A Reinforcement Learning–Based Resource Allocation Scheme for Cloud Robotics

Article *in* IEEE Access · March 2018

DOI: 10.1109/ACCESS.2018.2814606

3 authors, including:

Kan Zheng
Beijing University of Posts and Telecommunications
375 PUBLICATIONS **11,851** CITATIONS

SEE PROFILE

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Reinforcement Learning-based Resource Allocation Scheme for Cloud Robotics

HANG LIU[1], SHIWEN LIU[1], AND KAN ZHENG[1], (Senior Member, IEEE)
[1]Intelligent Computing and Communication (IC[2]) Lab, Key Lab of Universal Wireless Communications, Ministry of Education, Beijing University of Posts & Telecommunications, Beijing, China, 100876

Corresponding author: K. ZHENG (zkan@bupt.edu.cn)

ABSTRACT In recent years, robotic systems combined with cloud computing capability have become an emerging topic of discussion in academic fields. The concept of cloud robotics allows the system to offload computing-intensive tasks from the robots to the cloud. An appropriate resource allocation scheme is necessary for the cloud computing service platform to efficiently allocate its computing resources when the robots send requests asking for computing service. This paper proposes a resource allocation scheme based on Reinforcement Learning (RL), which can make the cloud to decide whether a request should be accepted and how many resources are supposed to be allocated. The scheme realizes an autonomous management of computing resources through online learning, reduces human participation in scheme planning, and improves the overall utility of the system in the long run. Numerical results demonstrate that the proposed RL-based computing resource allocation scheme has better performances than the Greedy Allocation (GA) scheme.

INDEX TERMS Cloud Robotics, Reinforcement Learning, Resource Allocation

I. INTRODUCTION

Robotics is an academic field involving the design, control, application and maintenance of robotic systems. Robotic systems are currently widely used in scientific research, manufacturing technology, civil engineering, and space research, etc. [1]. In the past decades, researchers have been studying on how to develop robots that can helpfully support daily activities of humans and how robots can work in naturally and socially acceptable ways so that people can accept them as new members of our society [2]. In a robotic system, multiple robots may work together to perform activities. Challenges for researchers in this field are improving the operating efficiency and enhancing the cooperation of the robots. In [3], it is described that a networked robotic system includes a group of robots which are connected through a wired and/or wireless communication network and complete a task cooperatively. It is also pointed that cloud computing can provide a solution to extend the capabilities of networked robotics.

Cloud computing is a computing model used for allocating resources on demand, evolving from cluster computing. With cloud computing service, computing tasks do not run locally but are uploaded to the cloud server. Recently, researchers have been studying the robotic systems with cloud-computing infrastructure, since cloud computing has the advantages of powerful computing capability and an access to a large amount of data. Moreover, in a robotic system, a highly complex computation needed for processing a task can considerably drain the battery energy of a mobile robot. However, if these computing-intensive tasks can be processed in a computer with a stable power source, the energy of the robots can be saved and also computing efficiency can be improved [4].

By combining the computational power of the cloud computing and the availability of internet-connected devices in robotics, the term "Cloud Robotics" was proposed in 2010 [5]. As a newly emerging type of robotic systems, the cloud robotics introduces the architecture of "back-end cloud" together with "front-end robots". Robots simply connect to the cloud and deliver information to the cloud as needed, without having to store a large amount of information or having superb computing capabilities. Compared with conventional robotic systems, cloud robotics can significantly improve the storage capability and learning capability of the robots. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2814606, IEEE Access

resource sharing among the robots is also enhanced. More importantly, in cloud robotics, the computing-intensive tasks can be offloaded from the robots to the cloud, and hence the robots can have less pressure on energy consuming and enjoy a better computing performance. At the cloud side, a computing service platform helps the robots to handle these computing tasks. Moreover, the information transmissions from the robots to the cloud do not occur periodically as in a time-triggered manner. Instead, an information transmission is only performed when a robot has a task to be handled and sends a service request to the cloud, and this is considered as an event-triggered model.

As in a large scale system, there are numerous robots and hence a considerable number of service requests may be delivered to the cloud at a time. Therefore, how to efficiently schedule tasks is one of the key issues to be solved urgently in cloud robotics, which directly affects the availability of cloud resources and the execution efficiency of tasks. Good cloud resource allocation schemes which can bring high benefit for the system should be studied.

The contributions of this paper are described as follows.

1) In this paper, the computing resource allocation process in cloud robotics is formulated as a Semi-Markov Decision Process (SMDP) to achieve the optimal resource allocation scheme. Considering the system utility, the objective of the allocation scheme is to schedule the cloud computing resources to efficiently serve the robots in the system, and gain a high income for the system in the long term.
2) A Reinforcement Learning (RL) based resource allocation scheme is proposed. The proposed scheme comprehensively considers the characters of different types of information processing tasks and realizes an autonomous management of computing resources through online learning.
3) The effectiveness of the proposed scheme is verified by numerical analysis. The results show that the proposed scheme not only reduces the costs of human participation in scheme planning, but also improves the overall utility of the system.

The rest of this paper is organized as follows. Section II reviews the related work. In Section III, we describe the system model. Then, we define the states, actions, and rewards of the system respectively in Section IV. In Section V, we propose a cloud computing resource allocation scheme based on Reinforcement Learning. In Section VI, we give the numerical results and compare the performance of the proposed allocation scheme based on RL with that of the scheme based on GA. Conclusions and prospects are given in VII.

II. RELATED WORK

Since the concept of cloud robotics was proposed, it has aroused the interest of many IT companies and research institutes in relevant fields. In recent years, European scientists launched the RoboEarth program, which uses the Internet to create a huge, open-source web database that allows robots around the world to access and update information [6], and a RoboEarth semantic mapping system was described in [7]. Du *et al*. proposed the concept of Robot as a Service (RaaS) and the framework of robot cloud center [8]. Quintas *et al*. proposed a service system of robots based on Service Oriented Architecture (SOA), in which a group of robots shares knowledge with the intelligent space [9]. Furrer *et al*. designed and implemented a platform called ubiquitous network robot platform (UNR-RF), which is open source and standardized, and can provide a reference for other researchers in this field [10]. In [11], it was shown that running a vision based navigation assistance of a service mobile robot completely on an external cloud is feasible. In [12], the authors presented an architecture for cloud-based collaborative 3D mapping with low-cost robots.

As the scale of the system expands and the number of users increases, resource constrains and waste caused by inappropriate allocating and scheduling schemes become the main factors that constrain the capability of the cloud robotics. To solve the problems, researches on efficient task scheduling strategies in the cloud have been carried out widely and have achieved a lot at present.

In [13], a dynamic collaboration between local networked robots and remote public clouds was presented. In [14], a hierarchical auction-based mechanism called LQM auction for autonomous negotiation in cloud robotics was proposed. In [15], M. Tawfeek *et al*. proposed an ant colony algorithm, which has a good performance for resource allocation in task processing and solves the scheduling problem of limited resources in the mining supply chain. Xu *et al*. firstly introduced the Berg distribution model in the task scheduling algorithm, where the tasks are classified according to QoS, and then the appropriate scheduling algorithm was selected to achieve a fair distribution of resources [16]. In [17], a fuzzy clustering chaotic-based differential evolution algorithm proposed by Cheng *et al*. solves the resource-constrained scheduling problems.

III. SYSTEM MODEL

The architecture of the cloud robotics is shown in Fig.1, which is composed of robots and a cloud computing service platform. In the system, the robots perform functions like sensing the environment, processing information and actuating some work. When the information processing of the robots involves some complex and computing-intensive tasks, they may seek help from the computing powerful cloud. It is preferred that the computing tasks from the robots to be handled at the cloud side as "Cloud Execution", rather than being processed on the local robots as "Standalone Execution" [18].

When a robot needs to process the data it collects by sensing the environment, for example, it first sends a service request to the computing service platform at the cloud side. If the platform accepts the request, the robot uploads the data to the cloud for computation demand, otherwise, the processing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2814606, IEEE Access

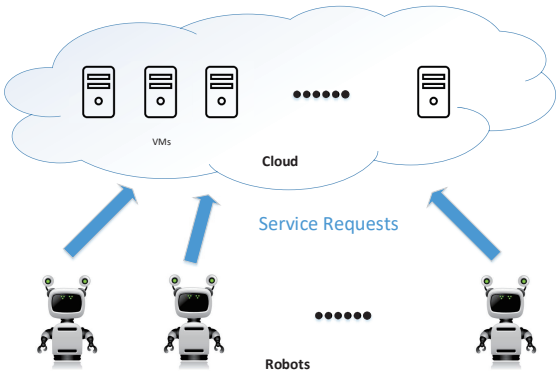

FIGURE 1: Architecture of cloud robotics

work has to be performed locally.

When the cloud computing service platform receives an information processing service request sent from the robot, the system needs to analyze the current load of computing resources at the cloud and then decide whether to accept the request or not, and if the request is accepted, the number of resources to be allocated to the service request should also be determined at the same time.

The concept of Virtual Machine (VM) is introduced in order to better describe the allocation scheme of cloud computing resources. A VM is the smallest unit in the resource allocation, which consists of the minimum amount of the hardware resource (e.g., CPU, hard disk and memory, etc.) required to handle a computing task. Moreover, a VM can only perform at most one computing task at a time, which means that the newly arrived task can be handled only when the performing of current task is completed.

Assume that the total number of VMs at the cloud computing service platform is M. In order to meet the demands of different types of information processing requests, we divide the requests from the robots into V types, each of which is denoted by v. We divide the number of resources allocated to a robot by the cloud into K levels, each is denoted by k, and the number of VMs of level k is n_k.

It should be noted that the focus of this paper is on the computing resource allocation in cloud robotics. As the processes of robots making requests and the cloud returning results of the allocation involve a small amount of data stream, the wireless communication between the robots and the cloud is not described in detail. Moreover, energy consumed constraints are critical to robots for they usually carry batteries with limited capacity. The extensive on-going energy efficient solutions of wireless communication technologies are categorized in [21] [22] [23].

IV. PROBLEM FORMATION

The process of the allocation of the cloud computing resources can be regarded as an SMDP. In the system, the more VMs are allocated to a computing task, the speed of the processing is faster and hence a higher utility can be gained. In the following of this section, the set of system states, the set of actions and the function of the reward signal are defined respectively.

A. SYSTEM STATES

The system states are defined as a set of two parts. One part denoted as L, is the number of different tasks currently running, where the tasks are distinguished by the type that it belongs to (i.e., v) as well as the level of the number of the resources it occupies (i.e., k). The other part denoted as e, is the event happening at the cloud service platform. Thus, the system state can be written in the following form:

$$S = \{s | s = \langle L, e \rangle\}, \tag{1}$$

where

$$L = \begin{bmatrix} l_{1,1} & \cdots & l_{1,K} \\ \vdots & \ddots & \vdots \\ l_{V,1} & \cdots & l_{V,K} \end{bmatrix}, \tag{2}$$

$$e \in \{R_1, \ldots, R_v, \ldots R_V, D_{1,1}, \ldots, D_{v,k}, \ldots, D_{V,K}\}, \tag{3}$$

and $l_{v,k}$ represents the number of running tasks which belongs to type v and occupies k-level of resources, R_v indicates that a service request of type v arrives, $D_{v,k}$ indicates that a service task of type v and level k is completed and the resources it occupies are released.

B. ACTION SPACE

As mentioned, when the cloud computing service platform receives a request (i.e., $e = R_v$), it decides whether to accept the request, and if so, the number of resources to be allocated is also determined. Hence, the set of actions in the system can be defined as

$$A(s) = \begin{cases} \{0, 1, \ldots, K\} & e \in \{R_1, \ldots, R_V\} \\ -1 & e \in \{D_{1,1}, \ldots D_{V,K}\} \end{cases}. \tag{4}$$

When a request is received, the possible set of actions can be selected is $a \in \{0, 1, \ldots, K\}$,where 0 in the set indicates that the request is refused, $k \in \{1, \ldots, K\}$ indicates that the request is accepted and k-level of computing resources is allocated. When a service task is completed, the action is $a = -1$, representing that the system only needs to release the resources occupied and change the system state, without doing any extra action.

C. REWARD FUNCTION

In the cloud robotics, different types of tasks have different demands on the information processing capabilities, and the returns they bring to the system are also different. Some service requests may require a small number of computing resources to bring a considerable income to the system, while some may require more computing resources to bring a big

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2814606, IEEE Access

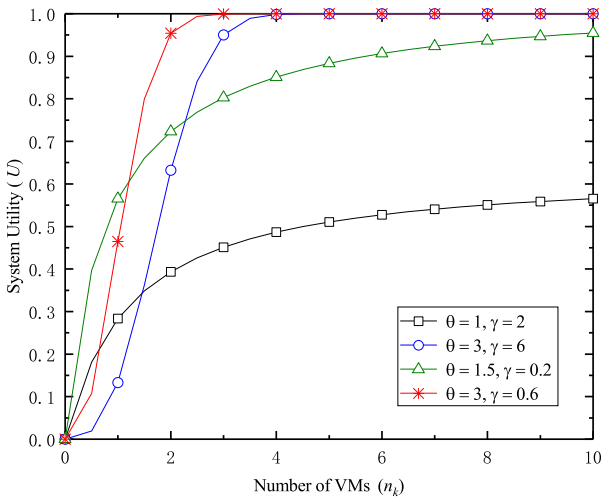

FIGURE 2: System utility function

income. In general, for a task, the more computing resources allocated, the better the information processing effect it has and hence the greater income it brings to the system. But the relationship between the number of resources allocated and the value of income is not simply in a linear manner. In order to reasonably characterize the relationship among the request type, the number of allocated resources and the system income, here a function in a Sigmoid form is applied to describe the relationship [19]:

$$U\left(n_k\right) = 1 - \exp\left(-\frac{n_k^{\theta}}{n_k + \gamma}\right), \tag{5}$$

where $U\left(n_k\right)$ is the system utility or system income, n_k is the number of VMs allocated, θ and γ are parameters for adjusting the form of $U\left(n_k\right)$ and they are usually determined by the type of the task. As shown in Fig.2 , since the selection of the parameters has a significant impact on the effect of the cloud computing service, it is important to design reasonable parameters for different types of service requests.

However, the processing of service tasks not only brings income to the system, but also brings some cost due to the occupancy of computing resources. Therefore, when the computing service platform at the cloud side processes a service task, the return to the system is composed of the system income as well as the cost. In addition, when the cloud computing platform rejects a service request, the information processing task has to be run locally, which cannot obtain a good processing effect and can heavily consume the resources of the robot (e.g., consuming a lot of battery charge, accelerating the aging of the robot device, etc.). From a cost reduction perspective, it is expected that the information processing tasks be done at the cloud side as much as possible. Thus, when defining the function of the reward signal, the reward for $a = 0$ is set to be a negative value, which means that if the system rejects a service request at a certain step, it gets a negative reward as a "punishment" and the cumulative rewards is reduced at this step.

Thus, for a given action, the immediate reward for state s can be written in this form:

$$r\left(s, s', a\right) = w\left(s, a\right) - \tau\left(s, s', a\right) c\left(s, a\right), \tag{6}$$

where $w\left(s, a\right)$ is the total system income when the system state is s and the action selected is a, and can be represented by applying the utility function $U\left(n_k\right)$:

$$w\left(s, a\right) = \begin{cases} U_v\left(n_k\right) & e = R_v, a = k \\ -\varepsilon & e \in \{R_1, \ldots, R_v\}, a = 0 \end{cases}, \tag{7}$$

and $\tau\left(s, s', a\right)$ is the time elapsed during the system state transition from s to s', with action a selected. $c\left(s, a\right)$ is the cost rate and can be described using the total number of VMs occupied.

$$c\left(s, a\right) = \alpha \sum_{v=1}^{V} \sum_{k=1}^{K} l_{v,k} n_k. \tag{8}$$

V. RL-BASED DYNAMIC COMPUTING RESOURCE ALLOCATION SCHEME

Based on the system model described in the previous section, a dynamic resource allocation scheme based on RL is designed in this section, in order to obtain a higher average return for the system in the long run. Different from the Dynamic Programming (DP) methodology, in this scheme, we do not need to know the state transition probabilities. The system adjusts itself automatically repsonding to the changes of the environment.

The Bellman optimal function is used to represent the relationship between the value function of the current state and that of the next state, when the optimal policy is selected.

$$V^*\left(x\right) = \max_{a \in A(x)} \left(r\left(x, a\right) - g^* \tau\left(x, a\right) + \sum_{y \in S} p\left(y|x, a\right) V^*\left(y\right) \right). \tag{9}$$

Change (9) into the form of state-action value function, i.e.,

$$R^*\left(x, a\right) = r\left(x, a\right) - g^* \tau\left(x, a\right) + \sum_{y \in S} p\left(y|x, a\right) \max_{b \in A(y)} R^*\left(y, b\right). \tag{10}$$

Then, the optimal policy of action selection can be determined in the following form:

$$\pi^*\left(x\right) = \arg\max_{a} R^*\left(x, a\right). \tag{11}$$

It is obvious that the optimal policy can be obtained as long as $R^*\left(\cdot, \cdot\right)$ is available. Thus the key point in designing the resource allocation scheme is to find $R^*\left(\cdot, \cdot\right)$. Conventionally, value iteration and policy iteration are the common methods used to derive $R^*\left(\cdot, \cdot\right)$. In these two method, the one step state-transition probabilities are needed, but these probabilities are difficult to acquire in large systems in reality. Therefore, an allocation scheme based on RL without the

Algorithm 1. RL-based dynamic computing resource allocation scheme

Initialization:

1: Decision making epoch: $m = 0$;
2: Action value function: $R_{old}(x,a) = R_{new}(x,a) = 0$, $\forall x \in S$, $a \in A(x)$;
3: Cumulative reward: $Q = 0$;
4: Time elapsed: $t = 0$;
5: Average return rate of the system: $g = 0$;
6: The number of times of updates of the value function for state x : $N(x) = 0$;
7: DCM parameters [20]: p_0, φ.

Step 1:

8: The cloud service platform detects that the system state at the current decision epoch is x.

Step 2:

9: Update DCM parameters according to DCM method: $p_m = \frac{p_0}{1+u}$, $u = \frac{m^2}{\varphi+m}$;
10: Select the action: Select $a = \arg\max R_{new}(x,a)$ with a big probability of $1 - p_m$, set $flag = 1$; Or randomly select an action in the action set $A(x)$ with a small probability of p_m, set $flag = 0$.

Step 3:

11: The system state is changed and if the next decision epoch has been reached, i.e., a new service request has been sent, it is detected that now the system state is y ;
12: The time elapsed during the transition from state x to state y is: $\tau(x,y,a)$;
13: The reward obtained during the transition from state x to state y is: $r(x,y,a)$;
14: Update the learning rate: $\alpha = \frac{1}{1+N(x)}$;
15: Update the value function for state x :

$$R_{new}(x,a) = (1-\alpha) R_{old}(x,a) + \alpha \left\{ r(x,y,a) - g\tau(x,y,a) + \max_{b \in A(y)} R_{old}(y,b) \right\};$$

16: IF $flag = 1$, proceed to Step 4, ELSE move to step 5 directly.

Step 4:

17: $N(x) = N(x) + 1$;
18: $Q = Q + r(x,y,a)$;
19: $t = t + \tau(x,y,a)$;
20: $g = \frac{Q}{t}$.

Step 5:

21: Set $R_{old}(x,a) \leftarrow R_{new}(x,a)$; $x \leftarrow y$;
22: $m = m + 1$;
23: Return to **Step 2**.

need for transition probabilities is proposed in this section. In this scheme, firstly Temporal-Difference (TD) method is applied to estimate $R^*(\cdot,\cdot)$, and then the allocation policy is determined due to the value of $R^*(\cdot,\cdot)$.

The detailed process is described in Algorithm 1.

VI. NUMERICAL RESULTS AND ANALYSIS

In this section, numerical results is provided to evaluate the performance of the proposed allocation scheme and compare it with the Greedy Allocation (GA) scheme. In the GA scheme, the cloud computing platform only considers the optimization of the data processing effects of the current request. The system always allocates the highest level of computing resources to the current task, so that the system can obtain a maximum reward at the current decision epoch.

The GA scheme is a local optimal algorithm as it becomes the global optimal solution under the condition of sufficient cloud computing resources. The global optimal solution can be obtained by DP such as value iteration and policy iteration. However, these approaches need the information of robots demand distribution, and have the polynomial complexity of $O(N^2)$, where N is the number of the states in the system [24]. In contrast, the proposed RL scheme in this paper has significantly reduced per-epoch computational complexity of $O(K)$ and does not need to know the information of robots demand distribution in previous. The RL scheme can achieve the optimal policy after several decision epochs. It requires obtaining the robots demand distribution through online learning, and hence the decision made during the pre-trial phase may not be optimal, causing the calculated average system return rate becoming slightly lower than the actual optimal value.

Comparing the overhead between the proposed scheme and the GA scheme, our scheme has only two more steps than the GA scheme, i.e., finding the maximum value function (Step 2 in Algorithm 1), and updating the value function of the currently selected action (Step 3 in Algorithm 1). Moreover, from the point of view of space complexity, the proposed RL scheme needs to occupy more memory than the GA scheme, due to the need of storing the lookup table of the value function. The size of the lookup table is proportional to the size of the state space. When the state space is too large, artificial neural networks can be applied to memorize and express value functions, which reduces the demand on storage, and hence reduces overhead.

The key parameters used in our analysis are provided in Table 1. We define $K = 3$ levels of numbers of cloud computing resources, corresponding to 3 levels of numbers of VMs allocated to a task, i.e., $n_1 = 1$, $n_2 = 2$ and $n_3 = 3$. It is assumed that there are $V = 2$ types of service requests from the robots, denoted as $v_1 = 1$ and $v_2 = 2$. The parameters of the utility function $U(n_k)$ function of these 2 types of tasks are θ_1, γ_1, θ_2, and γ_2. The arrival of these 2 types of requests are subject to Poisson distribution, with mean values $\lambda_1 = 0.6\lambda$ and $\lambda_2 = 0.4\lambda$, respectively. The running time for a task at the cloud is subject to an exponential distribution with a mean value of $1/\mu$. We set the arrival rate of requests in a range from 2 to 23, and the number of VMs in a range from 2 to 12, respectively. When the cloud rejects a task request, the system gets a negative reward $-\varepsilon$ as a "punishiment".

In the numerical results, we define $case_{v,k}$ representing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2814606, IEEE Access

TABLE 1: Key Parameters

Parameter	Value
Allocated resource level (K)	3
Request type (V)	2
Total resource number (M)	2 - 12
Request arrival rate (λ)	2 - 23
Request leaving rate (μ)	5
Reward for rejections ($-\varepsilon$)	-1
θ in $U(n_k)$ for type $v=1$ (θ_1)	1
θ in $U(n_k)$ for type $v=2$ (θ_2)	3
γ in $U(n_k)$ for type $v=1$ (γ_1)	2
γ in $U(n_k)$ for type $v=2$ (γ_2)	6
DCM parameter (p_0)	0.01
DCM parameter (φ)	10^{12}

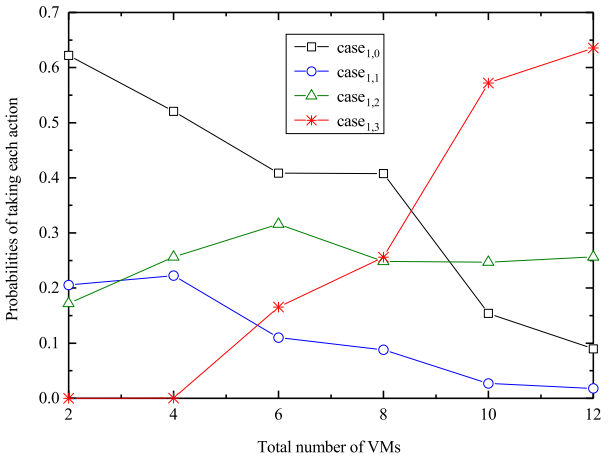

FIGURE 3: Probabilities of taking each action for service requests of $v=1$ under different numbers of total computing resources

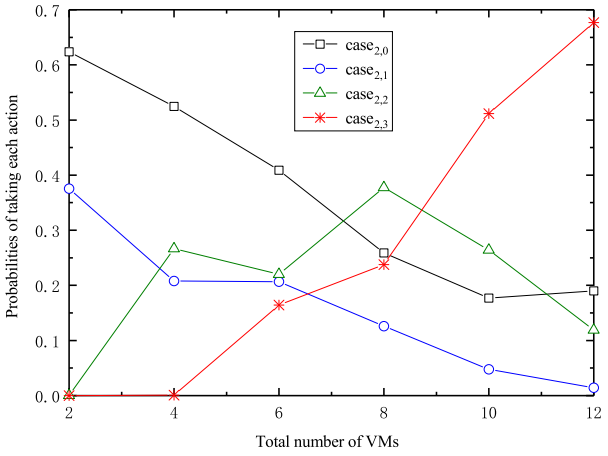

FIGURE 4: Probabilities of taking each action for service requests of $v=2$ under different numbers of total computing resources

that the system selects action k for a service request of type v, where $k=0$ indicates that the system rejects the service request and $k=1,2$, or 3 represents that the system accepts the request and allocates n_1, n_2, or n_3 VMs to the particular service task.

Fig.3 and Fig.4 show that when the average arrival rate of information processing requests is $\lambda=15$ in the RL-based resource allocation scheme, the probabilities of the system taking different actions for different types of tasks under different numbers of total computing resources. For tasks of type $v=1$, as the total number of VMs increases, the probability of $case_{1,3}$ increases gradually, while the probabilities of $case_{1,0}$ and $case_{1,1}$ show a downward trend, and the probability of $case_{1,2}$ increases at first and then decreases. For tasks of type $v=2$, the reward of allocating n_2 VMs is far greater than that of allocating n_1 VMs, while allocating n_3 VMs does not obviously increase the reward comparing to allocating n_2 VMs. Thus, when the total resources in the system are limited, the probability of $case_{2,2}$ goes up as the number of allocated VMs increases. However, when the resources are sufficient, the cloud computing platform tends to allocate n_3 VMs and hence the probability of $case_{2,2}$ begins to decrease.

Fig.5 illustrates the relationship between the total number of resources at the cloud side and the average system return rate. It can be seen that the average return of the system adopting the RL scheme is higher than that of the system adopting the GA scheme. This is because when an information processing request arrives at the cloud computing service platform, the GA scheme always allocates the highest level of cloud computing resources to the request. When the cloud has insufficient computing resources, the GA scheme has to reject the newly arrived requests. However, in the RL scheme, when a new request arrives at the cloud, the choice of the action is related to the type of the request. The system considers the reward that accepting the current request brings, as well as the expected return in the long-term. The RL-based scheme is more conservative and more reasonable in allocating computing resources than the GA-based scheme.

Fig.6 and Fig.7 show that when the total number of computing resources is $M=10$, the probabilities of the cloud taking different actions for different types of tasks under different arrival rates. When the arrival rate of the requests is relatively small, the information processing center at the cloud side has enough computing resources. In this case, the cloud computing platform tends to allocate as many VMs as possible to obtain a higher system reward for the current request. Therefore, the probabilities of $case_{1,3}$ and $case_{2,3}$ are the highest, and those of $case_{1,2}$, $case_{2,2}$, $case_{1,1}$ and $case_{2,1}$ are the second highest, and those of $case_{1,0}$ and $case_{2,0}$ are the lowest. However, with the increase of the arrival rate of service requests, the allocation strategy

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2814606, IEEE Access

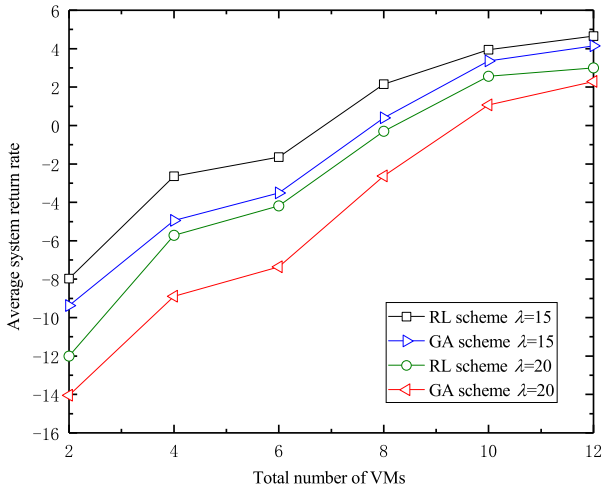

FIGURE 5: Average system return rate under different numbers of total computing resources

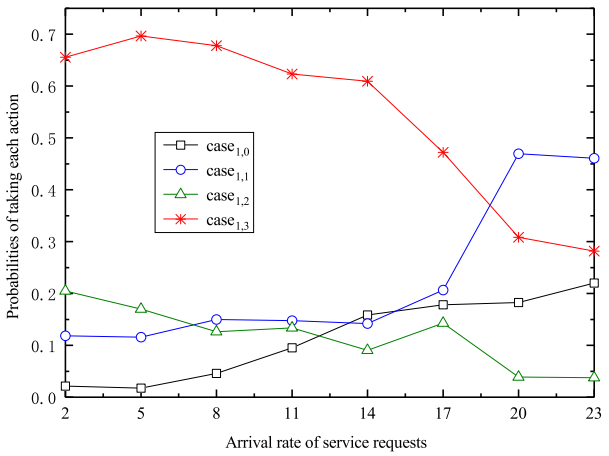

FIGURE 6: Probabilities of taking each action for service requests of $v = 1$ under different arrival rates of requests

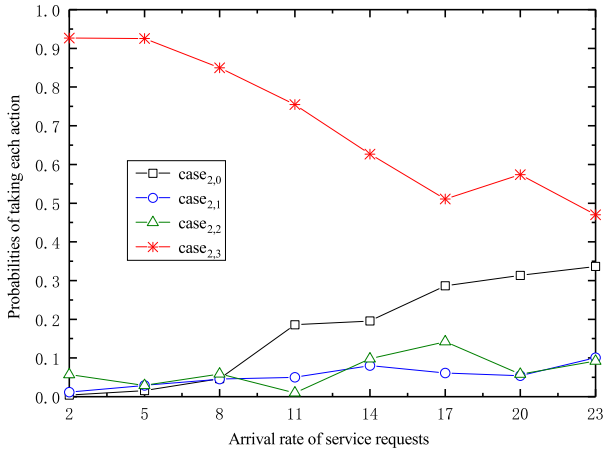

FIGURE 7: Probabilities of taking each action for service requests of $v = 2$ under different arrival rates of requests

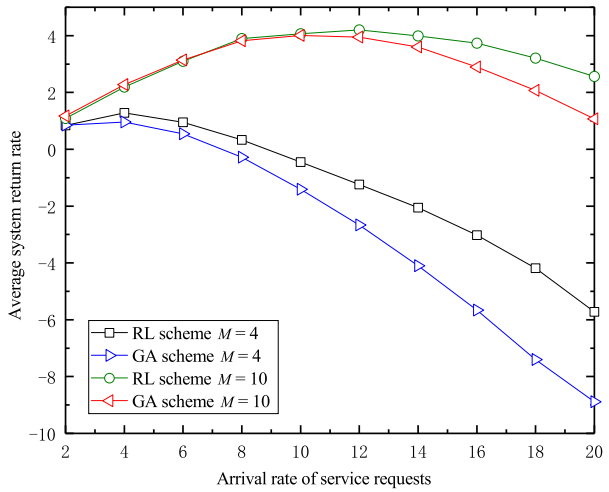

FIGURE 8: Average system return rate under different arrival rates of requests

is changed to become more conservative, and hence the probabilities of $case_{1,3}$ and $case_{2,3}$ begin to decline, and the probability of $case_{1,1}$ begins to increase. The probability of $case_{2,1}$ does not change obviously since when the cloud has only one VM, the reward for allocating it to a task of $v = 1$ is significantly greater than allocating it to a task of $v = 2$.

Fig.8 shows the relationship between the arrival rate of service requests and the average system return rate. As the arrival rate of requests increases, the average return rate of the system increases gradually. However, when the arrival rate becomes quite high, the average return rate of the system begins to decline. This is because the cloud computing resources are in short supply at this time, and the probability of the cloud refusing to offer service increases. At the same time, it is obvious that the performance of the RL-based allocation scheme proposed in this paper is superior to that of the GA-based scheme. The difference of the performance between the two schemes is not obvious when the arrival rate of the requests is low, since the information processing center at the cloud side has enough computing resources and tends to allocate as many resources as possible at this time, for both allocation schemes. However, when the arrival rate of service requests becomes higher, the average return rate of system in the RL-based allocation scheme is greater than that of the GA-based scheme.

VII. CONCLUSION

In this paper, Semi-Markov Decision Process is used to model the cloud computing resource management for cloud robotics. Considering both efficiency of information processing and energy saving of the system, this model gives a method to balance the return and cost. On this basis, we propose an RL-based dynamic resource allocation scheme. This scheme realizes an autonomous management of computing resources through online learning, reduces the cost of manpower participation in planning and maintenance, and

improves the overall utility of the system. Compared with the GA scheme, the RL scheme performs better under the condition of limited cloud computing resources and provides a larger average return for the system.

In future work, the research can furtherly focus on making the algorithm converge more quickly by adjusting the learning rate and other parameters in the algorithm, and optimizing the allocation result of the computing resources. It is also possible to design a more reasonable system utility function so that the research can be more suitably applied in practice.

ACKNOWLEDGMENT

This work was supported by the China Natural Science Funding under the grant 61671089, and the China Unicom Network Technology Research Institute.

REFERENCES

[1] L. Terrissa, R. Bouziane, and J. BrethÃl', "Towards a New Approach of Robot as a Service (RaaS) in Cloud Computing Paradigm," in *International Symposium in Knowldge Organization (ISKO2015)*, Nov., 2015.

[2] O. Demigha, W. Hidouci, and T. Ahmed, "Cloud Robotics: Architecture, Challenges and Applications," in *IEEE Network*, vol. 26, no. 3, pp. 28 - 34, May/June, 2012.

[3] G. Hu, W. Tay, and Y. Wen, "Cloud Networked Robotics," in *IEEE Network*, vol. 26, no. 3, pp. 21 - 28, May/June, 2012.

[4] R. Mateoa, "Scalable Adaptive Group Communication for Collaboration Framework of Cloud-Enabled Robots," in *17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems (KES2013)*, pp. 1239 - 1248, Sept., 2013.

[5] J. Kuffner, "Cloud-Enabled Robots," in *IEEE-RAS International Conference on Humanoid Robots*, 2010.

[6] M. Tenorth, U. Klank, D.Pangercic, and M.Beetz, "Web-Enabled Robots," in *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 58 - 68, June, 2011.

[7] L. Riazuelo, M. Tenorth, and D. Marco *et al.*, "RoboEarth Semantic Mapping: A Cloud Enabled Knowledge-Based Approach," in *IEEE Transactions on Automation Science and Engineering* , vol. 12, no. 2, pp. 432 - 443, Apr., 2015.

[8] Z. Du, W. Yang, Y. Chen, X. Sun, X. Wang, and C. Xu, "Design of a Robot Cloud Center," in *2011 10th International Symposium on Autonomous Decentralized Systems*, pp. 269 - 275, Mar., 2011.

[9] J. Quintas, P. Menezes, and J. Dias, "Cloud Robotics: Toward Context Aware Robotic Networks," in *2nd IASTED International Conference Robotics*, pp. 420 - 427, 2011.

[10] J. Furrer, K. Kamei, C. Sharma, T. Miyashita, and N. Hagita, "UNR-PF: An open-source platform for cloud networked robotic services," in *2012 IEEE/SICE International Symposium on System Integration (SII)*, pp. 945 - 950, Dec., 2012.

[11] J. Garcia, P. Blasco, F. Rio, and D. Muniz, "A Tradeoff Analysis of a Cloud-Based Robot Navigation Assistant Using Stereo Image Processing," in *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 444 - 454, Apr., 2015.

[12] G. Mohanarajah, V. Usenko, M. Singh, R. Andrea, and M. Waibel, "Cloud-Based Collaborative 3D Mapping in Real-Time With Low-Cost Robots," in *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 423 - 431, Apr., 2015.

[13] P. Pandey, D. Pompili, and J. Yi, "Dynamic Collaboration Between Networked Robots and Clouds in Resource-Constrained Environments," in *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 471 - 480, Apr., 2015.

[14] L. Wang, M. Liu, and M. Meng, "A Hierarchical Auction-Based Mechanism for Real-Time Resource Allocation in Cloud Robotic Systems," in *IEEE Transactions on Cybernetics*, vol. 47, no. 2, pp. 473 - 484, Feb., 2017.

[15] M. Tawfeek, A. El-Sisi, A. Keshk, and F.Torkey, "Cloud Task Scheduling Based on Ant Colony Optimization," in *2013 8th International Conference on Computer Engineering & Systems (ICCES)*, pp. 64 - 69, Nov., 2013.

[16] B. Xu, C. Zhao, E. Hu, and B. Hu, "Job Scheduling Algorithm Based on Berger Model in Cloud Environment," in *Advances in Engineering Software*, vol. 42, no. 7, pp. 419 - 425, 2011.

[17] M. Cheng, D. Tran., and Y. Wu, "Using a Fuzzy Clustering Chaotic-Based Differential Evolution with Serial Method to Solve Resource-Constrained Project Scheduling Problems," in *IEEE Transactions on Mobile Computing* , vol. 37, no. 1, pp. 88 - 97, 2011.

[18] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A Survey of Research on Cloud Robotics and Automation," in *IEEE Transactions on Automation Science and Engineering*, vol. 12, no.2, pp. 398 - 409, Jan., 2015.

[19] J. Lee, R. Mazumdar, and N. Shroff, "Non-Convex Optimization and Rate Control for Multi-Class Services in the Internet," in *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 827 - 840, 2005.

[20] C. Darken, J. Chang, and J. Moody, "Learning Rate Schedules for Faster Stochastic Gradient Search," in *Neural Networks for Signal Processing 1992 II., Proceedings of the 1992 IEEE-SP Workshop*, pp. 3 - 12, Aug./Sept., 1992.

[21] S. Zhang, Q. Wu, S. Xu., and G. Li, "Fundamental Green Tradeoffs: Progresses, Challenges, and impacts on 5G networks,"in *IEEE Communications Surveys and Tutorials*, vol. 19, no. 1, pp. 33 - 56, First quarter, 2017.

[22] K. Zheng, F. Liu, L. Lei, C. Lin, and Y. Jiang, "Stochastic Performance Analysis of A Wireless Finite-State Markov Channel," in *IEEE Transactions on Wireless Communications*, vol. 12, no. 2, pp. 782-793, 2013.

[23] F. Liu, K. Zheng, W. Xiang and H. Zhao, "Design and Performance Analysis of An Energy-Efficient Uplink Carrier Aggregation Scheme," in *IEEE Journal on Selected Areas in Communications*,vol. 32, no. 2, pp. 197-207, 2014.

[24] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, New York, NY, USA: Wiley, 2005.