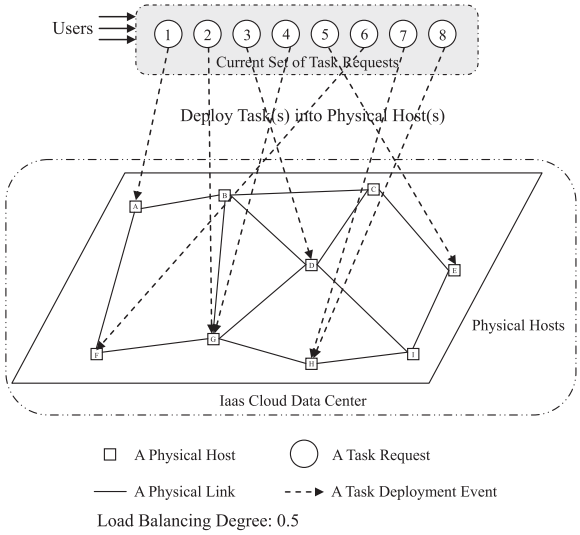


Fig. 1. Deploy tasks into physical hosts in the IaaS cloud data center.

Δt to be deployed into the cloud data center, which consists of m physical hosts. Its solution can be denoted as an n-dimensional solution vector $S = (s_1, s_2, \ldots, s_n)$, and its every element s_i stands for the optimal physical host which will deal with the corresponding task request i. The obtained solution vector S is the deployment strategy finally returned by LB-BC. It is used to determine which task will be deployed into which physical host. Cloud data centers having received the task deployment requests, the proposed LB-BC approach will work out the final deployment strategy obtained in the form of solution vector S by utilizing its algorithm mechanism. Then the cloud system will implement the deployment events according to the deployment strategy S. Assume that in the same network environment, the m physical hosts are heterogeneous and dynamic while using space shared allocation policy. Then our problem can be stated as follows: a physical hosts set with the optimal performance is found in the cloud data center such that the cloud data center can rapidly deploy tasks and execute the set of users' task requests. The problem of the cloud data center's load balancing can be solved through optimizing task deployment problem in every Δt time from a long-term perspective. We define a sextuple $Y = \{PH, TR, L_c, L_{mem}, R_c, R_{mem}\}$ to describe this problem scenario. PH represents the set of m available physical hosts $PH(m,t) = \{ph_1, ph_2, \ldots, ph_m\}$.$t$ represents the start time of task deployment. $TR(n,\Delta t,t) = \{tr_1, tr_2, \ldots, tr_n\}$ represents the set of users' task requests within a Δt time. L_c is the set of the current remaining CPU resource amount of m physical hosts in the set PH,$L_c(m,t)$=$\{L_c^1, L_c^2, \ldots, L_c^n\}$. L_{mem} is the set of the remaining memory resource amount of m physical hosts in the set PH at time 't', $L_{mem}(m,t) = \{L_{mem}^1, L_{mem}^2, \ldots, L_{mem}^m\}$. $R_c(n,t) = \{R_c^1, R_c^2, \ldots, R_c^n\}$ is the requested CPU resource amount of n task requests in TR. $R_{mem}(n,t) = \{R_{mem}^1, R_{mem}^2, \ldots, R_{mem}^n\}$ is the requested memory resource amount of n task requests in TR. To obtain the final solution vector, this paper has focused on finding out the final candidate set of optimal physical hosts. The set should have capabilities in fulfilling users task requests. It can not only execute requested tasks quickly within a certain time but also fast return information that users need. As a result, it has achieved load balancing of the whole network at the greatest extent in a long-term process. In order to meet the performance constraints of physical hosts, available remaining resource amount of a physical host L_i is defined as follows:

$$L_i = \alpha L_c^i + \beta L_{mem}^i \tag{1}$$

$$\alpha + \beta = 1. \tag{2}$$

In this paper, L_i is used to represent the remaining computing power of the physical host i; L_c is the remaining CPU resource; L_{mem} is the remaining memory resource; α is the weight value of CPU;β is the weight value of memory; the α and β values are determined and obtained by BP neural network learning. The remaining computing power of m physical hosts in the current cloud data center can be calculated according to formula (1) and formula (2). The performance constraint value can be defined as the maximum requested resource amount in the set TR, i.e.,

$$L_{Mreq} = \max_{i=1}^{n} R_i. \tag{3}$$

In this formula (3), R_i represents the requested resource amount of the ith task in task requests set. That is,

$$R_i = \alpha R_c^i + \beta R_{mem}^i. \tag{4}$$

The proposed LB-BC approach is aiming to achieve the long-term load balancing of cloud data centers. In this paper, we have utilized the standard deviation of all hosts' residual load rates to measure the load balancing degree of a cloud data center. The formulas of the expectation and the standard deviation are as follows:

$$E(X) = \frac{\sum_{i=1}^{N} X_i}{N} \tag{5}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - E(X))^2}. \tag{6}$$

The residual load rate of a host is defined as follows:

$$E_i = \frac{L_i}{T_i}, \quad i \in \{1, 2, 3, \ldots m\}, \tag{7}$$

where T_i represents the total computing power of host i, and it is defined as follows:

$$T_i = \alpha T_c^i + \beta T_{mem}^i, \quad i \in \{1, 2, 3, \ldots m\}, \tag{8}$$

where T_c^i denotes the total CPU resource amount of host i and T_{mem}^i denotes the total memory resource amount of host i. In this paper, it is thought that the situation that the residual load rate E_i of each host is as similar as possible can reflect the load balancing degree of a cloud data center. As a result, according to the above formulas, the proposed optimization objective can be represented as minimizing the following formula (9):

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{\alpha L_c^i + \beta L_{mem}^i}{\alpha T_c^i + \beta T_{mem}^i} - \frac{\sum_{k=1}^{m}\frac{\alpha L_c^k + \beta L_{mem}^k}{\alpha T_c^k + \beta T_{mem}^k}}{m}\right)^2}. \quad (9)$$

In this paper, the proposed LB-BC approach doesn't directly address the proposed problem by optimizing the above objective function but designs a framework and methods to achieve the better load balancing effect from the perspective of cloud data center's long-term operations.

In the proposed LB-BC approach,the physical hosts whose remaining resource amount is greater than the constraint value first constitute a dynamic and adaptive set as a new candidate set *NPH* of m' physical hosts. However, the set *NPH* can just be considered to be optimal on performance to some extent. In other words, there exist some unreasonable physical hosts which cannot be excluded from the physical hosts set *NPH* currently. For example, if the remaining CPU resource of a physical host r is very large and its remaining memory is very small, its L_r value is also very large and even larger than the constraint value of the task requests set. However, in fact, the physical host doesn't have ability to execute any one of these tasks obviously. In order to guarantee the computing performance of physical hosts used for executing requested tasks and exclude unreasonable physical hosts,we have utilized the clustering idea to search out the optimal clustering of physical hosts as the final candidate set of physical hosts for processing tasks from the set *NPH* by comparison of the similarity between physical hosts and a given threshold. The similarity function is defined as follows:

$$SD(nph_i, nph_j) = \frac{1}{\sqrt{\sum_{k=1}^{d}(a_i^k - a_j^k)^2}}, \quad (10)$$

where a_i^k and a_j^k respectively represents the kth attribute value of host i and host j. LB-BC has utilized three attributes to achieve the clustering process. L_c^i and L_{mem}^i are used as the two attributes of them. The posterior probability of each physical host in *NPH* is obtained and chosen as the third attribute by using Bayes probability model. That is, we have made use of the posterior probability value of each physical host in *NPH* as one of its attribute values used in the clustering process. First, the prior probability of each physical host is calculated according to the remaining resource amount of the physical host and the resource amount users requested. And then the posterior probability of each physical host is calculated on the basis of Bayes theorem. The formulas are as follows:

$$P(B_i|A) = \frac{P(A|B_i) * P(B_i)}{P(A)} \quad (11)$$

$$P(A) = \sum_{i=1}^{m'} P(A|B_i) * P(B_i). \quad (12)$$

Through the formula (12), the formula (11) can be represented as follows:

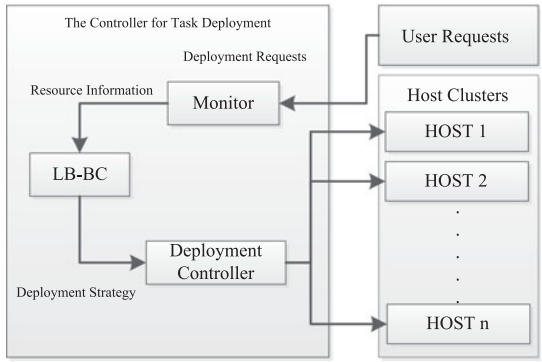


Fig. 2. The view of LB-BC's architecture.

$$P(B_i|A) = \frac{P(A|B_i) * P(B_i)}{\sum_{i=1}^{m'} P(A|B_i) * P(B_i)}. \quad (13)$$

$P(B_i|A)$ is the posterior probability of the physical host i.

Therefore, the proposed approach has optimized the physical hosts set used for deploying tasks and thus to furthest achieve load balancing of the cloud data center from cloud data centers' long-term operations point of view. The details will be given in the following.

4 A Heuristic Approach for Efficient Task Placement

4.1 Design of System Architecture

Fig. 2 describes the architecture of the proposed LB-BC approach in the cloud data center environment. It shows the interaction between LB-BC and other entities and that it plays a very important role in the whole architecture. First, the Monitor acquires the information of resource requested by users' tasks and the status information of remaining resource amount (including CPU and memory) of m available physical hosts in cloud data center. By using the information acquired from Monitor, LB-BC generates the deployment strategy, which is transmitted to Deployment Controller whose function is to control and carry out the deployment of requested tasks. In the end, the task requests accumulated within a Δt time are deployed to the corresponding physical hosts in the final optimal set of physical hosts obtained by the proposed LB-BC approach.

In this paper, the proposed LB-BC approach is a heuristic task deployment approach, which is used to deploy task requests received by the cloud data center into optimal target physical hosts in the IaaS cloud computing data center. Its algorithm is what combines Bayes theorem with clustering. It has achieved the overall load balancing of the entire network from the perspective of cloud data centers' long-term operations and thus to improve the performance and efficiency. The task deployment process of LB-BC is shown in Fig. 3. First, these physical hosts, each of which has a larger remaining resource amount than the maximum requested resource amount of all task requests, can be searched out to constitute a new candidate set to meet the performance constraint while making LB-BC have the potential of achieving the long-term load balancing. Second, the k physical hosts in the set of physical hosts can be regarded as k objects waiting for being clustered. Each

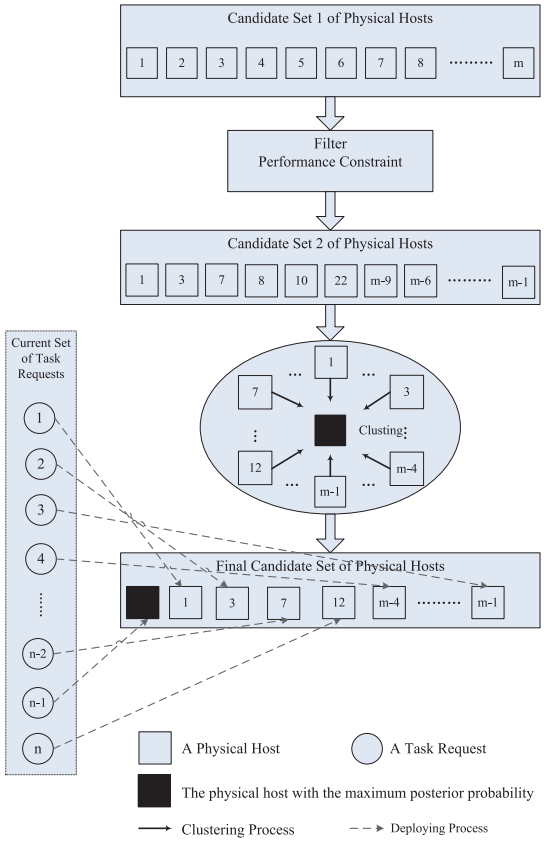


Fig. 3. The process of LB-BC task deployment.

physical host in the given set is given to a prior probability. The posterior probability of each physical host's handling tasks can be calculated through Bayes theorem. This probability can be regarded as an attribute of each object while the remaining CPU resource amount and the remaining memory resource amount of each physical host can be regarded as the other two attributes. The similarity degree values between physical hosts are calculated according to the three attributes of each physical host. A threshold value is determined on the basis of these similarity degree values. The physical hosts whose similarity degree values between them are within the given threshold can be seen as the optimal clustering to form the final set of candidate physical hosts. Finally, the tasks are placed on the hosts in the final set. And the clustering process of physical hosts in the cloud data center is the process of finding the optimal physical hosts for executing tasks.

4.2 Algorithm of LB-BC

In this section, we describe the specific process of LB-BC. Details are as follows:

Step 1: In IaaS cloud data centers, there are too large numbers of physical hosts. In order to avoid this situation that the selected physical hosts can't meet the resource requirements of requested tasks and to achieve the best clustering effect through minimizing the candidate set, we assume that there are m physical hosts in the cloud data center, and each physical host needs to be assigned to a constraint value for measuring its remaining available computing power in the cloud data center according to (1) and (2). The constraint value L_i of each physical host i in the cloud data center has been calculated. We define an empty set $NPH = \{\}$, and the performance constraint value of the set TR of task requests is defined as the maximum requested resource amount in NPH. The maximum requested resource amount L_{Mreq} of TR can be calculated in terms of formula (3). And if $L_i > L_{Mreq}$, host i will be placed into the set NPH. Having compared the constraint value L_i of each physical host with the performance constraint value L_{Mreq}, the new candidate set $NPH = \{nph_1, nph_2, \ldots, nph_{m'}\}$, $m' \leqslant m$ is obtained and it will be used as the candidate set of physical hosts for the following clustering process.

Step 2: These physical hosts with better performance in the cloud data center are put into the set NPH by the restricting of the constraint value of the requested tasks. However, we cannot ensure that each physical host from the set NPH can process any of the received task requests in TR since some physical hosts may have large amount of available memory resource while little CPU resource,or vice versa. Their L_i values are also very large, but they aren't what we want. Aiming to not only fully exploit the advantage (convenient and diverse) of the weighted sum of multiple kinds of resource but also overcome its disadvantage that there exist some unreasonable hosts, we have utilized the methods of probability theory and clustering to achieve the selection of optimal hosts while achieving a long-term load balancing. It is by picking out the optimal clustering of physical hosts with the relatively larger computing power to process these received tasks in TR that the objective of load balancing is achieved from a long-term perspective (i.e., too large numbers of Δts). Aiming to achieve a better clustering effect, the posterior probability of each physical host is introduced into the proposed LB-BC approach and used as an attribute of each object. And the posterior probability is obtained according to formula (13). Event A is defined as that those tasks are executed on some physical hosts. And event B_i is defined as the event that the physical host i is chosen. The ratio of the maximum requested resource amount L_{Mreq} of the tasks accumulated within a Δt time in the cloud data center and the remaining computing power L_i of the current physical host i in the set NPH is defined as the prior probability value of the current physical host i. L_{Mreq} is closer to L_i, and their ratio is closer to 1. The ratio value is contrary to what we want. The physical hosts which have more remaining resource should be more suitable for dealing with these tasks from the perspective of performance and load balancing. Therefore, in this paper the final prior probability of each physical host in the set NPH is set as follows:

$$P(A|B_i) = 1 - \frac{L_{Mreq}}{L_i}. \tag{14}$$

There are m' physical hosts in the set NPH, and the probability of choosing a physical host is

$$P(B_i) = \frac{1}{m'}. \tag{15}$$

We could obtain the formula (16) through formula (13), (14) and (15). $P(B_i|A)$ is the posterior probability of physical host i

$$P(B_i|A) = \{(L_i - L_{Mreq})L_1 \dots L_{i-1}L_{i+1}\}/\{m'L_1L_2 \dots L_{m'} - L_{Mreq}(L_2L_3 \dots L_{m'} + \dots + L_1L_2 \dots L_{i-1}L_{i+1} \dots L_{m'} + \dots + L_1L_2 \dots L_{m'-1})\}. \quad (16)$$

Step 3: The posterior probability value $P(B_i|A)$ of each physical host i in the set NPH has been obtained. Let P_i=$P(B_i|A)$. Now, each physical host i has three attributes including the posterior probability P_i, remaining CPU resource amount L_c and remaining memory resource amount L_m. m' physical hosts in NPH can be seen as m' objects of R^d space, wherein R^d represents a d-dimensional attribute space. In this paper, d is set to 3 obviously. The posterior probability values of the physical hosts in the set NPH are sorted descending. Let nph_j represent the physical host with the biggest posterior probability in NPH and thus nph_j is selected as the clustering center in this paper. The formula (17) to calculate the similarity degree value between any other host i from NPH and nph_j is as follows:

$$SD = \frac{1}{\sqrt{(P_i^1 - P_j^1)^2 + (L_{ci}^2 - L_{cj}^2)^2 + (L_{mi}^3 - L_{mj}^3)^2}}, \quad (17)$$

where $P_i^1 \neq P_j^1$, $L_{ci}^2 \neq L_{cj}^2$ and $L_{mi}^3 \neq L_{mj}^3$; otherwise, SD is assigned an infinite similarity value. P_j^1 is the first attribute value of physical host j and it is the posterior probability of physical host j. And L_{cj}^2 is the second attribute of physical host j, that is, it is the remaining CPU resource amount of physical host j. L_{mj}^3 is the third attribute of physical host j and it is the remaining memory resource amount of physical host j.

Step 4: The similarity degree values between nph_j and other objects in the set NPH are calculated in the case that nph_j is the clustering center. The threshold value $U_{threshold}^{Similarity}$ is given according to similarity degree values. If the similarity degree value SD is larger than the threshold value $U_{threshold}^{Similarity}$, then this object will become a member of the new set NPH'={}. The clustering center nph_j is the first member which is put into the set NPH'. The final candidate set is the final clustering result NPH', i.e., NPH'=$\{nph'_1, nph'_2, \dots, nph'_q\}$ ($q \leqslant m' \leqslant m$).

Step 5: These requested tasks will be deployed into the physical hosts in the set NPH'. Here, we have employed first input, first output (FIFO) processing order of task requests while the proposed LB-BC approach has deployed each task of TR into the physical host which currently has the maximum L_i value, aiming at further optimizing the performance and efficiency of the long-term load balancing. An example is given as shown in Fig. 4. And then the corresponding physical hosts in the set NPH' have executed the requested tasks in TR. The processing results will be responded to users after handling these task requests. The time from the beginning of the proposed LB-BC algorithm to having generated the final deployment strategy is set as the next time window Δt. In another word, during that period of time, the received task requests constitute the next round of task requests set TR. The number of task requests received during Δt time by the cloud data center is considered as the workload of the proposed LB-BC approach next time.

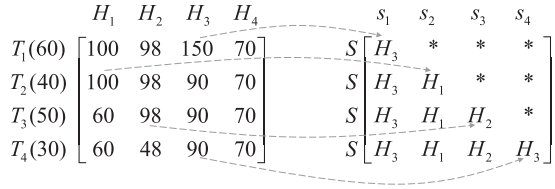


Fig. 4. An example of requested tasks' being deployed into the final optimal set.

Step 6: Repeat the above process.

The pseudocode of TA-GG's algorithm has been given as follows.

Algorithm 1. LB-BC(PH,TR,L_c,L_{mem},R_c,R_{mem})

Input: current PH, current L_c, current L_{mem}, received TR, received R_c, received R_{mem};
Output: final deployment solution vector S;

```
NPH = φ , NPH' = φ , S = NULL;
for each tr_i ∈ TR do
    R_i = α R_c^i + β R_mem^i;
end for
L_Mreq = max_{i=1}^{n} R_i;
for each ph_i ∈ PH do
    L_i = α L_c^i + β L_mem^i;
    if L_i > L_Mreq then
        NPH = NPH ∪ ph_i;
    end if
end for
for each nph_i ∈ NPH do
    the posterior probability P(B_i|A) of nph_i is obtained according to the formula (16);
end for
nph_j is the candidate physical host with P_j = max_{i=1}^{m'} P(B_i|A) in NPH;
NPH' = NPH' ∪ nph_j;
for each nph_i ∈ NPH do
    SD(nph_i, nph_j) is obtained according to the formula (17);
    if SD(nph_i, nph_j) > U_threshold^Similarity then
        NPH' = NPH' ∪ nph_i;
    end if
end for
for each tr_i ∈ TR do
    // Assume the n task requests of TR are sorted in terms of FIFO in advance.
    S[i] = the No. of the physical host nph'_i with the maximum remaining resource amount L_i in NPH';
    L_i = L_i - R_i;
end for
Return S;
```

From the microcosmic perspective, the process of calculating physical hosts posterior probability is the process of applying Bayes theorem. The reason why the process of calculating posterior probability values in LB-BC is in line with that of Bayes theorems application is that Bayes theorem provides effective means to amend the original judgment by using the collected information. Before sampling,

the subject has a judgment on every assumption and it is known as the prior probability value. We get the posterior probability value of each physical host in the set *NPH* by using Bayes theorem. First, event A and event B are defined, wherein the event of deploying some task into some physical host is event A, and the event of choosing some physical host is event B. In general, the event A's probability under the condition that event B happens is different from event B's probability under the condition that event A happens, that is to say, the chosen physical host isn't necessarily to be capable of executing the requested tasks. The physical host cannot process requested tasks surely if the requested resource amount of tasks is greater than the remaining resource amount of a physical host. Therefore, the prior probability value of each physical host depends on requested resource amount of tasks and its remaining resource amount. When the requested resource amount of tasks is close to the remaining resource amount of the physical host, the probability of placing the requested task into this physical host is small. Thus we have defined and presented the calculating formula (18) of the prior probability value

$$P(A|B_i) = 1 - \frac{MRRA}{RRA_i}, \tag{18}$$

where *MRRA* represents the maximum requested resource amount of all received tasks and RRA_i denotes the remaining resource amount of physical host i.

$P(B_i)$ is the reciprocal of the number of physical hosts in the set *NPH* and it means that the probability of choosing any host is equal. And the posterior probability value of each physical host is obtained according to formula (16), which is in accordance with the application process of Bayes theorem.

From the macro point of view, the process of the proposed LB-BC approach is the process that the clustering algorithm finds the best clustering. We have employed the euclidean distance value as the evaluating indicator of similarity degree. That is, this paper considers that any two objects are closer and their similarity degree is larger. LB-BC considers that a clustering is made up of some objects between which the distances are very short. Thus, the final goal is to get the compact and separate clustering. First of all, k objects are selected from n objects as original clustering centers. The rest of objects are distributed respectively to their most similar clustering that are represented by the corresponding clustering center according to their similarity degree or their distances with these clustering centers. And then each clustering center of acquired new clusterings is re-calculated. A clustering center is the mean of this clustering's all objects. This process is repeated until the standard measurement function starts to converge. The mean square error is always taken as the standard measurement function. The clustering idea has the following features. In the same cluster these members are as compact as possible. Between the different clusters, their members are separating with each other as much as possible.

The reason why the proposed LB-BC algorithm is in accordance with the idea of clustering algorithms is that it takes the distance as the evaluating indicator of the similarity degree. That is, the closer the distance between two objects, the greater the similarity degree. The LB-BC algorithm considers physical hosts as objects and the distance as the attribute of each object, which means that each physical host's posterior probability, remaining CPU resource amount and remaining memory resource amount are regarded as the evaluating indicators of the similarity degree and the similarity degree values between physical hosts are worked out by using the similarity degree formula of euclidean distance. The physical host with the largest posterior probability is chosen as the clustering center. The similarity degree values between each physical host with the clustering center are calculated. And then a threshold value is given. When a similarity degree value is greater than the threshold value, the corresponding physical host will become a member of the new set. The unbiased variance is used as the standard measurement function to assess the performance of the clustering process. When the standard measurement function value decreases gradually and is converged to a certain point, the clustering result and the performance of the physical hosts set are the best.

From the whole, the process of LB-BC algorithm is also in line with heuristic algorithms. This is mainly because the solution found by an algorithm with the heuristic approach is not necessarily to be the best one in one search. However, by continuous seeking and amending, it will be closer and closer to the final optimal solution. From the perspective of the algorithms goal, it is aiming to achieve the whole networks load balancing of the cloud data center. Load balancing cannot be achieved through the just once executing of the proposed LB-BC algorithm. It needs a long-term process. The optimal physical hosts set can be found in each time of executing and it makes the optimal physical hosts in the current cloud data center deal with the tasks requested by users. Thus, by doing this, the picked out optimal physical hosts are those which are close to the best performance through the LB-BC algorithm, and tasks requested by users can be handled quickly by the cloud data center. The LB-BC algorithm ensures that a better result can be obtained within a reasonable time in this process.

5 Performance Evaluation

In this section, the DLB deployment approach, the random deployment (RD) approach, and the proposed LB-BC deployment approach are compared as following aspects:

1) MakeSpan
2) Standard deviation values measuring load balancing effect
3) Throughput measuring external service performance
4) Failure number of task deployment events
5) Incremental percentage of their standard deviation values

In this paper, we have employed the CloudSim simulator [28], which allows such simulation scenarios by supporting dynamic creation of different kinds of entities and can add and remove data center entities at run-time. In the CloudSim platform, this paper has compared the proposed LB-BC deployment strategy with the random deployment strategy and the optimal deployment strategy (DLB). They are

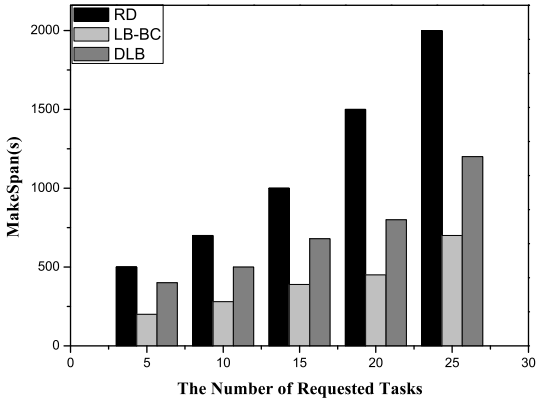


Fig. 5. Comparison of Random-Migration, DLB, and LB-BC on MakeSpan.

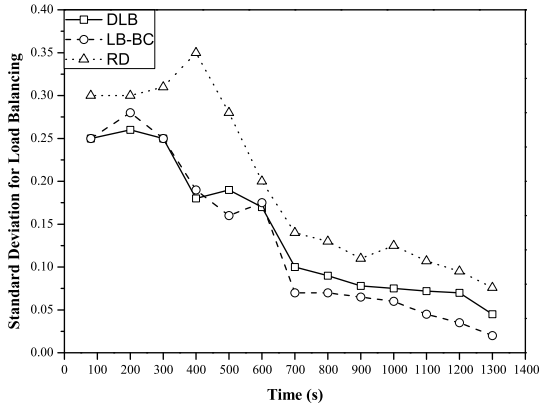


Fig. 6. Comparison of Random-Migration, DLB and LB-BC in load balancing.

evaluated and tested through five different sets of simulation experiments. The final simulation results have shown that the proposed LB-BC approach can not only make the failure number of deployment tasks lower than that of DLB, but also make the cloud data center have a better load balancing effect. Compared with other deployment approaches, LB-BC has a better stability and efficiency in the aspect of achieving the whole load balancing of cloud data centers, especially for deploying a large amount of continuous task requests.

5.1 Experimental Scenarios

In the CloudSim platform, in order to achieve better simulation performance while obtaining the satisfactory experimental results, 100 physical hosts with different configuration are deemed as the minimum scale of physical hosts, which can efficiently fulfill our requirements under the current simulation conditions. Thus, a resource pool with 100 physical hosts is created, and these physical hosts have different available computing resource. Twenty-four batches of task requests containing 50 requests continuously reaching the cloud data center and having different resource requirements are created. The proposed LB-BC module is invoked and fetches the resource information and state of the cloud resource pool regularly. The initial Δt is set to include 15 task requests.

5.2 Comparison on MakeSpan

In the experimental scenario, LB-BC is compared with the RD approach and the DLB approach on MakeSpan, which is the needed time of processing tasks set. The experimental results of the three approaches are shown as Fig. 5. The MakeSpan values of requested tasks set increase along with the number of the requested tasks increasing. The RD approach has deployed the requested tasks at random on the physical hosts in the cloud data center essentially. On this occasion, with the increase of the number of requested tasks, the ability of handling tasks will weaken gradually. Therefore, the needed time will also increase for sure. And the DLB deployment approach just predicts requested tasks' resource amount on the basis of its knowledge repository in the beginning and will trigger task deployment events according to the revenue value of load balancing. With the increase of the number of requested tasks, the communication cost between physical hosts will increase surely. The ability of handling tasks will weaken gradually, and the time needed will increase. However, it is smaller than that of the RD approach. The proposed LB-BC approach will select the optimal physical hosts set to deal with tasks in each round of iterations, and deploy tasks into the corresponding hosts. It not only avoids a large amount of communication cost, but also guarantees physical hosts' computing performance. The time of handling tasks will increase with the increase of the number of requested tasks. It is smaller than that of the DLB deployment approach and the RD approach with the same number of requested tasks. Compared with the RD approach and the DLB approach, LB-BC has smaller MakeSpan value in the same condition, which can be shown from Fig. 5. This experiment illustrates that LB-BC has not only a better load balancing effect but also a better MakeSpan value relatively.

5.3 Comparison on Load Balancing Effect

In this set of experiments, the changes of DLB and LB-BC's load balancing effect in the cloud data center with time changing are compared. Here, we have employed the standard deviation value indicated above and used to measure the degree of load balancing to conduct the set of experiments. Obviously, a smaller standard deviation value represents that the cloud data center has the better balancing of load. With the increase of the time for the deployment process in the cloud data center, the standard deviation value of the traditional RD strategy is always greater than those of the two definite deployment strategies, as can be seen in Fig. 6. And the standard deviation values of the DLB and LB-BC deployment strategies are gradually decreasing. DLB's standard deviation value is smaller than that of LB-BC in the beginning. When $t = 600$, the two standard deviation values are equal. However, when $t \geq 700$, the decline degree of LB-BC's standard deviation value is always greater than DLB's. This is because that the proposed LB-BC approach could find the optimal physical host quickly according to the remaining computing power. It ensures that the load balancing effect is achieved through making the optimal hosts with relatively more remaining computing power have the priority of executing requested tasks. While the DLB approach makes a prediction of each processor's load first and deploys tasks in the light of knowledge repository. It doesn't deal with the real-time workload

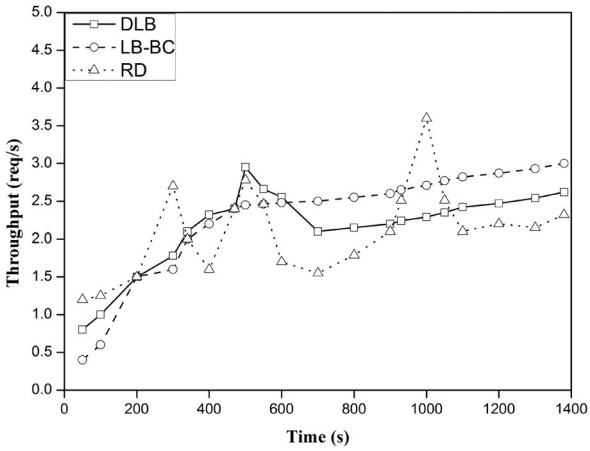


Fig. 7. Comparison of Random-Migration, DLB, and LB-BC in external service performance.

information of physical hosts and can't also select the optimal physical hosts for deploying tasks effectively. According to the results of this set of experiments, the proposed LB-BC approach has a better effect of load balancing and thus to improve the resource utilization of the cloud data center effectively.

5.4 Comparison on External Service Performance

In the experiment scenario, the proposed LB-BC approach is verified by comparing the external service performance of the cloud data center respectively implementing the three deployment approaches with time increasing. The throughput rate is chosen as the evaluation standard to measure the external service performance since it can represent the comprehensive evaluation of cloud systems, such as components ability of dealing with tasks, the transmitting ability of data and the ability of responding task requests to users etc. The experimental results are shown as Fig. 7. And it can be seen that the external service performance of the cloud data center is different by using three different deployment approaches. When users have access to the cloud data center, the computing performance of the cloud data center implementing RD is relatively better. However, with time increasing, its external service performance is not stable and has a waving mode. Also, the external service performance of DLB is relatively better than that of LB-BC at the initial time. With time increasing, When $t = 700$, the external service performance of the cloud data center implementing DLB becomes stable gradually. However, its throughput rate is always smaller than that of the proposed LB-BC approach. Whereas in the cloud data center implementing LB-BC, although its throughput rate is smaller than those of RD and DLB at the initial time, its external service performance gradually tends to be stable and be higher than that of DLB with time increasing. This is because the physical hosts in the set chosen by LB-BC have meet the demand of deploying tasks to the greatest extent while DLB have deployed tasks according to the information of knowledge repository and predicted load condition of each physical host. Then the tasks may be re-deployed into physical hosts by DLB in the light of load condition. To sum up, the conclusion that the LB-BC approach has better stability and efficiency can be made.

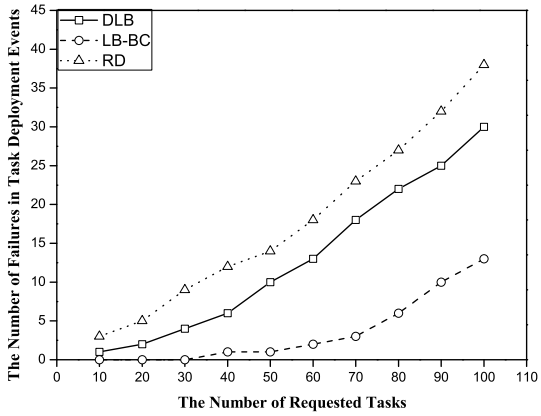


Fig. 8. Comparison of the number of failures in task deployment events.

5.5 Comparing on the Failure Number of Task Deployment Events

In this set of experiments, the dynamic host failure is simulated by CloudSim by scheduling some host failure events and host shut-down events to occur during deploying tasks. When some tasks are deployed, the task deployment events are likely to fail since the chosen physical hosts cannot fulfill some the requirements of the requested tasks. We have compared the proposed LB-BC approach with the RD approach and the DLB approach on the failure number of deploying tasks. As shown in Fig. 8, in the simulated cloud data center, with the increasing of requested tasks, the failure numbers of RD and DLB are more and more. However, in the cloud data center implementing LB-BC, with the requested tasks increasing, the failure number of deploying tasks is increasing very slowly and is always lower than those of RD and DLB. This is because DLB just deploys tasks into physical hosts according to the knowledge repositorys experiment values. It cannot acquire the information of each physical hosts remaining resource in the resource pool in real time. When the requested resource amount of requested tasks is greater than the remaining resource amount of a physical host, the deployment event will fail. On the contrary, LB-BC has deployed tasks from a long-term perspective. The remaining resource amount of each physical host chosen by LB-BC would be sure to be greater than the maximum requested resource amount of tasks when it deploys tasks every time. Thus, the proper physical hosts of most requested tasks could be found in the resource pool dynamically and adaptively. And the LB-BC approach picking out most proper physical host has reduced the failure number of deploying tasks to a certain extent.

5.6 Comparison on the Overall Load Balancing with Varying Number of Task Deployment Events in the Cloud Data Center

In this experimental scenario, with the number of task requests increasing, we have verified the load balancing effect of DLB and LB-BC by comparing their percentages of the incremental standard deviation values. These received requests constitute the total workload to be handled by the cloud data center and aren't the load of some physical host. As illustrated in Fig. 9, with the increase of the requested tasks, the percentage of load in the cloud data center

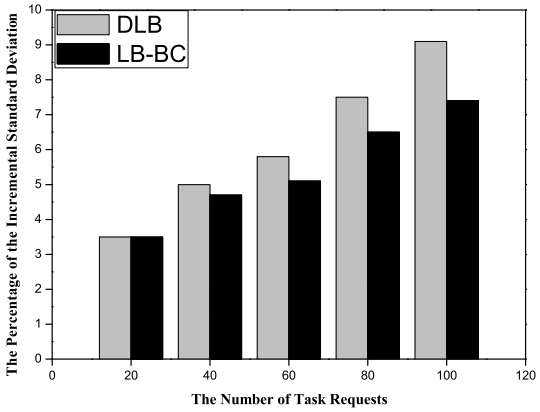


Fig. 9. Comparison of the incremental standard deviation with varying number of task deployment events in the cloud data center.

increases and the load balancing effect of LB-BC is better than that of DLB. This is because LB-BC deploys requested tasks from the globally optimal perspective and the DLB approach deploys tasks from the locally optimal perspective. LB-BC has ability to deploy every requested task on the optimal physical host quickly and to ensure the overall load balancing of the cloud data center from a long-term perspective. However, DLB just aims to solving this problem locally optimally and it deploys tasks after calculating the revenue value of workload of each processor. When the number of the requested tasks increases, since DLB doesn't have ability to pick out the optimal physical host for the current task and there exists frequent communication between processors, it results in the relatively worse overall load balancing effect in the current cloud data center. To sum up, the conclusion that LB-BC makes the cloud data center, especially for large-scale cloud data centers, have a better load balancing effect can be made. Besides, as can be noticed, RD isn't included in the experiment. RD doesnt have any heuristic information and adaptive abilities. It just deploys tasks randomly in cloud data centers. Thus, it has little significance to compare RD in the experiment scenario.

6 Conclusion and Future Work

This paper has proposed a task deployment approach LB-BC for the long-term load balancing effect and it has employed a heuristic idea based on Bayes theorem and the clustering process. LB-BC first has narrowed down the search scope by comparing performance values. Then, LB-BC has utilized Bayes theorem to obtain the posteriori probability values of all candidate physical hosts. Finally, LB-BC has combined probability theorem and the clustering idea to pick out the optimal hosts set, where these physical hosts have the most remaining computing power currently, for deploying and executing tasks by selecting the physical host with the maximum posteriori probability value as the clustering center and thus to achieve the load balancing effect from the long-term perspective. Simulation experiments demonstrate that the proposed LB-BC approach can deploy the instant tasks quickly and effectively in cloud data centers. It makes cloud data centers achieve a long-term load balancing of the whole network.

LB-BC should have the ability to select one physical host or more in the set obtained by the clustering approach to deploy tasks and thus to make the benefit acquired by users and cloud data centers maximized at the same time. Besides, the proposed LB-BC approach only applies to LAN. We will extend LB-BC to WAN in the further research. A larger scale of simulation experiments and prototype experiments are one of our next research work. We will attempt to implement LB-BC in a real cloud computing environment and evaluate its performance and efficiency as well as verify its load balancing effect of cloud data centers.

Acknowledgments

This work was supported by the National Science-Technology Support Project (2014BAH02F02), the National Natural Science Foundation of China (Grant No. 61133011), the Natural Science Foundation of Jilin (Grant No. 20101533), UK EPSRC Project NIRVANA (EP/L026031/1), EU FP7 Projects MONICA (GA-2011-295222), CLIMBER (GA-2012-318939) and CROWN (GA-2013-610524). G. Xu is the corresponding author.

References

[1] T. Erl, R. Puttini, and Z. Mahmood, *Cloud Computing: Concepts, Technology & Architecture*. 1st ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2013.

[2] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Comput. Syst.*, vol. 29, no. 4, pp. 1012–1023, Jun. 2012.

[3] G. Nan, Z. Mao, M. Li, Y. Zhang, S. Gjessing, H. Wang, and M. Guizani, "Distributed resource allocation in cloud-based wireless multimedia social networks," *IEEE Netw. Mag.*, vol. 28, no. 4, pp. 74–80, Jul. 2014.

[4] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, "Toward cloud-based vehicular networks with efficient resource management," *IEEE Netw. Mag.*, vol. 27, no. 5, pp. 48–55, Sep./Oct. 2013.

[5] G. Xu, Y. Ding, J. Zhao, L. Hu, and X. Fu, "A novel artificial bee colony approach of live virtual machine migration policy using Bayes theorem," *Sci. World J.*, vol. 2013, no. 2013, p. 369209, Sep. 2013.

[6] J. Zhao, L. Hu, Y. Ding, G. Xu, and M. Hu, "A heuristic placement selection of live virtual machine migration for energy-saving in cloud computing environment," *PloS One*, vol. 9, no. 9, p. e108275, Sep. 2014.

[7] L. Liu, H. Wang, X. Liu, X. Jin, W. He, Q. Wang, and Y. Chen, "GreenCloud: A new architecture for green data center," in *Proc. 6th Int. Conf. Ind. Session Autonomic Comput. Commun. Ind. Session*, Jun. 2009, pp. 29–38.

[8] Q. Wei, G. Xu, and Y. Li, "Research on cluster and load balance based on Linux virtual server," in *Proc. Inf. Comput. Appl.*, 2011, pp. 169–176.

[9] Y. Di, S. Wang, and X. Sun, "A dynamic load balancing model based on negative feedback and exponential smoothing estimation," in *Proc. 8th Int. Conf. Autonomic Auton. Syst.*, Mar. 2012, pp. 32–37.

[10] W. Chen, Y. Zhang, and Z. Xiong, "Research and realization of the load balancing algorithm for heterogeneous cluster with dynamic feedback," *J. Chongqing Univ.*, vol. 33, no. 2, pp. 2–14, 2010.

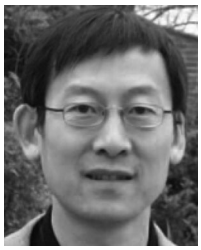
[11] S. Song, T. Lv, and X. Chen, "Load balancing for future internet: An approach based on game theory," *J. Appl. Math.*, vol. 2014, no. 2014, p. 959782, Feb. 2014.

[12] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Proc. 9th IEEE/ACM Int. Symp. Cluster Comput. Grid*, 2009, pp. 124–131.

[13] M. H. Willebeek-LeMair and A. P. Reeves, "Strategies for dynamic load balancing on highly parallel computers," *IEEE Trans. Parallel Distrib.*, vol. 4, no. 9, pp. 979–993, Sep. 1993.

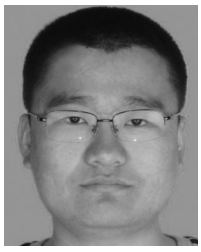
[14] T. You, W. Li, Z. Fang, H. Wang, and G. Qu, "Performance evaluation of dynamic load balancing algorithms," *TELKOMNIKA Indonesian J. Electr. Eng.*, vol. 12, no. 4, pp. 2850–2859, 2014.
[15] J. M. Bahi, S. Contassot-Vivier, and R. Couturier, "Dynamic load balancing and efficient load estimators for asynchronous iterative algorithms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 4, pp. 289–299, Apr. 2005.
[16] I. Pardines, and F. F. Rivera, "Minimizing the load redistribution cost in cluster architectures," in *Proc. 12th Euromicro Conf. Parallel, Distrib. Netw.-Based Process.*, 2004, pp. 326–331.
[17] K. Xu, Y. Zhang, X. Shi, H. Wang, Y. Wang, and M. Shen, "Online combinatorial double auction for mobile cloud computing markets," in *Proc. IEEE Int. Perform. Comput. Commun. Conf.*, 2014, pp. 1–8.
[18] S. M. Lau, Q. Lu, and K. S. Leung, "Adaptive load distribution algorithms for heterogeneous distributed systems with multiple task classes," *J. Parallel Distrib. Comput.*, vol. 66, no. 2, pp. 163–180, 2006.
[19] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[20] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Key challenges in cloud computing: enabling the future internet of services," *IEEE Int. Comput.*, vol. 17, no. 4, pp. 18–25, Jul./Aug. 2013.
[21] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu, "VMware distributed resource management: Design, implementation and lessons learned," *VMware Tech. J.*, vol. 1, no. 1, pp. 45–64, Mar. 2012.
[22] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Proc. 9th IEEE Int. Conf. Grid Cooperative Comput.*, 2010, pp. 87–92.
[23] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra, "Starling: Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration," in *Proc. 39th IEEE Int. Conf. Parallel Process.*, Sep. 2010, pp. 228–237.
[24] V. Shrivastava, P. Zerfos, K. W. Lee, H. Jamjoom, Y. H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *Proc. IEEE Conf. Comput. Commun.*, 2011, pp. 66–70.
[25] M. Rahman and P. Graham, "Hybrid resource provisioning for clouds," *J. Phys.: Conf. Ser.*, vol. 385, no. 1, p. 012004, 2012.
[26] C. Dupont, G. Giuliani, F. Hermenier, T. Schulze, and A. Somov, "An energy aware framework for virtual machine placement in cloud federated data centres," in *Proc. 3th IEEE Int. Conf. Future Energy Syst.: Where Energy, Comput. Commun. Meet (e-Energy)*, 2012, pp. 1–10.
[27] J. Zhao, Y. Ding, G. Xu, L. Hu, Y. Dong, and X. Fu, "A location selection policy of live virtual machine migration for power saving and load balancing," *Sci. World J.*, vol. 2013, no. 2013, p. 492615, Sep. 2013.
[28] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw.: Practice Experience*, vol. 41, no. 1, pp. 23–50, 2011.

Jia Zhao received the master's degree in 2008. He received the PhD degree from the College of Computer Science and Technology, Jilin University in 2013. He began his research on distributed system in 2005. Currently, he is with the College of Computer Science and Engineering, Chang-Chun University of Technology, China. His main research interests include distributed system, cloud computing, network technology. He has participated in several projects.



Kun Yang received the PhD degree from the Department of Electronic and Electrical Engineering, University College London (UCL), United Kingdom. He is currently a full professor and the head of Network Convergence Laboratory (NCL) in the School of Computer Science and Electronic Engineering, University of Essex, United Kingdom. Before joining in the University of Essex at 2003, he worked at UCL on several European Union Research Projects such as FAIN, MANTRIP, and CONTEXT. His current major research interests include heterogeneous wireless networks, fixed mobile convergence, and cloud computing. He has published more than 150 journal and conference papers in the above areas. He serves on the editorial boards of both IEEE and non-IEEE journals. He is a senior member of the IEEE and a fellow of the IET.

Xiaohui Wei is a professor and the dean in the College of Computer Science and Technology (CCST), Jilin University. He is currently the director of High Performance Computing Center, Jilin University. His current major research interests includes resource scheduling for large distributed systems, infrastructure level virtualization, large-scale data processing system and fault-tolerant computing. He has published more than 50 journal and conference papers in the above areas. He is a member of the IEEE.



Yan Ding received the bachelor's degree in 2011. He received the master's degree from the College of Computer Science and Technology, Jilin University in 2014. He began the study of computer science at Jilin University in 2007. His main research interests include distributed system, cloud computing, mobile cloud computing, the Internet of things and virtualization technology etc.

Liang Hu received the BS degree in computer systems from the Harbin Institute of Technology in 1993, and the PhD degree in computer software and theory in 1999. Currently, he is the professor and PhD supervisor of the College of Computer Science and Technology, Jilin University, China. His main research interests include distributed systems, computer networks, communications technology and information security system, etc. As a person in charge or a principal participant, he has finished more than 20 national, provincial, and ministerial level research projects of China.

Gaochao Xu received the BS, MS, and PhD degrees from the College of Computer Science and Technology, Jilin University in 1988, 1991, and 1995, respectively. Currently, he is the professor and PhD supervisor at the College of Computer Science and Technology, Jilin University, China. His main research interests include distributed system, grid computing, cloud computing, Internet of things, information security, software testing, and software reliability assessment, etc. As a person in charge or a principal participant, he has finished more than 10 national, provincial, and ministerial level research projects of China.