J Supercomput (2015) 71:2748–2766
DOI 10.1007/s11227-015-1416-x

Dynamically hierarchical resource-allocation algorithm in cloud computing environment

Zhanjie Wang[1] · **Xianxian Su**[1]

Published online: 3 April 2015

Abstract The development of big data challenges the computing power and communication capability of cloud architecture, but traditional resource-allocation algorithms perform poorly due to the large-scale communication among cloud nodes. In this paper, a dynamically hierarchical, resource-allocation algorithm is proposed for multiple cloud nodes collaborating in big data environment. Using fuzzy pattern recognition, the algorithm dynamically divides tasks and nodes into different levels based on computing power and storage factors. Thus a dynamically adjusted mapping is generated between tasks and nodes. When a new task arrives, only the nodes corresponding to the task level join in the bid. The algorithm takes advantages of dynamical hierarchy to reduce the communication traffic during resource allocation. Both theoretical and experimental results illustrate that the proposed algorithm outperforms the MinMin algorithm in terms of communication traffic and makespan.

Keywords Cloud computing environment · resource-allocation · Task level · Fuzzy pattern recognition

1 Introduction

Cloud computing [1] and big data [2,3] are the blistering issues in the modern day. Many researchers have done a lot of research and gained some achievements. Cloud computing is a computing prototype, which serves cloud users by providing comput-

✉ Xianxian Su
sxxhr0302@gmail.com

Zhanjie Wang
wangzhj@dlut.edu.cn

[1] Department of Computer Science, Dalian University of Technology, Dalian 116024, China

ing resources through a network. A cloud contains a parallel and distributed system interconnecting computing resources or virtual computing resources [4]. There are many popular cloud computing environments, such as Amazon EC2 [5] and IBM Smart Cloud [6]. In cloud computing environments, the users of the cloud service consume cloud resources as a service and pay for the service they use. The service is based on the service level agreement (SLA), which specifies the quality of service (QoS) between a service provider and a service consumer [7,8]. SLA usually includes the service price, with the level of QoS adjusted by the price of the service. The typical pay-per-use model reduces the initial cost of enterprises, thus most of them adopt clouds.

Recently, the rapid development of terminal devices led to certain amount of data surge into the Internet. Petabyte-scale collections of data come from click streams, transaction histories, sensors [9,10]. These data must be processed quickly, which has become an immense challenge to the computing power of the supercomputer and cloud architecture [11,12]. Thus, some cloud providers deploy geographically distributed data centers, or take some heterogeneous computer cluster as the nodes of cloud. These new cloud architectures insistently obliged an efficient resource-allocation algorithm [13,14].

Resource allocation in cloud computing environment is responsible for assigning the available resources to a set of tasks which form the workload. The resource-allocation algorithm contains some protocols and policies that can choose the most suitable resources and improve system performance [15]. Concerning resource-allocation problem, immediate policy and best effort policy [16] are typical in cloud environment. In recent years, some new heuristic algorithms such as deadline-drive [16–18] and energy-aware [19–21] have been proposed to improve the traditional algorithms, and have made certain progress. The resource-allocation algorithms which executed with virtual machines also have achieved better system efficiency [22–24].

However, most algorithms only consider a single factor, either the task execution time or communication traffic [25]. Therefor some multi-objective algorithms [26,27] are proposed to improve system performance in the distributed systems. These multi-objective algorithms integrate multiple factors such as computing power, workload and storage capacity. Furthermore, these algorithms still have limitations with regard to the dynamical changes of resources.

However, in the process of allocating resources, the node which generates tasks needs to notify all the other nodes and collect their information. Thus, a large number of communication messages burst out in a short time and as results it will increase the communication traffic, complexity and decrease the system performance severely. To solve this problem, we divide tasks and nodes into different levels to reduce the nodes number during the bidding process. In order to assign tasks to different levels, we compute task levels by using the fuzzy pattern recognition method [28,29]. Fuzzy pattern recognition is based on fuzzy information of an object in accordance with the principle of fuzzy mathematics [30]. The direct fuzzy pattern recognition method uses fuzzy sets to describe pattern type based on the principle of maximum membership degree. It aims at determining the mapping between the pattern and the task. In recent years, this technology is applied to task scheduling and resource allocation [31,32] in distributed systems [33], grid computing [34] and cloud computing [35]. After using

fuzzy pattern recognition it successfully stabilizes the result of task leveling, leading to a robust result. The robustness contributes towards simplifying the subsequent resource-allocation process.

In this paper, we propose a dynamically hierarchical, resource-allocation algorithm (DHRA) to solve the problem of a colossal number of messages generated during resources allocation. DHRA divides tasks into different levels through the method of fuzzy pattern recognition and heaves task information into the corresponding level in the task pool. Thus any nodes with idle ability choose tasks from the corresponding level of the task pool. That is to say only the nodes that are corresponding to the level of target task and having idle ability are able to join the bidding. Then the number of messages is reduced effectively, and communication traffic will also be reduced. Each node in DHRA has independent decision-making ability. The proposed algorithm not only increases system reliability, but also meets large-scale application service demand in cloud computing environment. By analyzing and experimenting with DHRA, the feasibility and effectiveness of DHRA can be proved and communication messages and communication traffic can be reduced.

In Sect. 2, we review the related works. In Sect. 3, we introduce the system framework. In Sect. 4, we describe DHRA algorithm. DHRA contains the level of task, the dynamic mapping between task level and nodes and the bidding process. In Sect. 5, we analyze DHRA and prove that DHRA can reduce the number of messages significantly. In Sect. 6, we prove DHRA can reduce the communication traffic and makespan more effective than MinMin algorithm by related experiments. We reach our final conclusion in Sect. 7.

2 Related work

On the verge of a big data era, the cloud computing system has become one of the most crucial research issues. The large-scale data center has helped relieve this issue to a certain degree. In the meantime, there has also been a considerable amount of research conducted using software approaches, such as resource allocation and task scheduling [16–24,26,27].

The deadline-drive algorithms are popular. In [16], the authors propose dynamic planning based scheduling algorithm, which supports deadline sensitive leases in Haizea while minimizing the total number of leases rejected by it. They find multiple slots in addition to finding a single slot while scheduling a deadline sensitive lease. It also applies two concepts (swapping and backfilling) in addition to preemption, while rescheduling already accommodated leases to make space for a newly arrived lease. Researchers in [17] present Aneka's deadline-driven provisioning mechanism. This allocation decision is driven by the application QoS, which is expressed in terms of the deadline for application completion. The deadline-driven policy considers the time left for the deadline and the average execution time of tasks that compose an application to determine the number of resources required. Similarly, in [18], deadline and budget constrained cost-time optimization algorithm is proposed in order to rely on sub tasks with considerations of communications between each other.

The energy-efficient algorithms are also a research hotspot. In [19], Anton et al. propose architectural principles for energy-efficient management in clouds. Further they have proposed energy-efficient resource-allocation policies and scheduling algorithms considering QoS expectations and power usage characteristics of the devices. It improves energy efficiency of the data center, while delivering the negotiated QoS. In [20], two energy-conscious task consolidation heuristics also have been presented. These heuristics assign each task to the resource on which the energy consumption for executing the task is explicitly or implicitly minimized without the performance degradation of the task. Wu et al. [21] propose a green energy-efficient scheduling algorithm using the DVFS technique for cloud computing data centers. The scheduling algorithm searches available servers to create VMs for allocating a job under the requirements of the job and selects the solution that consumes the least energy. These energy scheduling algorithm all efficiently increase resource utilization; hence, they can decrease the energy consumption for executing jobs.

Some researchers try to deal with resource allocation by using virtual machine. Xiao et al. [22] propose that uses virtual technology to allocate data center resources dynamically. They introduce the concept of "skewness" to measure the unevenness in the multi-dimensional resource utilization of a server. Daniel and Odej [23] introduce project Nephele, which explicitly exploits the dynamic resource allocation offered by today's IaaS clouds for both task scheduling and execution. Particular tasks of a processing job can be assigned to different types of virtual machines, which are automatically instantiated and terminated during the job execution. A new proactive workload management model [24] proposed for virtual resources to inspect the workload behavior of the running virtual machines, and to assert an appropriate scheduling and resource consolidation scheme in order to improve the system efficiency, utilization, and throughput.

Some multi-objective algorithms are proposed. In [26], researchers design a multi-agent based task scheduling mechanism, and schedule tasks according to multiple factors, such as task execution time and communication traffic. In [27], they study the multi-agent mechanism further, and discuss the weight of each factor. These multi-objective scheduling algorithms have better intelligence and stability, while the system efficiency and resource utilization are also improved.

3 System framework

DHRA is a hierarchical algorithm to reduce the number of messages and communication traffic in cloud computing environment. It divides tasks and nodes into different levels and brings out a dynamic mapping between tasks and nodes. Therefore the range of target nodes shrinks from all nodes to the nodes of the corresponding level in DHRA. Then the number of messages is reduced, meaning the reduction of communication traffic.

The system framework consists of: a task pool, nodes and the resource-allocation algorithm named DHRA. The system framework is shown in Fig. 1.

As shown in Fig. 1, the upper part is the task pool with different levels, these levels represented by L1, L2, L3 and so on. The lower part is nodes set. Each circle

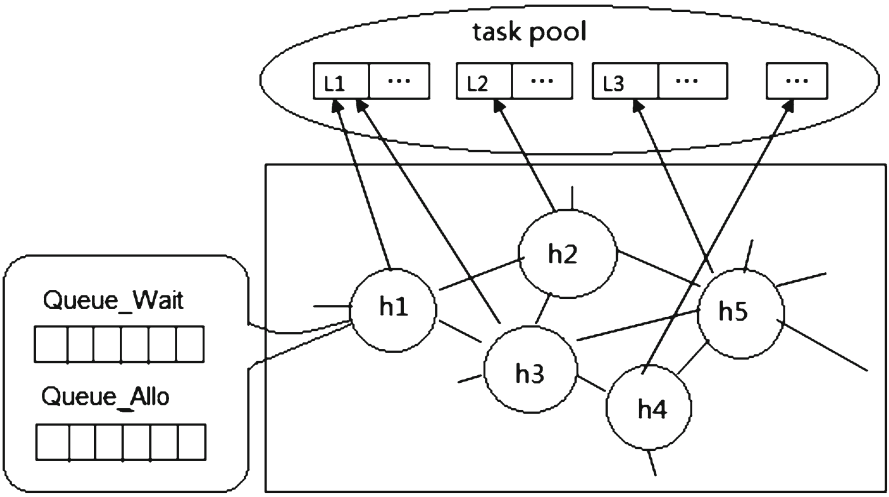

Fig. 1 The system framework

TaskID	NodeID	Computation Load	Required Storage Space	Required Disk Space	Life Cycle

Fig. 2 The format of task information

represents a node (such as h_1, h_2, h_3), which might be an independent computer or a cluster. These nodes all can meet the minimum task requirement. The left part of Fig. 1 shows that every node has two task queues. They are Queue_Wait (the queue of tasks to be scheduled) and Queue_Allo (the queue of tasks that has been allocated to the node). The arrows represent corresponding relations between nodes and levels.

The task pool with hierarchy is proposed in our paper. When a new task is generated, the task level is calculated according to the resource requirement (such as computation load, required storage space and required disk space). To search a better target node, information about every task should be recorded by the corresponding level of the task pool. Task information includes TaskID, NodeID, computation load, required storage space, required disk space and life cycle. A record in the task pool is formed by one task information. The format of task information in the task pool is shown in Fig. 2.

Similarly, the nodes adopt the idea of hierarchy. That is, the nodes are also divided into different levels. The level of nodes is decided according to their idle ability (such as idle computing power, idle storage space, idle disk space and work load). Because the idle ability of nodes is always changing, the mapping between tasks and nodes is not static but dynamical.

DHRA publicizes and manages task information of different levels in the task pool. Each node selects tasks whose resource requirement can be met from the corresponding level of the task pool, and sends the bid information to the task owner. DHRA chooses the maximum bidder as the successful node and submits the task to it. The successful node executes the task and returns the result to the task owner.

4 The dynamically hierarchical resource-allocation algorithm

In this section, DHRA algorithm is presented and discussed in detail. The proposed algorithm is based on FastBid algorithm [36] and task level. In this paper the resource-allocation problem is described as follows: n nodes and m tasks in cloud computing environment. Nodes set is represented by $H = \{h_1, h_2, \ldots, h_j, \ldots, h_n\}$. When m tasks need to be executed, the set of tasks is defined as $T = \{t_1, t_2, \ldots, t_i, \ldots, t_m\}$. Each task must be executed by one node. All nodes bid for target tasks which is in the corresponding level.

For a given task, DHRA calculates the task level. Its owner packages task information into a record and then throws it into the corresponding level of the task pool. Nodes with idle ability read tasks in the corresponding level of the task pool. If the resource requirement of the task can be met, the node will bid for it. For a specific task, only nodes which can meet the resource requirement and in the corresponding level join the bid for it. The successful bidder is selected according to the evaluation value which comprehensively considerate computing power, storage capacity and bandwidth. Then the target task was assigned to the successful node.

The algorithm contains the process of the task owner which is triggered by Queue_Wait and the process of bidders which are periodically start-up for selecting tasks from the corresponding level of task pool. The flow chart of DHRA is shown in Fig. 3.

The algorithm includes three main concerns:

1. The level of task.
2. The dynamic mapping of task level and node.
3. The evaluation value of task.

4.1 The level of task

In order to manage high efficiency of the dynamical hierarchical mapping, five levels are defined in the task pool, respectively represented by L1, L2, L3, L4 and L5. Every task belongs to one level. The process of deciding the level of a task is:

(1) The indicators of task level:

The indicators of task level are defined as a task information space, which is represented by T while $t_i (1 \leq i \leq m)$ represents different task information. In this paper, the information of a task which is considered as indicators to determine task level contains computation load (TC), required storage space (TM) and required disk space (TD). So the task information space is shown as:

$$
\begin{aligned}
&T = \{t_1, t_2, \ldots, t_i, \ldots, t_m\} \\
&\mathrm{TC} = \{\mathrm{tc}_1, \mathrm{tc}_2, \ldots, \mathrm{tc}_i, \ldots, \mathrm{tc}_m\} \\
&\mathrm{TM} = \{\mathrm{tm}_1, \mathrm{tm}_2, \ldots, \mathrm{tm}_i, \ldots, \mathrm{tm}_m\} \\
&\mathrm{TD} = \{\mathrm{td}_1, \mathrm{td}_2, \ldots, \mathrm{td}_i, \ldots, \mathrm{td}_m\} \\
&t_i = \{(\mathrm{tc}_i, \mathrm{tm}_i, \mathrm{td}_i) | \mathrm{tc}_i \in \mathrm{TC}, \mathrm{tm}_i \in \mathrm{TM}, \mathrm{td}_i \in \mathrm{TD}, 1 \leq i \leq m\}
\end{aligned}
$$

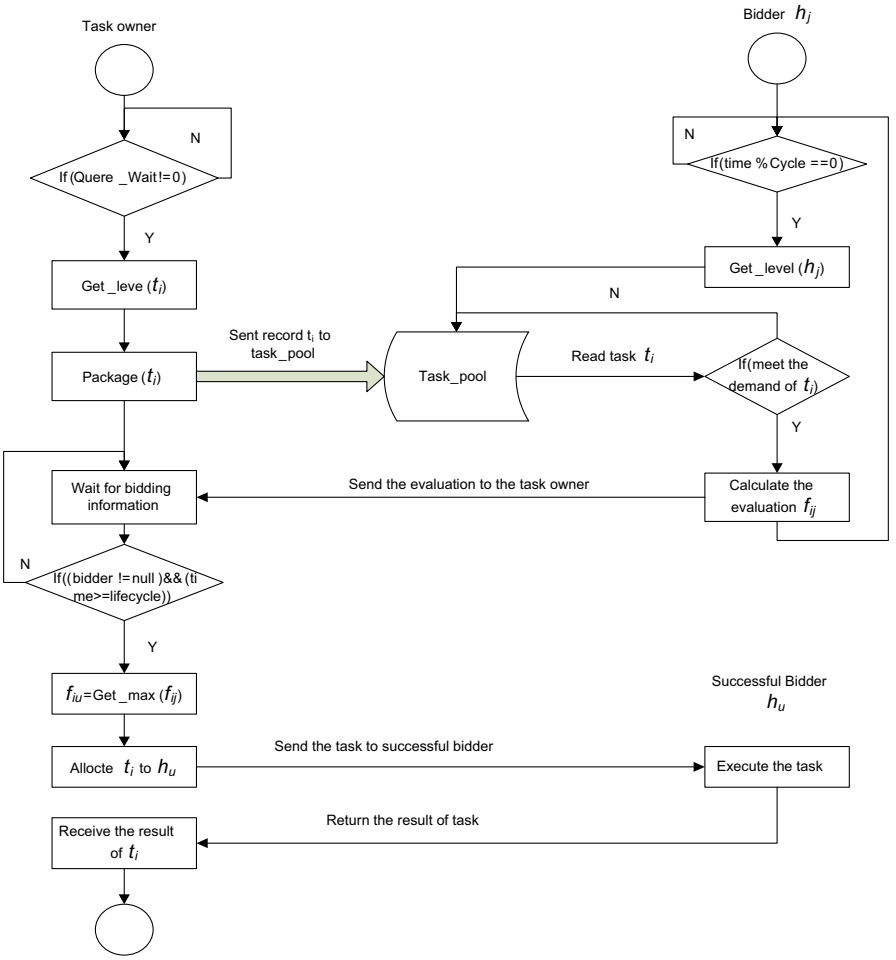

Fig. 3 The flow chart of DHRA

According to these indicators, every task belongs to the five fuzzy sets of task levels by a certain degree.

(2) Adjusting and conversion of task indicators:

For every task, in order to build the membership function of task level, the three task indicators (i.e.tc_i, tm_i, td_i) need to be normalized and standardized.

In this paper, the linear function of conversion is used to normalize the parameters so that it can obtain the value limited in (0, 1), then denote three parameters (i.e. tc_i, tm_i, td_i) after conversion them as tc_i_N, tm_i_N and td_i_N. Using the standard deviation standardization to eliminate the big gap of the indicators value, so that modify these normalized parameters (i.e. tc_i_N, tm_i_N, td_i_N) and get the standard values tc_i_S, tm_i_S and td_i_S respectively.

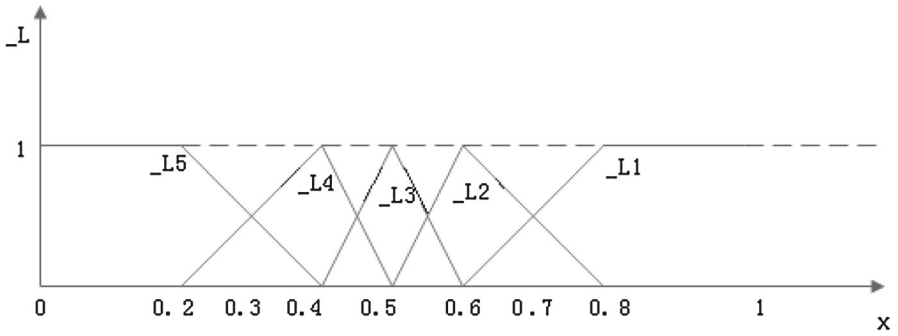

Fig. 4 The membership function of task level

(3) Membership function of task level:

The levels L1, L2, L3, L4 and L5 are a division of task information space, and constitute a group of fuzzy sets of tasks level. The corresponding membership functions are represented by _L1, _L2, _L3, _L4, _L5. x represents different value of tc_i_S, tm_i_S and td_i_S, then the membership function of fuzzy task level could be shown as:

$$
\begin{aligned}
\_\mathrm{L1} &= \begin{cases} 1; x \geq 0.8 \\ (x-0.6)^*5; 0.6 \leq x < 0.8 \\ 0; x < 0.6 \end{cases} \\
\_\mathrm{L2} &= \begin{cases} 0; x \geq 0.8 \\ (0.8-x)^*5; 0.6 \leq x < 0.8 \\ (x-0.5)^*10; 0.5 \leq x < 0.6 \\ 0; x < 0.5 \end{cases} \\
\_\mathrm{L3} &= \begin{cases} 0; x \geq 0.6 \\ (0.6-x)^*10; 0.5 \leq x < 0.6 \\ (x-0.4)^*10; 0.4 \leq x < 0.5 \\ 0; x < 0.4 \end{cases} \\
\_\mathrm{L4} &= \begin{cases} 0; x \geq 0.5 \\ (0.5-x)^*10; 0.4 \leq x < 0.5 \\ (x-0.2)^*5; 0.2 \leq x < 0.4 \\ 0; x < 0.2 \end{cases} \\
\_\mathrm{L5} &= \begin{cases} 0; x \geq 0.4 \\ (0.4-x)^*5; 0.2 \leq x < 0.4 \\ 1; 0 \leq x < 0.2 \end{cases} .
\end{aligned} \tag{1}
$$

The figure of this membership function is shown in Fig. 4.

Hence the membership degree of each task indicator to each task levels can be calculated according to the computation load of tasks, required storage space of tasks and required disk space of tasks.

(4) Calculation of task level:

tc_i_S, tm_i_S and td_i_S are introduced into the membership function Eq. (1). Then the membership degrees of computation load, required storage space and required disk space of tasks to every level are obtained, and are represented by $\_y(tc_i\_S)$, $\_y(tm_i\_S)$ and $\_y(td_i\_S)$ respectively while $y \in (L1, L2, L3, L4, L5)$. Hence the membership degree of each task to every level can be calculated according to Eq. (2).

$$\_y(t_i) = x_1{}^*\_y(tc_i\_S) + x_2{}^*\_y(tm_i\_S) + x_3{}^*\_y(td_i\_S);\ y \in (L1, L2, L3, L4, L5). \tag{2}$$

where $\_y(t_i)$ represents the membership degree of task t_i to the task level y. x_1, x_2 and x_3 are the weight of each indicator, while $x_1, x_2, x_3 \in (0, 1)$ and $x_1 + x_2 + x_3 = 1$.

Hence the task levels can be obtained on the basis of the maximum membership principle. The Equation is shown as Eq. (3)

$$\_y^*(t_i) = \max\{\_y(t_i) | y \in (L1, L2, L3, L4, L5), 1 \le i \le m\}. \tag{3}$$

Thus, we consider that level of task t_i is y, and the degree of membership is $\_y^*(t_i)$.

4.2 The dynamic mapping between task level and node

In this paper a dynamic mapping between task level and node is created. Each node corresponds to one task level at a certain point of time. The corresponding task level of nodes is calculated by using the same method of task level. The indicators of nodes are: the idle computing power ($hc_j (1 \le j \le n)$), the idle memory space ($hm_j (1 \le j \le n)$), and the idle disk space ($hd_j (1 \le j \le n)$). According to this information of node, the corresponding level of nodes can be decided according to the fuzzy pattern recognition. Since the workload of a node may change with the change of system, so the corresponding level of a node are dynamic. The dynamic mapping is constructed between tasks and nodes.

4.3 Get the evaluation value

A node can read all tasks in the corresponding level of the task pool. If it can meet the demand of a task, it needs to calculate an evaluation of itself to join the bidding. Then the task owner chooses the node which the maximum evaluation as the target node.

To achieve better comprehensive evaluation for multi-objective, we use Eq. (4) as the evaluation function to calculate the evaluation value of bidders:

$$f_{ij} = y_1 c_{ij} + y_2 / t\_end_{ij} + y_3 / w_j, (y_1 + y_2 + y_3 = 1). \tag{4}$$

f_{ij} is the evaluation value of task t_i on node h_j. c_{ij} represents the inter dependency between task t_i and node h_j which is calculated by using Eq. (5). t_end_{ij} represents the earliest completion time of t_i on h_j, which is calculated by using Eq. (6). w_j

represents the load of h_j which is calculated by using Eq. (8). y_1, y_2 and y_3 are the coefficients of c_{ij}, $1/t\_\text{end}_{ij}$ and $1/w_j$.

The definition of the inter dependency c_{ij} between t_i and h_j is:

$$c_{ij} = \left\{ \begin{array}{l} \frac{\sum_{j \in T_j} e_{ij}}{\sum_{l \in T} e_{il}}; \sum\limits_{l \in T} e_{il} \neq 0 \\ 0; \sum\limits_{l \in T} e_{il} = 0 \end{array} \right\} (i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n) \tag{5}$$

where T_j is the task set which is allocated to h_j, $\sum_{j \in T_j} e_{ij}$ represents total communication traffic between task t_i and T_j, and $\sum_{l \in T} e_{il}$ represents total communication traffic between task t_i and all tasks. Allocating tasks to the node which has maximum c_{ij} can reduce communication traffic effectively.

The earliest completion time t_end_{ij} of task t_i on node h_j is:

$$t\_\text{end}_{ij} = h\_\text{end}_j + t\_\text{Process}_{ij} + t\_\text{trans}_{ij} \tag{6}$$

where t_end_{ij} represents the calculating completion time of all tasks in node h_j which is calculated by Eq.(7). t_Process_{ij} represents the time of processing task t_i on node h_j. And t_trans_{ij} represents the time of transferring task t_i to node h_j.

The calculating completion time of all tasks in a node is represented by h_end_j. The length of ready queue is defined as Queue_Length_j. Task execution time of Queue_Allo is defined as $t\_\text{Process}_k (0 \leq k \leq \text{Queue\_Length}_j)$. Then the completion time h_end_j of node h_j is defined as:

$$h\_\text{end}_j = \sum_{k=1}^{\text{Queue\_Length}_j} t\_\text{Process}_k. \tag{7}$$

The load of the node is according to various parameters related to efficiency of the node timely, including network traffic S, CPU utilization U, memory utilization M and disk utilization D. And then calculate load w of each node by Eq. (8) according to these parameters.

$$w = x_1 S + x_2 U + x_3 M + x_4 D \left(x_i \geq 0, \sum x_i = 1 \right). \tag{8}$$

The evaluation of nodes to a task according to c_{ij}, t_end_{ij} and w_j can be calculated by using Eq. (4) and sends it to the task owner.

5 Analysis

In DHRA, the bidding process can avoid a large number of messages in the process of resources allocation. When allocating resources by using traditional algorithms, if a task is generated, the owner has to collect information of all the other nodes. All the other nodes also have to send their information to the task owner. If a system

with a large number of nodes and tasks, extensive messages will be produced in the process of resource allocation. Communication traffic will increase sharply and system performance will certainly decrease. Moreover, with the increasing of nodes, the complexity of resource allocation will also increase. DHRA uses task pool which has different levels to publish tasks. It only needs to send task information to the corresponding level in the task pool. Then any node can select reasonable task from the corresponding level instead of blindly send their resource information. Therefore the number of messages is reduced and communication traffic also decreases.

When there are n nodes and m tasks, the set of nodes is defined as $H = \{h_1, h_2, \ldots, h_n\}$ and the set of tasks is defined as $T = \{t_1, t_2, \ldots, t_m\}$.

(1) Traditional resource-allocation algorithms

In traditional resource-allocation algorithms, if there is task $t_i (1 \leq i \leq m)$ on node $h_j (1 \leq j \leq n)$, h_j has to notify all other nodes $h_k (1 \leq k \leq n, k \neq j)$ that task t_i is generated. Nodes $h_k (1 \leq k \leq n, k \neq j)$ should send their evaluation information to h_j after receiving the notification. Then the node h_j selects the target node among all bidders. Therefore, the number of messages is $2(n-1)$ for each task. For m tasks, the whole number of messages is represented by below formula:

$$a = 2m(n-1). \tag{9}$$

(2) DHRA

When allocating resources in DHRA, the tasks will be divided into different levels. The set of levels is defined as $L = \{L_1, L_2, \ldots, L_p\}$. The process of selecting the target node has three phases: recording tasks in different levels, selecting one task and bidding.

(1) Recording tasks in different levels:

In DHRA, task level is calculated by using Eq. (3) and task information will be packaged into a record and it will be sent to the corresponding level in the task pool. Only a piece of information is produced when there is a new task to be executed is generated. For m tasks, the number of messages in this phase is:

$$b_1 = m. \tag{10}$$

(2) Selecting one task:

In this phase, all nodes collect their resource information and calculate the corresponding level of the node. Then nodes select one task from the corresponding level in the task pool. If demand of the task can be met, task information will be sent to the node.

For the set of levels $L = \{L_1, L_2, \ldots, L_p\}$, assumed that the number of tasks in level L_k is $N_k (1 \leq k \leq p)$. Then the set of tasks in level L_k is defined as $T_{L_k} (1 \leq k \leq p)$ and the set of nodes in level L_k is defined as $H_{L_k} (1 \leq k \leq p)$.

The sets of tasks in different levels are:

$$\begin{aligned} T_{\mathrm{L1}} &= \{t_{\mathrm{task}i_1}, t_{\mathrm{task}i_2}, \ldots, t_{\mathrm{task}i_{N1}}\}, \\ T_{\mathrm{L2}} &= \{t_{\mathrm{task}j_1}, t_{\mathrm{task}j_2}, \ldots, t_{\mathrm{task}j_{N2}}\}, \\ &\ldots \\ T_{Lp} &= \{t_{\mathrm{task}k_1}, t_{\mathrm{task}k_2}, \ldots, t_{\mathrm{task}k_{Np}}\}, \end{aligned}$$

$t_j \in T(1 \le j \le m)$ and thus $T_{Li}(1 \le i \le p)$ is a sub-set of T.
The sets of nodes of each level are:

$$\begin{aligned} H_{\mathrm{L1}} &= \{h_{\mathrm{node}i_1}, h_{\mathrm{node}i_2}, \ldots, h_{\mathrm{node}i_{M1}}\}, \\ H_{\mathrm{L2}} &= \{h_{\mathrm{node}j_1}, h_{\mathrm{node}j_2}, \ldots, h_{\mathrm{node}j_{M2}}\}, \\ &\ldots \\ H_{Lp} &= \{h_{\mathrm{node}k_1}, h_{\mathrm{node}k_2}, \ldots, h_{\mathrm{node}k_{Mp}}\}, \end{aligned}$$

$h_j \in H(1 \le j \le n)$ and thus $H_{Li}(1 \le i \le p)$ is a sub-set of H.
Because the node needs to select one task from the corresponding level, the number of messages in this step is:

$$\begin{aligned} b_2 = &(h_{\mathrm{node}i_1} + h_{\mathrm{node}i_2} + \cdots + h_{\mathrm{node}i_{M1}}) + (h_{\mathrm{node}j_1} + h_{\mathrm{node}j_2} + \cdots + h_{\mathrm{node}j_{M2}}) \\ &+(h_{\mathrm{node}k_1} + h_{\mathrm{node}k_2} + \cdots + h_{\mathrm{node}k_{Mp}}). \end{aligned} \tag{11}$$

And because of:

$$\begin{aligned} &(h_{\mathrm{node}i_1} + h_{\mathrm{node}i_2} + \cdots + h_{\mathrm{node}i_{M1}}) + (h_{\mathrm{node}j_1} + h_{\mathrm{node}j_2} + \cdots + h_{\mathrm{node}j_{M2}}) \\ &+(h_{\mathrm{node}k_1} + h_{\mathrm{node}k_2} + \cdots + h_{\mathrm{node}k_{Mp}}) = n. \end{aligned} \tag{12}$$

So according to Eqs. (11) and (12) there is:

$$b_2 = n. \tag{13}$$

If a node met the demand of one task, task information will be sent to the node. Otherwise no more information needs to be sent. So in this step the number of messages is less than b_2, which is:

$$b_3 \le b_2. \tag{14}$$

So according to Eqs. (13) and (14), there is:

$$b_3 \le n. \tag{15}$$

(3) Bidding:

Table 1 The comparison of the number of messages

m, n	$a = 2mn$	$b_{\max} = m + 3n$	m, n	$a = 2mn$	$b_{\max} = m + 3n$
$m = 5, n = 5$	40	20	$m = 200, n = 50$	19, 600	350
$m = 10, n = 8$	140	34	$m = 500, n = 100$	99, 000	800
$m = 15, n = 10$	270	45	$m = 1000, n = 200$	398, 000	1600
$m = 50, n = 20$	1900	110	$m = 5000, n = 1000$	9, 990, 000	8000
$m = 100, n = 30$	5800	190	$m = 10{,}000, n = 2000$	39, 980, 000	16, 000

This phase is accomplished by negotiation algorithm in DHRA. After reading tasks information in the task pool, the node will send bidding information to the owner of the target task. The number of messages in this phase is:

$$b_4 \leq b_3. \tag{16}$$

So according to Eqs. (15) and (16), there is:

$$b_4 \leq n. \tag{17}$$

So the total number of messages of DHRA is:

$$b = b_1 + b_2 + b_3 + b_4 \leq m + n + n + n = m + 3n. \tag{18}$$

The maximum value of b is defined as $b_{\max}$, according to Eq. (18), there is:

$$b_{\max} = m + 3n. \tag{19}$$

With the increasing number of m and n, the numbers of messages by using traditional algorithms and DHRA are shown in Table 1.

Function $g(m, n)$ is defined as $g(m, n) = a - b_{\max} = 2m(n - 1) - (m + 3n)$, and there is:

$$\begin{aligned} g(m, n) &= 2mn - 2m - m - 3n = (mn - 3m) + (mn - 3n) \\ &= m(n - 3) + n(m - 3). \end{aligned} \tag{20}$$

When $n \geq 3$ and $m \geq 3$, there is $g(m, n) \geq 0$, which equals to $2m(n - 1) - (m + 3n) \geq 0$. So we can draw the conclusion in Eq. (21).

$$2m(n - 1) \geq (m + 3n), (n \geq 3, m \geq 3). \tag{21}$$

According to Eqs. (9) and (21) there is:

$$a \geq (m + 3n), \quad (n \geq 3, m \geq 3). \tag{22}$$

And according to Eqs. (19) and (22) we can draw the conclusion in Eq. (23).

$$b \leq a. \tag{23}$$

From Table 1, we can find that DHRA can save more number of messages compared to the traditional algorithm with the increasing of m and n. When the number of tasks is small, the number of messages of traditional algorithms and DHRA have the complexity of $O(n)$. When m is close to n, the complexity of traditional algorithms is $O(n^2)$ and the complexity of DHRA is still $O(n)$.

So with the increasing number of tasks and nodes, DHRA can reduce the number of messages effectively. Therefore, the communication traffic is also decreased.

Dividing tasks into different levels not only reduces the number of messages and communication traffic, but also reasonable the choice of the target node during allocating resources. For tasks in corresponding level, nodes join only the bidding for tasks whose resource demand can be met. Therefore the idle ability of nodes will be fully used. The problem of nodes with great idle ability being occupied by small tasks can be solved. Besides, if there are some nodes with great idle ability, the tasks with large demand can be processed as early as possible.

6 Experiments and comparisons

In this section, we simulate DHRA by using CloudSim [37], and compare the efficiency of DHRA with MinMin algorithm. In the experiments of the two algorithms, not only the environment but also the dynamical changes of nodes should be same. The results illustrate that DHRA have less makespan and communication traffic than MinMin algorithm.

6.1 Parameter settings

In the simulation experiments, the number of tasks is t and the number of nodes is n. While $n = 10$, $n = 20$ and $n = 30$, different number of random tasks are generated respectively to compare DHRA and MinMin. When $n = 10$, the number of tasks sets [30, 130]. When $n = 20$, the number of tasks sets [50, 150]. When $n = 30$, the number of tasks sets [100, 200].

In the experiments, not only the number of tasks and nodes should be same for two algorithms, but also the parameters of tasks and nodes should be same. Thus the parameters of nodes are set as follows: the range of the rating of CPU (which represents the computing power) is [1.8, 4.2 GFlops]. The range of storage space is [600, 1400 M]. The range of disk space is [60, 140 G]. The range of bandwidth is [1800, 4200 Mbps]. The parameters of tasks are set as follows: the range of computation load is [180, 420 GFlop]. The range of traffic with other tasks is [1, 3 M]. The range of required storage space is [18, 42 M]. The range of required disk space is [60, 140 M]. In each experiment, we allocate resources for the same set of tasks using the two algorithms in the same environment.

6.2 Comparison of communication traffic

The set of nodes and the set of tasks are generated randomly according to the scale of parameters above. Firstly, compare DHRA and MinMin in communication traffic through three sets of experiment with different number of nodes and tasks. The communication traffic includes not only the communication traffic between tasks, but also the communication traffic of sending tasks to the target node and the biding message. When $n = 10$, the results of communication traffic of different number of tasks are shown in Fig. 5. When $n = 20$, the results are shown in Fig. 6. When $n = 30$, the results are shown in Fig. 7.

From Figs. 5, 6 and 7 we can see that DHRA has less communication traffic than MinMin when $n = 10$, $n = 20$ and $n = 30$. The reason is that DHRA takes the communication traffic into account in the process of allocation. And in DHRA the negotiation algorithm is multi-level. So the biding relate to only the nodes in corresponding level. The nodes with lack or too more computing power does not send the bid messages, so DHRA has less communication traffic than MinMin.

As shown in Figs. 5, 6 and 7, the difference of communication traffic between DHRA and MinMin increase with the increasing of the number of nodes and tasks. In DHRA, for every task only the nodes with corresponding level join in the bid. Hence

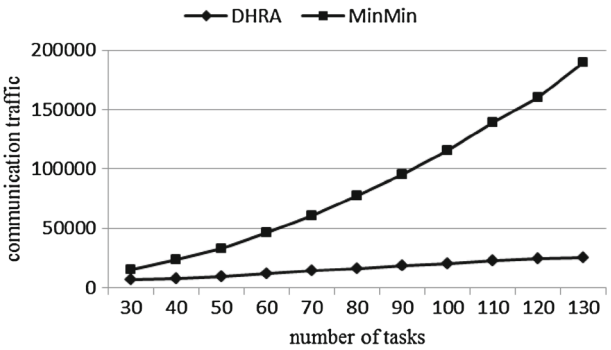

Fig. 5 The comparison of communication traffic when $n = 10$

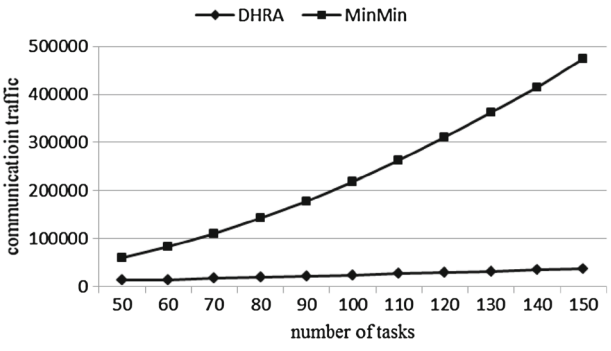

Fig. 6 The comparison of communication traffic when $n = 20$

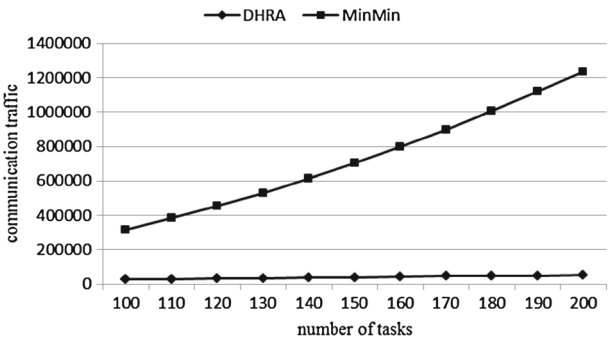

Fig. 7 The comparison of communication traffic when $n = 30$

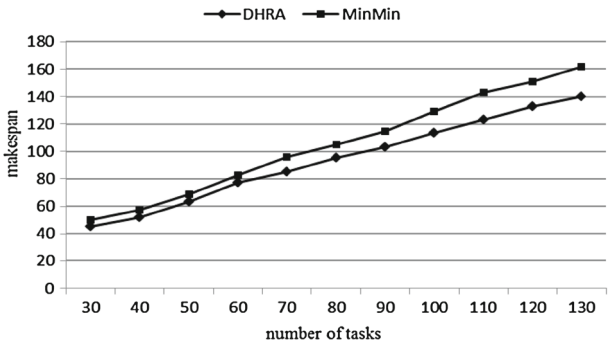

Fig. 8 The comparison of makespan when $n = 10$

with the increasing of the number of nodes and tasks, the communication traffic of DHRA grows slowly. That is to say, with the increasing of nodes and tasks number, DHRA greater advantages in terms of communication traffic of system. This result coincides with the analysis in Sect. 5.

6.3 Comparison of task completion time

After the comparison of communication traffic, we compare task completion time of DHRA with MinMin based on the same set of nodes and tasks. When $n = 10$, the results of makespan of different number of tasks are shown in Fig. 8. When $n = 20$, the results are Fig. 9. When $n = 30$, the results are shown in Fig. 10.

As Figs. 8, 9 and 10, same to the communication traffic, the makespan of DHRA also less than MinMin, and the difference between DHRA and MinMin grow with the number of nodes and tasks. Compare between the three figures, with the increasing number of tasks at the same number of nodes, the makespan of the set of dependent tasks also increase. With the increasing number of nodes, the makespan is not increasing and sometimes is decreasing. So we conclude that DHRA efficiently decreases the communication traffic of cloud computing environment. At the same time, it not only avoid prolong the makespan, but shorten it.

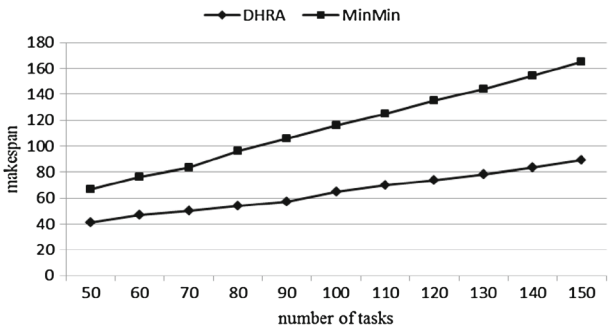

Fig. 9 The comparison of makespan when $n = 20$

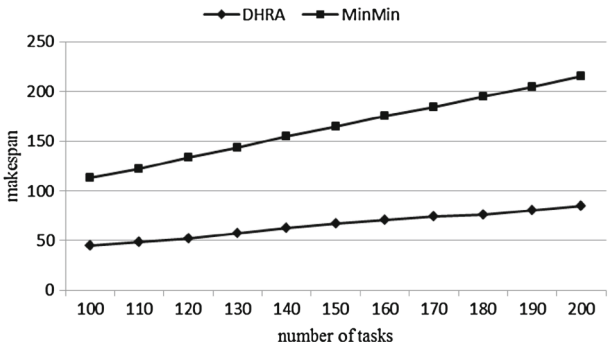

Fig. 10 The comparison of makespan when $n = 30$

7 Conclusions

In this paper, a dynamically hierarchical resource-allocation algorithm is proposed to solve the problem of extensive messages generated by the increasing of nodes in cloud computing environment. Tasks are divided into different levels based on computation load, required storage space and required disk space through fuzzy pattern recognition, and thrown into the corresponding levels of the task pool. The levels of nodes computed through the same method based on the computing power and workload are dynamically related to the corresponding task pool levels. Thus the bidding process only involves nodes and tasks of the same level and the number of messages can be reduced effectively, which means the reduction of the communication traffic, and the occupancy rates of computing, storage and communication in the system are controlled into reasonable range at the same time, making it more adaptable to the big data environment. In theoretical analysis and simulation experiments, we compared the communication traffic and the task completion time between DHRA and MinMin. The results show that DHRA can reduce message number and communication traffic significantly, with the equal or even fewer tasks completion time compared with MinMin. DHRA is proved to be efficient for resource allocation in cloud computing environment.

Acknowledgments This research was supported by the National Natural Science Foundation of China (Grant No. 61173160), Intel Industrial Liaison and Comprehensive Reform Program of Intel Semiconductor (US) LLC in 2014 of Ministry of Education of the People's Republic of China. We would like to thank the anonymous reviewers for their attentive reading and for their constructive comments that have helped to further strengthen this paper.

References

1. Michael A et al (2010) A view of cloud computing. Commun ACM 53(4):50–58
2. Seref S, Duygu S (2013) Big data: a review. In: 2013 international conference on collaboration technologies and systems, San Diego, CA, USA, pp 42–47
3. McAffe A, Brynolfsson E (2012) Strategy & competition big data: the management revolution. Harv Bus Rev 90(10):60–66, 68, 128
4. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616
5. Amazon EC2 (2015). http://aws.amazon.com/cn/ec2/. Accessed 7 Feb 2015
6. IBM SmartCloud (2015). http://www.ibm.com/cloud-computing/cn/zh/index.html. Accessed 7 Feb 2015
7. Seokho S, Gihun J, Sung CJ (2013) An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider. J Supercomput 64:606–637
8. Nathuji R, Kansal A, Ghaffarkhah A (2010) Q-clouds: managing performance interference effects for QoS-aware clouds. In: 5th ACM European conference on computer systems (EuroSys 2010), Paris, April 13–16, 2010
9. Marx V (2013) The big challenges of big data. Nature 498(7453):255–260
10. Sam M (2012) From databases to big data. IEEE Intenet Comput 12:089–7801
11. JiSu P, Hyongsoon K, Young-Sik J, Eunyoung L (2014) Two-phase grouping-based resource management for big data processing in mobile cloud computing. Int J Commun Syst 27:839–851
12. Hassan MM, Song B, Hossain MS, Alamri A (2014) QoS-aware resource provisioning for big data processing in cloud computing environment. In: 2014 international conference on computational science and computational intelligence, Las Vegas, NV, USA, March 10–13, 2014
13. Simon SW, Jelena M (2014) Optimal application allocation on multiple public clouds. Comput Netw 68:138–148
14. Liang Q, Zhang J, Zhang YH, Liang JM (2014) The placement method of resources and applications based on request prediction in cloud data center. Inf Sci 279:735–745
15. Yin C, Huang BQ, Liu F et al (2011) Common key technology system of cloud manufacturing service platform for small and medium enterprises. Comput Integr Manuf Syst 17:495–503
16. Amit N, Sanjay C, Gaurav S (2012) Policy based resource allocation in IaaS cloud. Future Gener Comput Syst 28:94–103
17. Christian V, Rodrigo NC, Dileban K, Rajkumar B (2012) Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka. Future Gener Comput Syst 28:58–65
18. Guiyi W, Athanasios V, Vasilakos YZ, Naixue X (2010) A game-theoretic method of fair resource allocation for cloud computing services. J Supercomput 54:252–269
19. Anton B, Jemal A, Rajkumar B (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener Comput Syst 28:755–768
20. Young CL, Albert YZ (2012) Energy efficient utilization of resources in cloud computing systems. J Supercomput 60:268–280
21. Wu CM, Chang RS, Chan HY (2014) A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. Future Gener Comput Syst 31:141–147
22. Xiao Z, Song WJ, Chen Q (2013) Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans Parallel Distrib Syst 24(6):1107–1117
23. Daniel W, Odej K (2011) Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. IEEE Trans Parallel Distrib Syst 22(6):985–997
24. Ahmed S, Kenli L, Aijia O, Zhiyong L (2014) Proactive workload management in dynamic virtualized environments. J Comput Syst Sci 80:1504–1517

25. Haluk T, Salim H, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274
26. Wang ZJ, Liu JJ (2011) Research on multi-agent-based distributed task scheduling mechanism with multi-objective. J Dalian Univ Technol 51(5):755–760
27. Wang ZJ, Fang T (2014) Task scheduling model based on multi-agent and multi-objective dynamical scheduling algorithm. J Netw 9(6):1588–1595
28. Grekovs R (2002) Methods of fuzzy pattern recognition. In: Scientific proceedings of RIGA Technical University
29. Pedrycz W (1990) Fuzzy sets in pattern recognition: methodology and methods. Pattern Recognit 23(1–2):121–146
30. Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York
31. Zhao RH, Lu XL, Wang M (2011) Safety evaluation of urban flood control system based on variable fuzzy pattern recognition. Adv Mater Res 159:264–269
32. Cui XJ, Cao BY (2010) Using two-level fuzzy pattern recognition in the classification of convex quadrilateral. In: 2nd international conference on quantitative logic and soft computing, vol 82, pp 527–534
33. Fatma AO, Rasha MZ (2010) Dynamic task scheduling using fuzzy logic in distributed memory systems. In: 2010 7th international conference on informatics and systems, Cairo, Egypt, vol 5
34. Sepideh A, Ali M, Amir MR, Hamid B, Hengameh DT (2013) A new fuzzy negotiation protocol for grid resource allocation. J Newt Comput Appl. doi:10.1016/j.jnca.2012.12.030
35. Guo FY, Yu L, Tian S, Yu J (2014) A workflow task scheduling algorithm based on the resources' fuzzy clustering in cloud computing environment. Int J Commun Syst. doi:10.1002/dac.2743
36. Akihiro K, Nathan S (2008) Optimized algorithms for multi-agent routing. In: AAMAS '08 proceedings of the 7th international joint conference on autonomous agents and multiagent systems, New York, pp 1585–1588
37. Rodrigo NC, Rajiv R, Anton B, César AFDR, Rajkumar B (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithm. Softw Pract Exp 41:23–50