SCIENCE CHINA
Information Sciences

• **RESEARCH PAPER** •

May 2017, Vol. 60 052103:1–052103:17
doi: 10.1007/s11432-015-5492-6

Credit-based scheme for security-aware and fairness-aware resource allocation in cloud computing

Di LU*, Jianfeng MA, Cong SUN, Xindi MA & Ning XI

School of Computer Science, Xidian University, Xi'an 710071*, China*

Received August 28, 2015; accepted October 9, 2015; published online September 12, 2016

Abstract Cloud computing systems include different types of participants with varied requirements for resources and multiple tasks; these varying requirements must be considered in the design of fairness-aware resource allocation schemes for better resources sharing. However, some participants may be malicious with a goal to damage the resource allocation fairness and increase their own utility. Hence, the resource scheduling policy must guarantee allocation fairness among the participants; further, it must ensure that fairness is not affected by the malicious usage of resources, that could cause resource exhaustion and lead to denial of service. In order to address this challenge, we propose a credit-based mechanism for resource allocation that will avoid the malicious usage of resources and, simultaneously, guarantee allocation fairness. In our scheme, a credit factor is introduced for each participant in order to evaluate the history of resource utilization and determine future resource allocation. Our model encourages a participant to release the occupied resources in timely manner after the completion of a task and imposes a punishment for malicious occupation of resources. We prove the fairness of our model and provide linear and variable gradient approaches to determine the credit factor for different scenarios. We simulate our model and perform experiments on a real cloud computing platform. The results prove the rationality, effectiveness and correctness of our approaches.

Keywords credit, resource allocation, security, fairness, cloud computing

Citation Lu D, Ma J F, Sun C, et al. Credit-based scheme for security-aware and fairness-aware resource allocation in cloud computing. Sci China Inf Sci, 2017, 60(5): 052103, doi: 10.1007/s11432-015-5492-6

1 Introduction

In cloud computing, virtual machines (VMs), known as the computing participants, are used to provide services or execute user tasks. When VMs execute in a cloud and are connected via a software-defined network (SDN), each VM can be logically considered as a "computing node (CN)". In order to ensure that each CN can execute a task by using an appropriate amount of resources, several challenges must be addressed during the design of the resource allocation scheme. First, cloud systems are open and dynamic, and hence, they consist of different types of CNs having varied requirements for public resources and multiple tasks; these requirements should be considered while designing fair resource allocation schemes to ensure better sharing of resources. For instance, a CN may have several tasks with different priorities. For high-priority tasks (e.g., some time-critical tasks), the CN must obtain some resources promptly to complete the tasks immediately. For low-priority tasks, the CN may tolerate a certain amount of

* Corresponding author (email: dlu@xidian.edu.cn)

delay in the completion of these tasks, especially when resources are in limited supply or many nodes compete for the same resources. Hence, fairness should be guaranteed during resource allocation among multiple participants. Second, a CN could be selfish in order to maximize its own utility (in this context, we refer to the user of the CN). Further, some CNs may even be malicious and intend to exhaust the system resources. Therefore, some special mechanisms are needed to detect these malicious participants and resist their harmful behavior while allocating resources. Hence, a security-aware resource allocation scheme with fairness is crucial in the resource management of cloud computing system.

Fairness guarantees that each VM can be allocated a suitable amount of resources to execute a task. Thus, the resource demands of a VM can be adequately satisfied without causing resource starvation in other VMs. Some typical algorithms for allocation fairness have been studied for single and multiple types of resources. In max-min fairness [1, 2], the share of user i cannot be increased without decreasing the share of another node j whose share is smaller than or equal to that of i. The algorithm, proportional fairness [3] attempts to determine a balance point for resource allocation among competing interests. In α-fairness [4], fairness and efficiency are considered. The authors propose an approach that determines a trade-off between fairness and efficiency to improve resource utilization. For the allocation of resources of multiple types, Refs. [5–8] focus on multiple instances of the same resource. The authors of [9] proposes dominant resource fairness (DRF), an algorithm that is designed to ensure fairness in the allocation of multiple types of resources, such as CPU, memory and bandwidth. Although these algorithms provide effective and fair allocation strategies for different scenarios, the malicious behaviors in resource utilization are not considered in their design.

As mentioned earlier in this section, a CN is "selfish", i.e., each CN attempts to maximize its share of the system resources at the expense of others CNs. This behavior may result in resource exhaustion, that leading to unavailability of the cloud system. Hence, malicious resource occupation can lead to resource starvation or denial-of-service attacks [10–13].

In our research, we observed that a malicious CN can possess a fraction of the resources after each task, which cannot be detected immediately and then it will possess a large portion of the resources after completing several tasks. Thus, the system resource can be exhausted gradually. We term this behavior as "slow resource depletion (SRD)". Through SRD, a malicious node can also accomplish a denial-of-service attack, which cannot be detected in a short period of time. In [14, 15], the authors propose a reputation based approach to prevent the exhaustion of system resource by the selfish behavior of users. In [16, 17], the authors propose approaches to restrain the resource allocation via a credit mechanism in a cloud computing and a P2P network, respectively. The authors of [18–22] provide a credit-based mechanism for resource management in a grid environment. The study in [10] emphasizes that a compromised VM can obtain a large portion of the resources and damage the resource sharing mechanism of the cloud; the authors propose a bidding-based method to solve this issue. The studies in [11, 23] state that threats in resource-sharing are typically caused by an intruder scheduler. Hence, the abuse of resource can be handled by improving the allocation algorithm in the resource scheduler. However, in these studies, fairness is not taken into account.

The study in [9] proposes dominant resource fairness (DRF) with four basic principles to guarantee fairness in resource allocation: share incentive, strategy-proofness, envy-freeness and Pareto efficiency; these principles can also be considered as fairness strengthening criteria. DRF is derived from the max-min algorithm; therefore, it has a time complexity of $O(n \cdot k)$, where n denotes the number of users and k indicates the average number of task per VM. Hence, DRF demonstrates better performance than other non-linear algorithms, such as genetic algorithm [24, 25] and the game theory-based approach [26]. Further, DRF focuses on the primary resource demands of users instead of considering all types of resources equally. However, DRF cannot prevent the exhaustion of system resources resulting from the malicious use of resources by various methods, such as SRD.

Hence, we propose a credit-based dominant resource fairness (cbDRF) algorithm based on DRF that considers allocation fairness and prevents the resource abuse caused by a malicious computing node. In cbDRF, a credit is assigned to each node to measure its reputation; a credit factor is introduced to evaluate the credit value according to the usage of resources by the node. If the node occupies a resource for a long period of time without releasing a suitable amount, its credit will be reduced, thus decreasing its future resource share. We prove that cbDRF satisfies the four important fairness properties — sharing incentive, strategy-proofness, envy-freeness and Pareto efficiency — proposed in [9]. Further, cbDRF satisfies the release incentive and punitive allocation properties that are significant in protecting the fairness mechanism and system resources.

The rest of the paper is organized as follows: In Section 2, the proposed cbDRF algorithm is described, and the linear and variable gradient approaches to determine the credit factor are presented. In Section 3, we prove that cbDRF satisfies the four basic fairness principles; futher, we propose and analyze the simulations for the two models. Next, a comparison with representative related work is presented and the experiments on a real cloud computing platform (OpenStack1)) are described. Finally, the concluding remarks are stated in Section 4.

2 System model

2.1 Dominant resource fairness

In [9], the DRF algorithm for allocation of multiple types of resources is presented. This algorithm is based on the principle of dominant share of the users. However, the main concept of DRF is derived from max-min fairness; DRF attempts to apply max-min fairness to the dominant resource shares [27]. Specifically, four important properties of resource allocation are proposed in [9]; these properties are also fairness strengthening criteria.

The DRF algorithm can be briefly described as follows: Let us consider a system consisting of two types of resources r_1 and r_2 having capacities of R_1 and R_2 respectively, and two users, i and j, whose resource demand vectors are $\boldsymbol{D}_i$ and $\boldsymbol{D}_j$, respectively. Then $\boldsymbol{D}_i = \{d_{i,r_1}, d_{i,r_2}\}$, $\boldsymbol{D}_j = \{d_{j,r_1}, d_{j,r_2}\}$. If $\frac{d_{i,r_1}}{R_1} > \frac{d_{i,r_2}}{R_2}$ and $\frac{d_{j,r_1}}{R_1} < \frac{d_{j,r_2}}{R_2}$, the dominant resource (DRes) of user i is r_1, and the DRes of user j is r_2. Assuming that x_i and x_j are the number of task users i and j respectively, x_i and x_j can be determined by (1):

$$\begin{cases} d_{i,r_1}x_i + d_{j,r_1}x_j \leqslant R_1, \\ d_{i,r_2}x_i + d_{j,r_2}x_j \leqslant R_2, \\ \dfrac{d_{i,r_1}}{R_1}x_i = \dfrac{d_{j,r_2}}{R_2}x_j. \end{cases} \tag{1}$$

Then, the dominant share (DShr) of i is $d_{i,r_1}x_i$, and that of j is $d_{j,r_2}x_j$. Thus, the number of tasks of the user x_k is determined by its DRes. Hence, the user share of a resource is indirectly determined by DRes. For example, the share of user i corresponding to r_2 is $d_{i,r_2}x_i$, where x_i is determined by r_1.

2.2 Credit-based DRF model

2.2.1 *Threats to resource scheduling*

As mentioned earlier, resource scheduling is threatened by the malicious use of resources, caused by factors such as long time resource occupation and slow resource depletion, which can lead to a DoS attack on the system resources. In order to address this issue, we use a credit factor to measure the reputation of the CN according to its resource utilization. The usage of the credit factor will ensure the active release of occupied resources by the nodes (release incentive) and the imposition of punitive allocation for malicious behavior. In this section, we first present the linear cbDRF model, and then, propose the variable gradient model.

2.2.2 *Task-dependent DRF model*

Let us consider a system with n types of resources. The demand vector of the node i is $\boldsymbol{D}_i$. The tasks of the node are periodic; hence, we assume that user i executes task $T_{\langle t,m\rangle}$ in the period $\langle t,m\rangle$, and

1) OpenStack. http://www.openstack.org/.

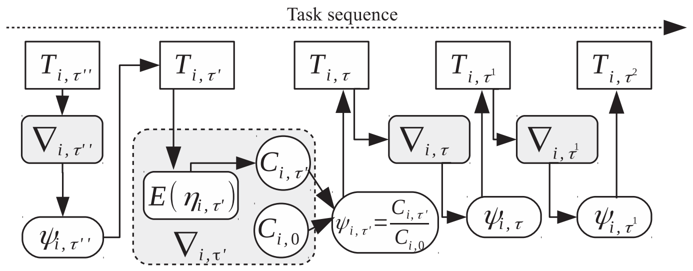

Figure 1 Credit-based DRF model in the task sequence.

the demand vector of i can be denoted as $\boldsymbol{D}_{i,\langle t,m\rangle}$. In this definition, t denotes the start time of the computational job, and m indicates its duration. If i has x_i sub-tasks, the resource share vector of i can be denoted by (2):

$$\boldsymbol{S}_{i,\langle t,m\rangle} = \boldsymbol{D}_{i,\langle t,m\rangle} \cdot x_{i,\langle t,m\rangle}, \tag{2}$$

where we have $\boldsymbol{S}_{i,\langle t,m\rangle} = \{s_{i,1}^{\langle t,m\rangle}, s_{i,2}^{\langle t,m\rangle}, \ldots, s_{i,n}^{\langle t,m\rangle}\}$, $\boldsymbol{D}_{i,\langle t,m\rangle} = \{d_{i,1}^{\langle t,m\rangle}, d_{i,2}^{\langle t,m\rangle}, \ldots, d_{i,n}^{\langle t,m\rangle}\}$.

The demand vector of the node varies for a different computational period, and hence, the DShr of the computing node must be re-computed. According to the definition of dominant resource, we can define the DRes of node i in $\langle t,m\rangle$ as $\mu_{i,\langle t,m\rangle} = \max_k\{d_{i,k}^{\langle t,m\rangle}/R_k\}$, $k = 1,2,3,\ldots,n$. Then, the DShr ($s_i^{\langle t,m\rangle}$) of i can be determined as (3), in which the DRes is r_k:

$$\begin{cases} d_{i,k}^{\langle t,m\rangle} x_i^{\langle t,m\rangle} + \sum_{j\neq i} d_{j,k}^{\langle t,m\rangle} x_j^{\langle t,m\rangle} \leqslant R_k, \\ \mu_{i,\langle t,m\rangle} x_i^{\langle t,m\rangle} = \mu_{j,\langle t,m\rangle} x_j^{\langle t,m\rangle}, j = 1,2,\ldots, j \neq i. \end{cases} \tag{3}$$

Thus, the DShr of node i in $\langle t,m\rangle$ is $s_i^{\langle t,m\rangle} = d_{i,k}^{\langle t,m\rangle} x_i^{\langle t,m\rangle}$. Thus, we obtain the task-dependent DRF model.

2.2.3 *cbDRF model*

In order to measure the reputation of a node, we assign a credit $\boldsymbol{C}$ to each node. $\boldsymbol{C}$ has an initial value that increases or decreases with resource utilization. An evaluation parameter η is introduced to evaluate the credit variation. We define a function $E(\eta)$ such that $\boldsymbol{C} = E(\eta)$.

Let us assume that the initial credit is $\boldsymbol{C}_0$; we define credit factor as $\psi = \boldsymbol{C}/\boldsymbol{C}_0$. Then, $\psi \to 0$ if $\boldsymbol{C}$ decreases, and $\psi \to 1$ if $\boldsymbol{C}$ increases. Let us consider node i and two computational periods, $\tau' = \langle t', m'\rangle$ and $\tau = \langle t,m\rangle$, where $t > t', t = t' + m'$. The credit in the period τ' is $\boldsymbol{C}_{i,\tau'} = E(\eta_{i,\tau'})$. Thus, the credit factor of i in τ' can be denoted as $\psi_{i,\tau'} = \boldsymbol{C}_{i,\tau'}/\boldsymbol{C}_0, (\boldsymbol{C}_{i,\tau'} < \boldsymbol{C}_0)$. Then, cbDRF can be defined as follows:

$$\boldsymbol{S}_{i,\tau} = \boldsymbol{D}_{i,\tau} \cdot \lceil x_{i,\tau} \cdot \psi_{i,\tau'} \rceil. \tag{4}$$

According to (4), credit factor $\psi_{i,\tau'}$ evaluates the resource utilization in the previous task of node i, and $x_{i,\tau}$ denotes the number of subtasks in τ. Thus, the resource share of node i is closely related to the resource utilization in the previous task.

Figure 1 shows the cbDRF model in the task sequence of the node. $T_{\langle\cdot,\cdot\rangle}$ denotes the task period. The credit $\boldsymbol{C}$ is determined by η, which is derived from the utilization of resources, and the credit factor ψ is determined by $\boldsymbol{C}/\boldsymbol{C}_0$. Then we define the process of evaluating ψ as ∇. Thus, as shown in Figure 1, the task number of the τ phase is determined by the credit factor, $\psi_{\tau'}$, obtained from the previous period τ'. Hence, cbDRF is also a feedback-based model.

In cbDRF, if a node does not release an appropriate amount of its allocated resources in a timely manner, its credit factor decreases, thus decreasing its resource share. Thus, punitive allocation is imposed on the node. This mechanism guarantees that the nodes do not possess excessive resource after a task completes.

2.3 Linear credit evaluation

Let us consider a simple scenario — the node tasks are light in weight and have similar complexity. Thus, the amount of released resources after the completion of each task is similar. Hence, we can set a threshold to measure the released resources, and thus, determine the value of credit.

We define the theoretical quantity of released resource of i after the period τ' as $\Re_{i,\tau}$, which can be simply denoted as $\Re_{i,\tau} = x_{i,\tau'} \cdot \sum_k d_{i,k}^{\tau'}$. The total amount of allocated resources of i in period τ', denoted as $A_{i,\tau'}$, is equal to $\Re_{i,\tau'}$. However, in a real environment, the computing node would not completely release all the allocated resources after finishing a task, because the node must perform some cleanup (e.g., writing data to disk, cleaning temporary storage or memory space) and prepare for the next task. Hence, $A_{i,\tau'} \geqslant \Re^{\text{act}}_{i,\tau'}$, where $\Re^{\text{act}}_{i,\tau'}$ denotes the actual amount of released resources at the end of the period τ'.

In order to assess the release of resources by node i, we define the evaluation parameter η as follows:

$$\eta_{i,\tau'} = \frac{\Re^{\text{act}}_{i,\tau'}}{A_{i,\tau'}}, \quad 0 \leqslant \eta_{i,\tau'} \leqslant 1. \tag{5}$$

From (5), if $\eta \to 0$, it implies that the node continues to possess a large amount of the resource after the completion of its task in period τ'. However, if $\eta \to 1$, it indicates that the node releases the allocated resource in a timely manner after its task ends.

In order to evaluate η, a threshold ρ is introduced, and we define the following equation to compute $\boldsymbol{C}$:

$$\begin{cases} \boldsymbol{C}_{i,\tau} \leftarrow \boldsymbol{C}_{i,\tau'} + \varepsilon, & \text{if } \rho \leqslant \eta_{i,\tau} \leqslant 1, \\ \boldsymbol{C}_{i,\tau} \leftarrow \boldsymbol{C}_{i,\tau'} - \varepsilon', & \text{if } 0 < \eta_{i,\tau} < \rho. \end{cases} \tag{6}$$

Eq. (6) constructs function $E(\eta)$ that depicts the linear relation between $\boldsymbol{C}_i$ and η based on the threshold ρ. The credit $\boldsymbol{C}_i$ increases (by ε units) if the computing node releases appropriate amount of allocated resources when a task ends. Conversely, $\boldsymbol{C}_i$ decreases (by ε' units) if the node continues to possess an excessive amount of resources after completing a task.

Figure 2 shows a linear model for evaluating the credit of a node. The gray block shows the details of the implementation of process $\nabla_{i,\tau'}$ in the period τ'. The evaluation parameter for node i in the period τ, $\eta_{i,\tau}$, is determined as $\Re^{\text{act}}_{i,\tau}/A_{i,\tau}$. In comparison with Figure 1, in a linear model, E is constructed using $\Re^{\text{act}}_{i,\tau}$, $A_{i,\tau}$ and threshold ρ.

2.4 Variable gradient credit evaluation

In the linear model, a constant ρ is feasible, and hence, the variation of $\boldsymbol{C}$ is linear. Let us consider an example of a more complex case, in which a different amount of memory is utilized in each task, and hence, the time for resource release can vary according to the memory consumption. The node may not release an appropriate amount of a possessed resource when the task ends. Hence, its credit $\boldsymbol{C}$ will decrease linearly, thus leading to a quick reduction in the share. In this case, the node can be mistakenly punished. Hence, the evaluation of $\boldsymbol{C}$ should include resource-release status in multiple tasks instead of a single task. Thus, we intend to propose a variable gradient mechanism to assess $\boldsymbol{C}$ instead of using a constant ρ.

2.4.1 *Resource occupation factor*

Let us assume that node i releases $\Re^{\text{act}}_{i,r,\tau'}$ for resource r after the phase τ'; the possessed resource of i can be denoted by $\Delta_{i,r,\tau'} = A_{i,r,\tau'} - \Re^{\text{act}}_{i,r,\tau'}$. Then, the evaluation parameter η can be redefined as $\eta^D_{i,r,\tau'} = \Delta_{i,r,\tau'}/\Re^{\text{act}}_{i,r,\tau'}$. If we consider n types of resources, the resource occupation factor (ROF) of the node can be defined by (7) (assuming that the resources have the same weight):

$$\eta^D_{i,\tau'} = \frac{\sum_{k=1}^{n} \frac{\Delta_{i,r_k,\tau'}}{\Re^{\text{act}}_{i,r_k,\tau'}}}{n}. \tag{7}$$

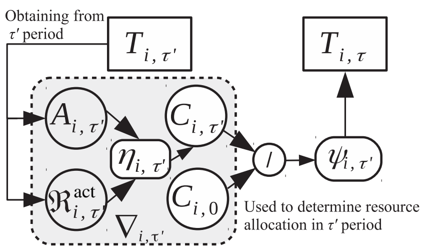
Obtaining from τ' period
$T_{i,\tau'}$
$T_{i,\tau}$
$A_{i,\tau'}$
$C_{i,\tau'}$
$\eta_{i,\tau'}$
$\mathfrak{R}^{\text{act}}_{i,\tau'}$
$C_{i,0}$
$/$
$\psi_{i,\tau'}$
$\nabla_{i,\tau'}$
Used to determine resource allocation in τ' period

Figure 2 Process for static credit evaluation.

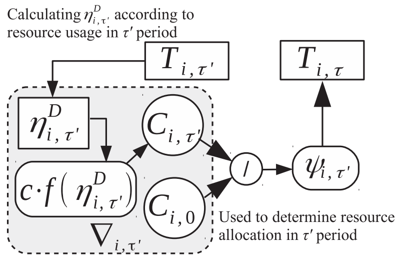
Calculating $\eta^D_{i,\tau'}$ according to resource usage in τ' period
$T_{i,\tau'}$
$T_{i,\tau}$
$\eta^D_{i,\tau'}$
$C_{i,\tau'}$
$c\cdot f(\eta^D_{i,\tau'})$
$C_{i,0}$
$/$
$\psi_{i,\tau'}$
$\nabla_{i,\tau'}$
Used to determine resource allocation in τ' period

Figure 3 Process for variable gradient credit evaluation.

From (7), if $\eta^D_{i,\tau} \geqslant 1$, node i continues to occupy more resources than the released amount; however, if $0 < \eta^D_{i,\tau} < 1$, it indicates that node i has released most of its allocated resources. In particular, when $\eta^D_{i,\tau} = 1$, node i occupies half of the allocated amount of resources. We define the following principle:

$$\begin{cases} \boldsymbol{C}_{i,\tau} \to \boldsymbol{C}_0, & \left(\eta^D_{i,\tau'} \to 0\right); \\ \boldsymbol{C}_{i,\tau} \to 0, & \left(\eta^D_{i,\tau'} \to \infty\right). \end{cases} \tag{8}$$

Let us assume that credit $\boldsymbol{C}_{i,\tau}$ and $\eta^D_{i,\tau'}$ satisfy $\boldsymbol{C}_{i,\tau} = c \cdot f(\eta^D_{i,\tau'})$, where c is the upper bound of $\boldsymbol{C}_{i,\tau}$, and $f \in (0,1]$. As mentioned earlier, the pivot of $\eta^D_{i,\tau'}$ is 1, and hence, the inflection point of f is 1. We must further define function f to satisfy the principle (8) (Eq. (8)).

2.4.2 *Variable gradient $\boldsymbol{C} \sim \eta$ model*

Based on the discussion in Subsection 2.4.1, f should have the following properties:

Property 1. $f(a) > f(b)$, if $0 < a < b$.

Property 2. $f(0) = 1$ and $\lim_{x\to\infty} f(x) = 0$.

Hence, f is defined by using a normalized arctangent function; it can be defined as follows:

$$f(\eta) = -\frac{\arctan(k\eta - \xi) + \arctan(\xi)}{\frac{\pi}{2} + \arctan(\xi)} + 1. \tag{9}$$

In (9), ξ is a control parameter that determines the variation rate of f. As mentioned earlier, the inflection point of f is 1; hence, k is introduced as the coefficient to adjust the inflection point of f, and thus, $k\eta = \xi$ at that point. The detailed analysis of f will be presented in the next section.

Thus, the credit of node i in the period τ can be denoted as

$$\boldsymbol{C}_{i,\tau} = c \cdot f(\eta^D_{i,\tau'}) = -c \cdot \left[\frac{\arctan(k \cdot \eta^D_{i,\tau'} - \xi) + \arctan(\xi)}{\frac{\pi}{2} + \arctan(\xi)} - 1\right]. \tag{10}$$

As shown in (10), $\eta^D_{i,\tau'}$ is determined by the ratio of reserved resources to released resources in the period τ'. We know that $f \in (0,1]$; hence, $\boldsymbol{C}_{i,\tau} \in (0,c]$ and f control the variation rate of the credit $\boldsymbol{C}_{i,y}$ of node i (y denotes the arbitrary task period).

In Figure 3, the process $\nabla_{i,\tau'}$ to determine $\psi_{i,\tau'}$ by using a variable gradient model is detailed. In comparison with the linear model, $\eta^D_{i,\tau'}$ is constructed according to the proportion of the reserved resources of the node ($\Delta_{i,r_k,\tau'}/\mathfrak{R}^{\text{act}}_{i,r_k,\tau'}$), and the normalized function f implements $E(\eta)$.

3 Model analysis

This section focuses on two aspects: (1) analysis of the cbDRF scheme, and (2) discussion of the allocation to nodes in the linear and variable gradient credit models.

3.1 Analysis of cbDRF

In this subsection, we present and prove the key features of cbDRF.

Theorem 1. In cbDRF, let us assume that n types of resources exist, where r is the DRes of node i with the demand $d_{i,r}$ for r. Let $s_{i,k},(k \neq r)$ denote the share of the non-DRes resource; then $s_{i,k}$ is determined by $d_{i,r}$ (denoted as $s_{i,k} \sim d_{i,r}$).

Proof. According to the definition of DRF, the share vector of i can be denoted as

$$\boldsymbol{S}_i = \boldsymbol{D}_i \cdot x_i \quad \text{and} \quad \frac{d_{i,r}}{R_r} x_i = \frac{d_{j,p}}{R_p} x_j, \quad i \neq j,$$

where r and p denote the DRes of i and j respectively, and R_y is the capacity of resource y; x_i and x_j are the number of sub-tasks of nodes i and j respectively. Hence, x_i is determined by $d_{i,r}$, and therefore, $\boldsymbol{S}_i$ is indirectly determined by $d_{i,r}$. Thus, $s_{i,k} \sim d_{i,k}$.

Theorem 2. In cbDRF, the credit factor ψ has a cumulative effect in the task sequence.

Proof. Let us consider that node i executes the task sequence shown in Figure 1. Let us assume that node i occupies excessive resources after each task phase between T_{i,τ_k} and $T_{i,\tau_{k+\lambda}}$; then, $\boldsymbol{C}_i$ would continue to decrease. ψ depends on $\boldsymbol{C}_i$, and hence, with the decrease in $\boldsymbol{C}_i$, we obtain the relation: $\psi_{i,\tau_k} > \psi_{i,\tau_{k+1}} > \psi_{i,\tau_{k+2}} > \cdots > \psi_{i,\tau_{k+\lambda-1}} > 0$. This relation indicates that $\boldsymbol{C}_i$ would gradually decrease if i does not release sufficient resources after each task. Similarly, if i releases its occupied resources in a timely manner after the task ends, the credit factor satisfies the relation: $\psi_{i,\tau_k} < \psi_{i,\tau_{k+1}} < \psi_{i,\tau_{k+2}} < \cdots < \psi_{i,\tau_{k+\lambda-1}} \leqslant 1$.

Corollary 1. cbDRF is able to encourage the node to release its occupied resources after completing the task.

Proof. Let us assume that node i releases the occupied resources in a timely manner when each task phase ends. According to Theorem 2, $\boldsymbol{C}_i$ will increase, and thus ensure that the number of sub-task will not decrease in the future tasks. Hence, to ensure that the sub-task and resource share are not influenced by the reduction of the node credit, the node must release the appropriate amount of occupied resources in a timely manner.

Corollary 2. cbDRF satisfies the share incentive property.

Proof. According to Corollary 1, in order to ensure that its own share is not reduced in the future task, the node will actively release an appropriate amount of resources when the current task ends, thus also guaranteeing the share of other nodes.

Corollary 1 provides a crucial security property, release incentive, for cbDRF. Corollary 2 implies that cbDRF also satisfies an important property, share incentive, of DRF. Below, we analyze the security properties of cbDRF, including punitive allocation introduced in cbDRF, and strategy-proofness, envy-freeness, and Pareto-efficiency proposed in DRF.

Theorem 3. Punitive allocation will be imposed on a node if it occupies an excessive amount of resources after each task ends.

Proof. Let us consider that node i executes task $T_{i,\tau_k} \sim T_{i,\tau_{k+\lambda}}$; let A_{i,τ_p} and $\Re^{\text{act}}_{i,\tau_p}$ denote the allocated resources and actually released resources, respectively, in T_{i,τ_p}. If $\Re^{\text{act}}_{i,\tau_p} \ll A_{i,\tau_p}$, according to Theorem 2, ψ continues to decrease, thus leading to $x_{i,\tau_p} \ll x^{\exp}_{i,\tau_p}$ (x_{i,τ_p} denotes the expected number of sub-tasks); hence, according to (4), $\boldsymbol{S}_i$ decreases.

Theorem 4. cbDRF satisfies the Envy-free property.

Proof. Let us assume that node i "envies" the share of j; thus, j has a greater share than i ($s_j > s_i$), and these resources are also important to i. Let us define these resources as $r_k,(k = 1, 2, \ldots, \lambda)$ and consider two cases:

(1) r_k is the DRes of i and j. In this case, r_k is unique, and $d_{i,r_k} < d_{j,r_k}$ according to assumption. From Eq. (1), $\frac{d_{i,r_k}}{R_k} x_i = \frac{d_{j,r_k}}{R_k} x_j$, and therefore, $x_i > x_j$. Thus, the DShr of i and j can be balanced by increasing the number of sub-tasks of i, and the share of i will not be affected.

Table 1 Comparison between DRF and cbDRF

Property	DRF	cbDRF	Basis
Release incentive	×	√	Corollary 1
Sharing incentive	√	√	Corollary 2
Strategy-proofness	√	√	Theorem 5
Envy-freeness	√	√	Theorem 4
Pareto efficiency	√	√	Theorem 6
Punitive allocation	×	√	Theorem 3

(2) r_k is not the DRes of i, but important to i, and j possesses a greater share. Let us assume that the DRes of i and j are q and p respectively; then,

$$\frac{d_{j,r_p}}{R_p}x_j = \frac{d_{i,r_q}}{R_q}x_i > \frac{d_{j,r_k}}{R_k}x_j > \frac{d_{i,r_k}}{R_k}x_i. \tag{11}$$

Here, let us consider two cases:

(a) if $\frac{d_{i,r_p}}{R_p} > \frac{d_{i,r_q}}{R_q}$, then $x_j < x_i$. In order to satisfy (11), $d_{j,r_k} \geqslant d_{i,r_k}$; hence, the demand of i for resource r_k is significantly lower than the corresponding demand of j. Thus, r_k is not critical to i, which contradicts the assumption.

(b) if $\frac{d_{i,r_p}}{R_p} < \frac{d_{i,r_q}}{R_q}$, then $x_j > x_i$. According to (11), $d_{j,r_k} \geqslant d_{i,r_k}$. Thus, similar to case (a), for node i, r_k is not more important to i than to j, which also contradicts the assumption.

Theorem 5. cbDRF satisfies the strategy-proofness property. The node cannot increase its share by lying about its resource demands.

Proof. Let us assume that node i can increase its share by using demand vector $\hat{\boldsymbol{D}}_i$ that is larger than the original $\boldsymbol{D}_i$; if the DRes of i is r_k, then $\hat{d}_{i,r_k} > d_{i,r_k}$. If the DRes of j is r_p, then $\frac{d_{i,r_k}}{R_k}x_i = \frac{d_{j,r_p}}{R_p}x_j$ and $\frac{\hat{d}_{i,r_k}}{R_k}x_i = \frac{d_{j,r_p}}{R_p}x_j$. $\frac{d_{j,r_p}}{R_p}x_j$ is invariable, and hence, we obtain $\frac{d_{i,r_k}}{R_k}x_i = \frac{\hat{d}_{i,r_k}}{R_k}x_i$, which contradicts $\hat{d}_{i,r_k} > d_{i,r_k}$.

Theorem 6. cbDRF satisfies the Pareto efficiency (PE) property.

Proof. Let us assume that node i can increase its share without reducing the share of the others nodes; then, according to the definition of PE, for node i, there exists a Pareto improvement (PI). According to Lemma 8 in [9], with DRF, there exists at least one saturated resource. Let us assume that node i increases its share from s_{i,r_k} to $\hat{s}_{i,r_k}$ for resource r_k; then, i cannot increase s_{i,r_k} by increasing d_{i,r_k} according to Theorem 5. Therefore, i must increase s_{i,r_k} by increasing the number of sub-tasks, x_i. However, from Lemma 8 in [9], i has at least one saturated resource w, that cannot be increased further by improving x_i. This contradicts our assumption, and thus, a PI to node i does not exist.

Theorem 6 provides a constraint for the amount of occupied resources of the nodes, implying that the resource is saturated and the node cannot improve its share without reducing the share of other node; however, such behavior is forbidden in the system.

In order to prevent the violation of fairness, Ref. [9] indicates that four basic properties should be satisfied in the allocation algorithm: sharing incentive, envy-freeness, strategy-proofness, and Pareto efficiency. Table 1 shows the comparison between DRF and cbDRF; cbDRF has two additional properties, release incentive and punitive allocation, to achieve fairness and security through a credit-based mechanism. These two properties ensure that nodes maximizing their own resource share by malicious behavior will be punished and their credit will be significantly decreased. Thus, eventually, malicious nodes cannot obtain any resource.

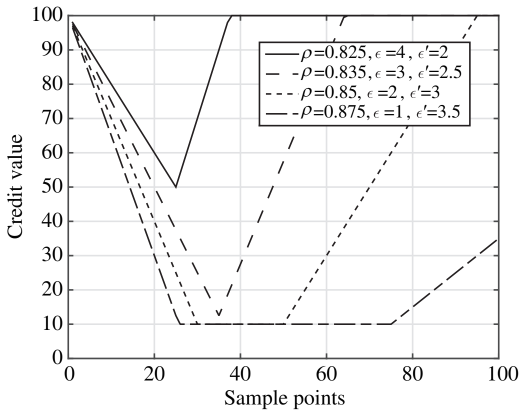

Figure 4 Credit variation for various values of ρ.

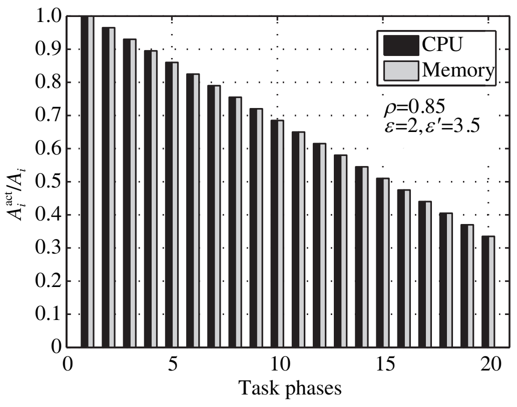

Figure 5 Credit variation when $\eta_{i,\tau_p} < \rho$.

3.2 Linear model analysis

In Subsection 2.3, Eq. (6) provides the relation between $\boldsymbol{C}$ and η based on threshold ρ. This equation can also be formulated as $\boldsymbol{C}_i = g(x)$ (considering node i as an example), where

$$g(x) = \boldsymbol{C}_{i,\tau_p} + x, \begin{cases} x = \epsilon, \iff 0 < \eta_{i,\tau} < \rho, \\ x = \epsilon', \iff \rho \leqslant \eta_{i,\tau} \leqslant 1, \end{cases} (\epsilon > 0, \epsilon' < 0). \tag{12}$$

Eq. (12) indicates that the choice of ρ and ϵ (or ϵ') determines the variation of $\boldsymbol{C}_i$.

Figure 4 shows the variation of the credit of the node for different ρ and ϵ (ϵ'), where $\boldsymbol{C}_0 = 100$ and η varies from 80 to 89.9 in increment of 0.1. Initially, $\eta < \rho$, and hence, the credit value continues to reduce until $\eta \geqslant \rho$. It can be observed that as ϵ' increases, $\boldsymbol{C}_i$ declines faster. Similarly, when $\eta \geqslant \rho$, $\boldsymbol{C}_i$ begins to increase with the speed ϵ.

Let us consider the special case where $\boldsymbol{C}_i = 0$ in period τ; thus, $\psi_\tau = 0$, and then, $A_\tau = 0$. Given that $\eta_\tau = \Re_\tau^{\text{act}}/A_\tau$, the denominator cannot be zero, hence, we will assign a minimal value $\boldsymbol{C}_{\min}$ for $\boldsymbol{C}_i$. In Figure 4, $\boldsymbol{C}_{\min} = 10$.

Next, we evaluate the linear model by simulation. We consider a system with 500 CPUs, 50000 units of memory, and a node i that executes tasks having random resource demands: $d_{i,\text{CPU}} \in [20, 100]$ and $d_{i,\text{Mem}} \in [300, 1000]$. We study two typical scenarios: (1) node i releases insufficient resources after each task phase; (2) node i releases the appropriate amount of resources at the end of every task phase.

Figure 5 shows that punitive allocation continuously imposed on node i when i does not release the appropriate amount of occupied resources for 20 tasks. In this figure, we choose $\rho = 0.85$, $\epsilon = 2$, $\epsilon' = -3.5$; the x-axis represents the tasks, and the y-axis represents the ratio of actual allocation to the theoretical allocation, defined as A_i^{act}/A_i, which can reveal the extent of punishment imposed on the share of node i. The credit factor in τ_p is $\psi_{i,\tau_p} = \boldsymbol{C}_i/\boldsymbol{C}_0$; hence, according to (4), the actual amount of share in τ_p is $s_{i,r_k,\tau_p}^{\text{act}} = d_{i,r_k}^{\tau_p} \cdot \lceil x_{i,\tau_p} \cdot \psi_{i,\tau_{p-1}} \rceil$, and the theoretical share is $s_{i,r_k}^{\tau_p} = d_{i,r_k}^{\tau_p} \cdot x_{i,\tau_p}$, where r_k denotes the resource type. Thus, the actual allocation can be denoted as $A_{i,\tau_p}^{\text{act}} = \sum_{k=1}^n s_{i,r_k,\tau_p}^{\text{act}}$, and the theoretical allocation is $A_{i,\tau_p} = \sum_{k=1}^n s_{i,r_k}^{\tau_p}$. Hence, A_i^{act}/A_i directly reflects the difference between the actual and theoretical amount of allocation. The CPU and memory use the same credit value, and therefore, they have the same rate of change for A_i^{act}/A_i.

The simulation shown in Figure 6 uses the same parameters as the simulation in Figure 5. In Figure 6, node i releases insufficient resources in first 10 tasks, and hence, its resource share decreases continuously. From the 11th task, node i begins to release the appropriate amount of the resource, and therefore, its share begins to increase. Given that $|\epsilon'| > |\epsilon|$, the rate of increase is less than the rate of decrease.

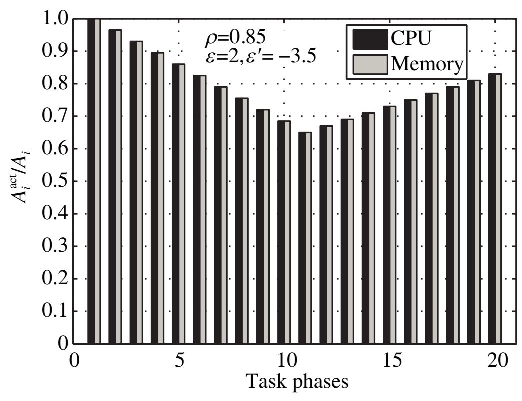

Figure 6 Credit variation for $\eta_{i,\tau_p} < \rho, p \in [1,10]$ to $\eta_{i,\tau_p} \geqslant \rho, p \in [11,20]$.

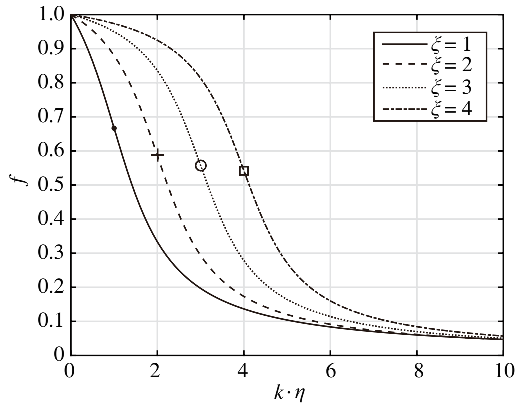

Figure 7 Credit curve for different ξ. The markers on each curve indicate the inflection point ($k\eta = \xi$) of the function for different ξ.

3.3 Variable gradient model analysis

In Subsection 2.4, Eq. (10) represents the variable gradient model for the node credit. The constant c indicates the upper bound of the credit value, hence, we will discuss $f(\eta)$ (Eq. (9)) in this subsection.

Figure 7 shows the curves of function f for different ξ. The markers on each curve indicate the inflection point of the function. As ξ increases, the curve moves to the positive side of the x-axis, while the top and the bottom of the curve show a gradual slope.

For further analysis of function f, we calculate its derivative as follows (ξ is a constant):

$$f'(\eta) = -k/\left\{\left[1 + (k\eta - \xi)^2\right] \cdot \left[\frac{\pi}{2} + \arctan(\xi)\right]\right\}. \tag{13}$$

We define $\mathcal{C} = \left[\frac{\pi}{2} + \arctan(\xi)\right]$, which is the constant term; then, Eq. (13) can be written as

$$f'(\eta) = -k/\left\{\mathcal{C}\left[1 + (k\eta - \xi)^2\right]\right\}. \tag{14}$$

In order to facilitate our analysis, we define $\chi(\eta) = -\frac{1}{\left[1+(k\eta-\xi)^2\right]}$; then, $f'(\eta) = \frac{k}{\mathcal{C}} \cdot \chi(\eta)$.

Figure 8 shows the curves of $\chi(\eta)$ for different ξ. Similar to Figure 7, the curve of the function moves to the positive side of the x-axis as ξ increases. Thus, $|\chi|$ reaches its maximal value at the point where $k\eta = \xi$, and this point is the inflection point of $\chi(\eta)$ (also the inflection point of $f'(\eta)$). We can easily obtain the rate of the credit variation from the inflection point:

(1) $\eta \in [0, \xi/k)$. When η approaches ξ/k, the punitive allocation becomes more severe, and the credit reaches its maximal decrease rate;

(2) $\eta \in (\xi/k, \infty)$. When η approaches ∞, the punitive allocation gradually reduces in severity, and we obtain $\lim_{\eta\to\infty} f(\eta) = 0$.

Conversely, when η approaches 0, It implies that the node releases the appropriate amount of resources, including all its occupied resources, and hence, f will increase.

Next, We simulate the resource allocation with the variable gradient model. We consider a system having 500 CPUs and 50000 memory units, and a node that runs tasks having random resource demands: $d_{i,\rm CPU} \in [20,100]$ and $d_{i,\rm Mem} \in [300,1000]$. The proportion of the released resource decreases from 80% to approximately 20%. We continue to use $A_i^{\rm act}/A_i$ to evaluate the allocation variation during the tasks of the node.

Figure 9 shows the decline in resource allocation when the amount of the released resource continues to decrease for different values of parameter ξ. We observe that when ξ increases, the slope of the curve increases. Thus, ξ controls the extent of punitive allocation imposed on the node. Initially, as the released amount decreases, the allocation declines slowly. However, as the released amount continuously decreases,

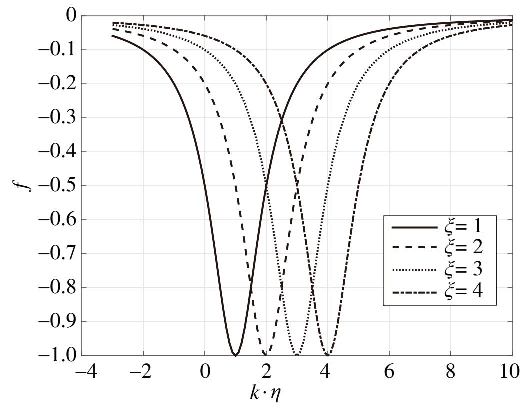

Figure 8 Curves of $\chi(\eta)$ for different ξ.

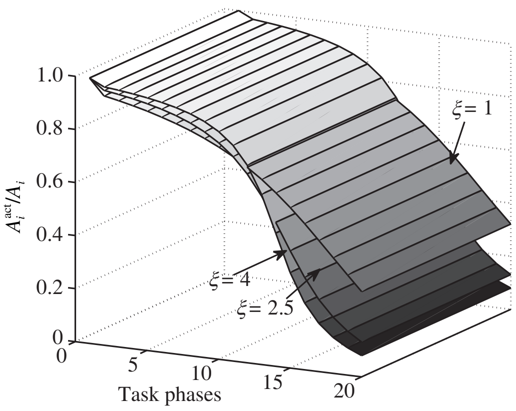

Figure 9 Resource allocation variation for different ξ.

the reduction in allocation becomes more severe. Finally, the rate of decline in allocation becomes lower, which will approaches zero at ∞.

From Figure 9, if a node does not release the appropriate amount of a resource several times, its credit will not decrease distinctly. However, a continuous reduction in the released amount will result in a rapid decline of the share of the node. Let us consider that the node executes a task sequence having different resource demands. When a task with greater resource demands ends, the node may not release an adequate amount of occupied resources compared to a light task having lesser resource demands. With the linear model, when $\eta < \rho$, the credit of the node will decrease ϵ'. Thus, the share of node can be severely affected (the same applies to the credit of the node) if many resource-intensive tasks exist in the task sequence. However, with the variable gradient model, the share of the node will not be distinctly decreased unless the node has a very low proportion of released resource in every task, and in that case, the node must be restrained for protecting the system. For example, if $\xi = k = 4$, then $f(0.875) \approx 0.7$. $\eta = 0.875$ implies that occupied resource of the node is greater than 80% of the actual released amount; in this case, the node continues to have approximately 70% of its initial credit (C_0).

3.4 Comparison with related work

In this subsection, our approaches are compared with several existing methods that use a credit mechanism for resource allocation, and the corresponding analysis is also provided.

In [16] (Shen2014), the authors proposes a simple approach to calculate the credit of a node, which is similar to our linear model. In this model, a factor $f_i = \Re_i^{\rm act}/A_i$ is introduced (to simplify our discussion, this factor is named "release-allocation ratio (RAR)"), where $\Re_i^{\rm act}$ and A_i denote the amounts of actually released resource and the allocated resource, respectively. Further, a threshold $0.8 \leqslant \alpha < 1$ is defined, and if $f_i > \alpha$, the credit of i, C_i, will increase by one credit unit. This approach is suitable only for the case described in Subsection 3.2.

In [15] (Satsiou2010[1]), the node credit is based on $\max\{\Re_i^{\rm act}/A_i\}$, and can be obtained as

$$C_i^k = \frac{\Re_i^{{\rm act},k}/A_i^k}{\max_{j\in N}\left\{\Re_j^{{\rm act},k}/A_j^k\right\}}. \tag{15}$$

In (15), k represents the kth time (or task) period and N denotes the set of nodes. In the best case, $\Re_i^{{\rm act},k}/A_i^k = \max_{j\in N}\{\Re_j^{{\rm act},k}/A_j^k\}$; thus, $C_i^k = 1$. Let $C_{\max}^k = \max_{j\in N}\{\Re_j^{{\rm act},k}/A_j^k\}$. Let us consider an extreme case, $C_{\max}^k = 1$, which implies that at least one node completely releases the occupied resource. Thus, the model is reduced to the one proposed in [16].

In (Satsiou2010[2]) [14], the node credit for resource allocation is calculated as follows:

$$C_i^k = \frac{1}{N_i^k}\sum_{\tau\in T^k}\frac{\Re_i^{{\rm act},\tau}}{A_i^\tau}. \tag{16}$$

N_i^k denotes the total number of resource request from node i until period k and T^k is the set of periods in which i requests the resource. $\Re_i^{\mathrm{act},\tau}/A_i^\tau$ is the RAR in the τth period, $\tau \in T^k$. Therefore, RAR indicates that whether i has released a sufficient amount of resource in the period τ. From (16), we observe that in [14], the node credit C_i^k is the arithmetic average of RAR during a set of periods (T^k), which can have any value between 0 and 1. The actual amount of resource allocated to i in the kth period will be determined by the credit value by $C_i^{k-1}A_i^k$. Hence, after calculating the credit in the period k , the RAR of the previous $k-1$ periods must be added; thus, the RAR in each period must be stored and the storage overhead can grow rapidly as the number of tasks increases.

In [18] (Tian2009), the authors present a credit evaluation method to establish a trust mechanism for resource allocation in grid computing; it can be described by the following equation:

$$C_i^k = \begin{cases} U_i^k, & k = 1, \\ \sum\limits_{n=1}^{k-1} (1-\alpha) C_i^n + \alpha U_i^k, & k \geqslant 2. \end{cases} \tag{17}$$

According to [18], an allocation can be treated as a transaction between a node and the resource scheduler. As shown in (17), C_i^k denotes the credit of i in the period k; U_i^k, which has a value between 0 and 1, is the evaluation value for the node i determined by the resource scheduler based on the quality of the kth transaction. α represents the weight of the recent transaction, and when $\alpha > 0.5$, it implies that the system scheduler places significant emphasis on the transaction quality. Eq. (17) shows that the node credit is closely related to the past $k-1$ periods, which is similar to the method in [14]. Hence, C_i^n, $(n = 1, \ldots, k)$ for each period must also be stored.

In [17] (Gupta2015), a credit-based bandwidth resource allocation method that uses a penalty factor is proposed. The credit model can be depicted in a simple manner as shown in (18):

$$C_i = \left(\frac{\Re_i^{\mathrm{act}}}{A_i}\right)^{1-\eta_i} \times \frac{R_i}{D_i} \times \frac{D_i}{\mathrm{DC}_i}. \tag{18}$$

From (18), the first term is the RAR with the exponent $1-\eta_i$, D_i represents the resource demand of i, and DC_i denotes the download capacity of i. η_i is the penalty factor and $\eta = 0$ and $\eta = 1$ correspond to the maximum and minimum penalty, respectively, for node i. The value of η_i is equal to the exponential moving average (EMA) of the ratio RAR [28]. DC_i is necessary only for the allocation of network resources, such as bandwidth. For other types of resources, such as CPU, memory, and hard disk, DC_i can be equal to D_i. Thus, Eq. (18) can be simplified as follows:

$$C_i = \left(\frac{\Re_i^{\mathrm{act}}}{K_i}\right)^{1-\eta_i} \times \frac{R_i}{D_i}. \tag{19}$$

In (19), we know that the factor $(\Re_i^{\mathrm{act}}/A_i)^{1-\eta_i}$ is used to control the value of the credit of a node. The most severe punishment can be imposed on a node when $\eta_i = 0$. Considering an SRD attack, if the malicious node retains a fraction of the resource after each task—for example, 5%—η_i will be constantly equal to 5% based on the EMA. Therefore, $1-\eta_i = 0.95$, thus resulting a fixed and light punishment to the malicious node.

Figure 10 shows the results of our approach compared with other representative research described in this section. Our simulation consists of 100 task phases and 10 nodes; the credit value of node i, $C_i \in [0,1]$, and its initial value is 1. The nodes will request a different amount of the resource in each task phase and will release the resource when the task completes. The 100 task phases are divided into two parts: (1) In the first 50 phases, the nodes randomly occupy a fraction of the resource without releasing a sufficient amount, thus potentially resulting in credit reduction; (2) In the next 50 phases, the system tries to allocate a small portion of the resource to the node in order to test the node behavior and determine whether the node releases an appropriate resource amount. Our approaches to evaluate the node credit include the linear model and variable gradient model (VG model); the parameter settings are $\rho = 0.85$ for the linear model and $\xi = 10, k = 10$ for the VG model.

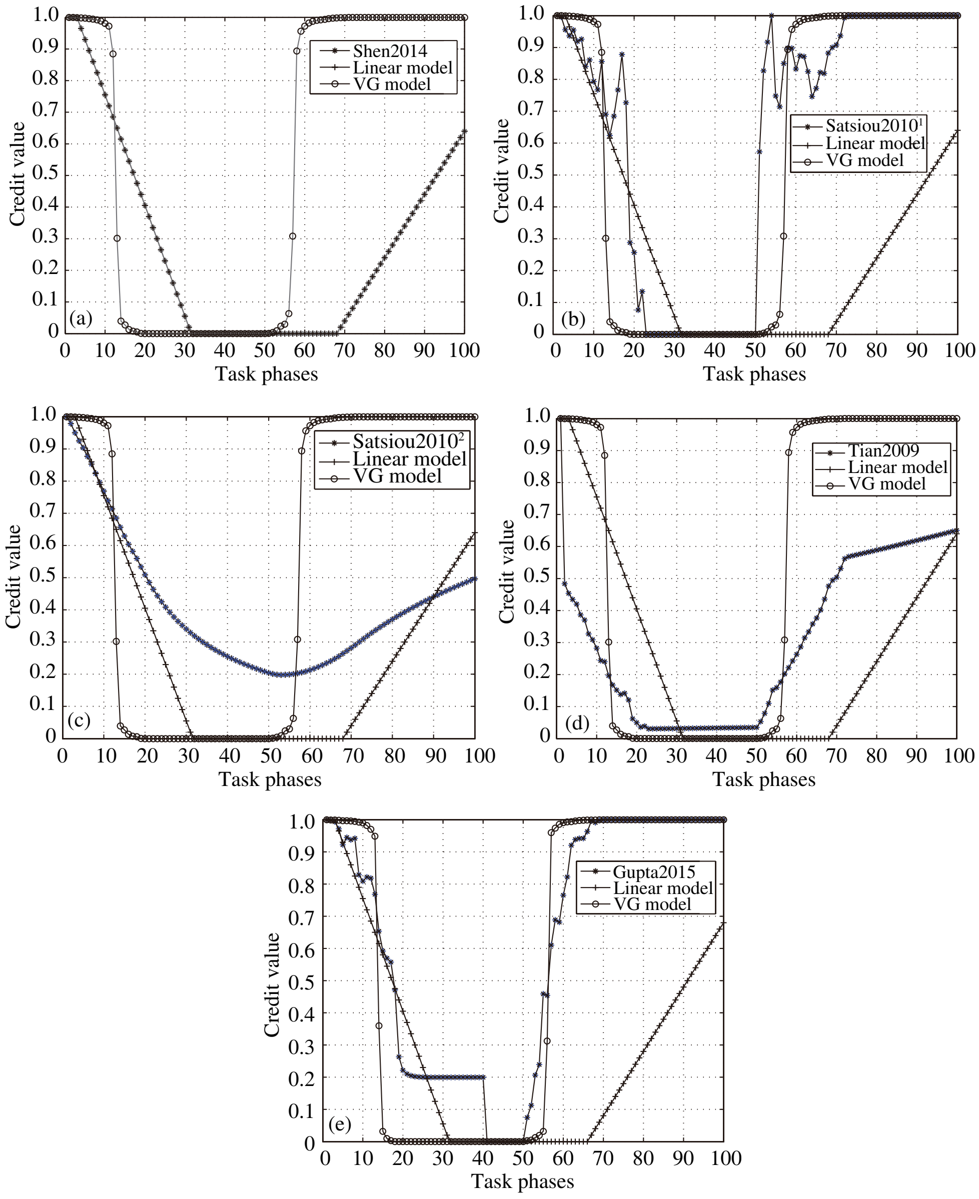

Figure 10 Comparison of our model with other representative approaches. (a) Shen2014; (b) Satsiou2010^1; (c) Satsiou2010^2; (d) Tian2009; (e) Gupta2015.

As shown in Figure 10(a), Shen2014 demonstrates the same characteristics as our linear model; the lines corresponding to both the methods are coincident. Thus, Shen2014 is equivalent to the linear model, which is applicable for the case in which a node executes the tasks with similar (or the same) resource demands (Subsection 3.2). The curve of the VG model indicates that for the first 10 task phases, with the increment corresponding to the occupied resource, the node credit shows no obvious reduction; however, from the 12th task phase, the node credit begins to distinctly decrease, thus demonstrating an increase in the severity of the punishment. With the continuous increments in the occupied resource amount, the node credit becomes zero, and the punishment attains a maximum. Thus, the node will be marked as malicious node, and will not be allocated any resources. From the 50th task phase, the system tries to allocate some amount of the resource to the node having zero credit, and the node begins to release a sufficient amount of the allocated resource. We can observe that the node credit begins to increase gradually, i.e., only if the node releases an occupied resource continuously and in a timely manner after each task, the credit can be restored to its initial value.

Figure 10(b) shows that the curve of Satsiou2010^1 [15] fluctuates during the task phases, thus reflecting an instability in the credit variation. This behavior is caused by the credit calculation method in [15]. Let us define $\mathrm{MAX}_{\mathrm{RAR}}^{k} = \max_{j\in N}\{\Re_j^{\mathrm{act},k}/A_j^k\}$, $\mathrm{RAR}_i^k = \Re_i^{\mathrm{act},k}/A_i^k$ and $\mathrm{MAX}_{\mathrm{RAR}}^{k+1} = \max_{j\in N}\{\Re_j^{\mathrm{act},k+1}/A_j^{k+1}\}$, $\mathrm{RAR}_i^{k+1} = \Re_i^{\mathrm{act},k+1}/A_i^{k+1}$. Let us consider $\mathrm{RAR}_i^k = \mathrm{RAR}_i^{k+1}$; if $\mathrm{MAX}_{\mathrm{RAR}}^{k} > \mathrm{MAX}_{\mathrm{RAR}}^{k+1}$, then $C_i^k < C_i^{k+1}$; however, if $\mathrm{MAX}_{\mathrm{RAR}}^{k} < \mathrm{MAX}_{\mathrm{RAR}}^{k+1}$, then $C_i^k > C_i^{k+1}$. The credit change is unstable, and hence, Satsiou2010^1 cannot effectively control malicious resource occupation.

From Figure 10(c), we observe that Satsiou2010^2 is able to control the credit according to the RAR. However, this approach cannot effectively reduce the credit when a malicious node continues to occupy an excessive amount of the resource. The minimal credit is approximately 0.2; hence, the malicious node can be allocated a resource even if the severity of the SRD increases. In Figure 10(d), let $U_i^k = \Re_i^{\mathrm{act},k}/A_i^k, \alpha = 0.5$; we observe that the curve of Tian2009 does not reach zero credit although the node did not release any resource. Similar to the case in Figure 10(c), a malicious node can obtain a small amount of resource even if the credit reaches a minimum (approximately 3.5%, according to Figure 10(d)). However, this approach is not suitable for the case described in Subsection 2.4; if the node fails to release a sufficient amount of the resource in a timely manner, this method results in distinct credit reduction and does not consider whether the node needs time to complete the release. This issue also exists in Satsiou2010^2 and Gupta2015 (Figure 10(e)). Additionally, according to the discussion above, Gupta2015 has another problem; if the node retains a constant fraction of the resource after each task, the credit value remains constant, and the punishment will not change. In Figure 10(e), from the 20th phase to the 40th phase, the RAR factor of the node is constant (approximately 0.15), and after the 40th phase, the credit value is set to zero. Thus, the credit value remains unchanged. In this case, this method cannot effectively and completely control the malicious resource occupation of the node.

3.5 Experimental results

In this subsection, we adopt OpenStack as the experimental platform to prove the effectiveness of our credit evaluation model. We use a six-server cluster to deploy the OpenStack cloud platform; each server has 16 GB memory and 12 Xeon$^{\mathrm{TM}}$CPUs with 24 cores. Four servers are used as “Compute” nodes, and one of them is selected to be the control center; for the remaining two servers, one is configured as a “Network” node, and the other as a “Storage” node. We have created various virtual machine flavors (VM templates) to satisfy the different resource demands of users. We adopt Ubuntu 12.04 as the operating system in the VM.

In our experiment, each user is assigned an initial credit. When the users need to execute some tasks, they attempt to request resources to build their own VMs for the execution of computational tasks. After placing a request for creating a VM, the user must submit the estimated time for utilizing the VM (estimated occupation time, EOT) to the platform. Thus, the control center will evaluate the credit of the user by monitoring whether the node releases an occupied resource in a timely manner. If the user occupies a resource for a long time and exceeds the submitted EOT (we refer to this time as “time exceeding EOT (TEEOT)”, which is equal to the difference between the actual end time and the EOT), the control center will reduce the user credit. Thus, the user cannot be allocated a sufficient amount of resource in the next request.

Figure 11 shows the resource allocation variation (RAV) for different credit evaluation approaches using the OpenStack platform. The x-axis represents the task phases and the y-axis indicates the amount of the allocated resource (AAR). Similar to the simulation in Figure 10, 100 task phases are divided into two parts: (1) in the first 50 phases, the node randomly increases its TEEOT, which can lead to credit reduction; (2) in the next 50 phases, the system tries to allocate a small amount of the resource to the node to test the user behavior and determine whether the user releases the resources in a timely manner after the completion of a task. In order to facilitate the comparison with the simulation results in Subsection 3.4, we allow the user to request the same amount of resources (CPU: 8 cores and Memory: 8 GB) in each task phase. Then, the user increases the TEEOT, thus possibly resulting in credit decrease and affecting the resource allocation in future task phases.

In Figure 11, we can observe that our linear model (Figure 11(a)) and Shen2014 (Figure 11(c)) demon-

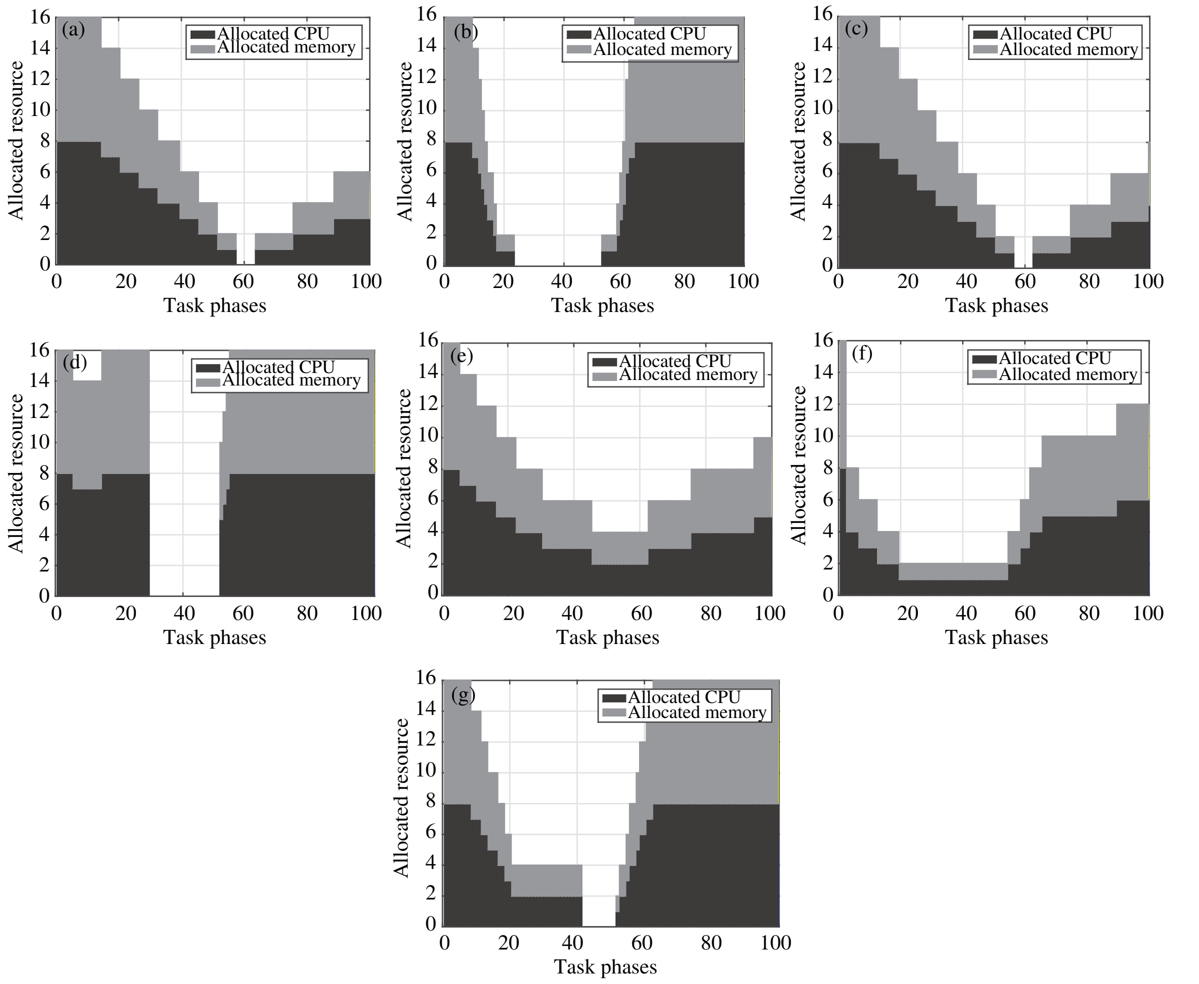

Figure 11 CPU and memory allocation variation for each credit evaluation approach. (a) Linear model; (b) variable gradient model; (c) Shen2014; (d) Satsiou2010^1; (e) Statsion2010^2; (f) Tian2009; (g) Gupta2015.

strate the same characteristics ($\rho = 0.7$ and credit increase/decrease unit is 1 and 2); thus, for equal parameters, Shen2014 is equivalent to our linear model. Figure 11(b) shows the resource allocation variation with the VG model for $k = 2, \xi = 2$. We observe that the AAR does not decrease rapidly at the beginning of the TEEOT increment; however, the AAR begins to decrease distinctly as the TEEOT continues to increase. Conversely, AAR will gradually increase when the TEEOT approaches zero, and rapidly increase when the TEEOT continues to be controlled to a proper value (equals to zero). Satsiou2010^1 continues to show unstable control on user credit (Figure 11(d)). With the increase of the TEEOT, the AAR can increase (10th task phase) or suddenly decrease to zero (from the 30th task phase). Thus, Satsiou2010^1 is ineffective in controlling the user credit. Figure 11(e) (Satsiou2010^2) demonstrates similar features as the curve in Figure 10(c) — the credit evaluation approach fails to effectively restrain the malicious use of a resource even if the TEEOT continues to increase. According to our discussion in Subsection 3.4, Tian2009 has a similar issue. From Figure 11(f), it can be observed that, even in the worst case, the user can obtain one core CPU and 1 GB memory. Thus, Tian2009 fails to control malicious use of resources. Figure 11(g) shows the AAR based on Gupta2015; it is obvious that when the user maintains a constant TEEOT, $1 - \eta$ in (19) will also be a constant, thus resulting in a light or fixed punishment on user credit. Hence, the AAR from the range of the 20th to the 40th task phase remains unchanged. The results proves that Gupta2015 cannot effectively control the malicious behavior of a user for a fixed TEEOT; this conclusion matches the simulation result in Figure 10(e).

4 Conclusion

In this paper, we proposes cbDRF, a credit-based model that provides a fairness-aware and security-aware mechanism for resource allocation. First, we describe the task-dependent model based on DRF. Then, we present the credit-based fairness model, cbDRF. This model can ensure fairness in allocation and can encourage the nodes to release occupied resources in a timely manner after completing tasks. Any nodes which exhibit malicious behavior related to the usage of resources, using methods such as SRD and the maximization of utility by occupying a resource for a long time period, will be penalized by punitive allocation. In a practical environment, both of the models (linear model and VG model) can be used simultaneously depending on the requirements of the platform. The simulation and experimental results reveal that cbDRF can inhibit nodes from occupying excessive resources without releasing them after use, and prove the effectiveness and correctness of our approaches.

Acknowledgements This work was supported in part by Key Program of NSFC-Guangdong Union Foundation (Grant No. U1135002), National High Technology Research and Development Program of China (863 Program) (Grant No. 2015AA011704) and Fundamental Research Funds for the Central Universities (Grant Nos. XJS15047, JB150308, JB150309).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Bertsekas D P, Gallager R G, Humblet P. Data Networks. New Jersey: Prentice-Hall International, 1992
2 Tan L, Pugh A C, Yin M. Rate-based congestion control in ATM switching networks using a recursive digital filter. Control Eng Practice, 2003, 11: 1171–1181
3 Massoulié L, Roberts J. Bandwidth sharing: objectives and algorithms. In: Proceedings of 18th Annual Joint Conference of the IEEE Computer and Communications Societies, New York, 1999. 1395–1403
4 Zukerman M, Tan L, Wang H, et al. Efficiency-fairness tradeoff in telecommunications networks. IEEE Commun Lett, 2005, 9: 643–645
5 Baruah S K, Cohen N K, Plaxton C G, et al. Proportionate progress: a notion of fairness in resource allocation. Algorithmica, 1996, 15: 600–625
6 Zhu D, Mossé D, Melhem R. Multiple-resource periodic scheduling problem: how much fairness is necessary? In: Real-Time Systems Symposium, Cancun, 2003. 142–151
7 Blanquer J M, Özden B. Fair queuing for aggregated multiple links. In: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM, 2001. 189–197
8 Liu Y, Knightly E. Opportunistic fair scheduling over multiple wireless channels. In: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications, San Francisco, 2003. 1106–1115
9 Ghodsi A, Zaharia M, Hindman B, et al. Dominant resource fairness: fair allocation of multiple resource types. In: Proceedings of the 8th USENIX Symposium on Networked System Design and Implementation, Boston, 2011. 323–336
10 Hu H, Li Z, Hu H. An anti-cheating bidding approach for resource allocation in cloud computing environments. J Comput Inf Syst, 2012, 8: 1641–1654
11 Zhou F, Goel M, Desnoyers P, et al. Scheduler vulnerabilities and coordinated attacks in cloud computing. J Comput Secur, 2013, 21: 533–559
12 Booth G, Soknacki A, Somayaji A. Cloud security: attacks and current defenses. In: Proceedings of the 8th Annual Symposium on Information Assurance, New York, 2013. 56
13 Lazri K, Laniepce S, Ben-Othman J. Reconsidering intrusion monitoring requirements in shared cloud platforms. In: Proceedings of the 8th International Conference on Availability, Reliability and Security, Salzburg, 2013. 630–637
14 Satsiou A, Tassiulas L. Reputation-based resource allocation in P2P systems of rational users. IEEE Trans Parall Distrib Syst, 2010, 21: 466–479
15 Satsiou A, Tassiulas L. Reputation-based internet sharing in wireless neighborhood community networks. In: Proceedings of International Conference on Communications, Cape Town, 2010. 1–5
16 Shen H, Liu G. An efficient and trustworthy resource sharing platform for collaborative cloud computing. IEEE Trans Parall Distrib Syst, 2014, 25: 862–875
17 Gupta R, Singha N, Singh Y N. Reputation based probabilistic resource allocation for avoiding free riding and formation of common interest groups in unstructured P2P networks. Peer-to-Peer Netw Appl, in press. doi: 10.1007/s12083-015-0389-0
18 Tian J, Yuan P, Lu Y. Security for resource allocation based on trust and reputation in computational economy model for grid. In: Proceedings of the 4th International Conference on Frontier of Computer Science and Technology, Shanghai, 2009. 339–345

19 Mashayekhy L, Grosu D. A reputation-based mechanism for dynamic virtual organization formation in grids. In: Proceedings of the 41st International Conference on Parallel Processing, Pittsburgh, 2012. 108–117

20 Bendahmane A, Essaaidi M, Moussaoui A E, et al. Tolerating malicious resources to ensure safe computations in grid systems. In: Proceedings of International Conference on Multimedia Computing and Systems, Ouarzazate, 2011. 1–6

21 Bawa R K, Sharma G. Reliable resource selection in grid environment. Int J Grid Comput Appl, 2012, 1: 1–10

22 Kaur D, SenGupta J. P2P trust and reputation model for securing grid resource management. In: Proceedings of International Conference on Advances in Engineering, Science and Management, Nagapattinam, 2012. 524–529

23 Bouchenak S, Chockler G, Chockler H, et al. Verifying cloud services: present and future. ACM SIGOPS Operat Syst Rev, 2013, 47: 6–19

24 Campegiani P. A genetic algorithm to solve the virtual machines resources allocation problem in multi-tier distributed systems. In: Proceedings of the 2nd International Workshop on Virtualization Performance: Analysis, Characterization, and Tools, Boston, 2009

25 Gu J, Hu J, Zhao T, et al. A new resource scheduling strategy based on genetic algorithm in cloud computing environment. J Comput, 2012, 7: 42–52

26 Teng Y L, Huang T, Liu Y Y, et al. Cooperative game approach for scheduling in two-virtual-antenna cellular networks with relay stations fairness consideration. China Commun, 2013, 10: 56–70

27 Joe-Wong C, Sen S, Lan T, et al. Multiresource allocation: fairness-efficiency tradeoffs in a unifying framework. IEEE/ACM Trans Netw, 2013, 21: 1785–1798

28 Gupta R, Singh Y N. Trust estimation in peer-to-peer network using BLUE. ArXiv:1304.1649, 2013