Dynamic Task Scheduling Via Policy Iteration Scheduling Approach for Cloud Computing

Bin Hu[1,2], Ning Xie[3], Tingting Zhao[4] and Xiaotong Zhang[1]
[1] School of Computer and Communication Engineering, University of Science and Technology Beijing
Beijing, China
[2] College of Information Science and Technology,Bohai University
Jinzhou, China 121013
[e-mail: bhdxhb@163.com]
[3] School of Software Engineering,Tongji University
Shanghai, China
[4] Department of Computer Science and Technology,Tianjin University of Science and Technology
Tianjin, China
[e-mail: tingting@tust.edu.cn]
*Corresponding author: Tingting Zhao

Received March 3, 2016; revised June 25, 2016; revised August 19, 2016; accepted September 18, 2016; published March 31, 2017

Abstract

Dynamic task scheduling is one of the most popular research topics in the cloud computing field. The cloud scheduler dynamically provides VM resources to variable cloud tasks with different scheduling strategies in cloud computing. In this study, we utilized a valid model to describe the dynamic changes of both computing facilities (such as hardware updating) and request task queuing. We built a novel approach called Policy Iteration Scheduling (PIS) to globally optimize the independent task scheduling scheme and minimize the total execution time of priority tasks. We performed experiments with randomly generated cloud task sets and varied the performance of VM resources using Poisson distributions. The results show that PIS outperforms other popular schedulers in a typical cloud computing environment.

***Keywords*:** Cloud computing, resource scheduling, dynamic task scheduling, policy iteration

A preliminary version of this paper appeared in ISITC 2015, Cct.16-17,Tianjin, China. This version includes a concrete analysis and supporting implementation dynamic task scheduling. This research was supported by a research grant from National High Technology Research and Development Program 863. Research and applications in Collaborative Optimization and multi-link monitoring about metallurgical production process(No.2014AA041801-2).

https://doi.org/10.3837/tiis.2017.03.001 **ISSN : 1976-7277**

1. Introduction

Cloud computing involves internet-based ("cloud") development and the use of computer technology ("computing"). It is a style of computing in which resources are provided as an internet service for users who lack expertise or control over the technological infrastructure. It is a general concept that incorporates software as a service (SaaS), Web 2.0, and other recent, well-known technology trends to use the internet to satisfy the computing needs of users. A good example is Google Apps, which provides common business applications online that are accessed from a web browser while the software and data are stored elsewhere on Google servers [1]. The structure of cloud computing is usually described as consisting of three layers: Application, platform, and infrastructure. These services are delivered and consumed in real time over the internet. The top layer, SaaS, uses common resources and an application to meet multiple user demands simultaneously. Platform as a Service (PaaS) provides a services to software developers including developing, testing, deploying, and hosting. PaaS helps to speed up development progress as well as providing platform software tools and services. Infrastructure as a Service (IaaS) provides a computer infrastructure. Aside from providing computational capacity flexibility for end-users, IaaS also includes usage-based payment that allows end-users to pay as they go. The latest technologies on the cloud update in the IaaS layer. In IaaS, physical computing resources (such as CPUs, networks, and memories) are assigned, split, or dynamically resized into a large number of virtual machines which are provided to end-users to meet their demands. Employing these services can resolve several problems in regards to cloud data center power wastage and management. End-users are able to apply for their own virtual machines rather than actual physical resources, which prevents mutual influence among other users. Workload-sharing enlarges the resource pool and provides even more flexibility and cheaper resources.

The main goal of cloud computing is to improve the utilization of resources to meet the QoS needs of cloud users. The scheduling of user tasks in the cloud plays an important role in improving the performance of cloud services. Independent task scheduling is one of the more common cloud task scheduling techniques. The independent task scheduling process is depicted in **Fig. 1**.

The task scheduler is in charge of mapping user tasks to the most appropriate computing resource via different scheduling strategies. Its performance directly affects the efficiency of the entire cloud computing environment. Resource allocation and task scheduling have been extensively studied in terms of high-performance computing [2, 3] and in embedded systems [4, 5]. In cloud computing, however, task scheduling, automatic features, and resource allocation [6] require different algorithms in the IaaS. Many cloud management systems have adopted simple resource allocation strategies, for example, the greedy strategy adopted by Amazon EC2 [7]. Others abound, including the heuristic scheduling method that has been proposed for the Eucalyptus system, the priority queuing applied in OpenNebula, and the stochastic scheduling method employed in the OpenStack system. Unfortunately, these existing systems cannot globally improve the performance of virtual machines for cloud tasks.

In this paper, we focus on dynamic independent task scheduling. The two major contributions are:

1) A valid model describing the dynamic changes of both computing facilities (such as hardware-updating) and the task queue.

2) A novel algorithm based on the policy iteration that globally optimizes the independent task scheduling scheme.

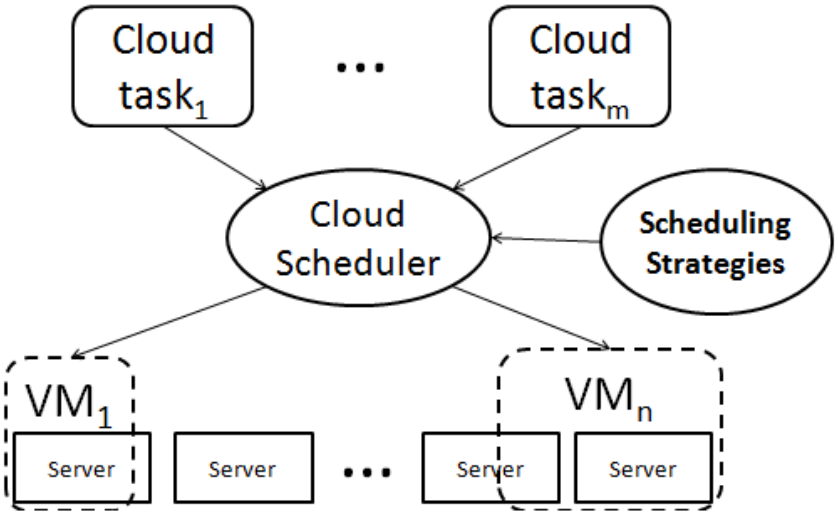

Fig. 1. Task scheduling model in cloud computing

2. Related Work

Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large swath of the IT industry and make software even more attractive as a service. It certainly shapes the way in which IT hardware is designed and purchased [7] and has become a wildly popular research object in recent years. In 2005, Amazon began to provide EC2 service over the internet to allow users to rent virtual computers on which to run their own computer applications; it also allows scalable deployment of applications by providing a web service by which a user can boot an Amazon Machine Image to create a virtual machine. These services mark important steps moving into the age of cloud computing. A number of public clouds are currently available for customers (and researchers,) including Amazon AWS, GoGrid, and Rankspace. Other companies such as Microsoft Azure, IBM SmartCloud, Google Cloud Computing, HP Cloud, Baidu Cloud, and Alibaba YunOS also provide cloud computing services. Open source cloud services have also been developed in recent years in effort to improve the research on cloud computing. One of the most well-known open source testbeds is called Open Cirrus [8], which consists of 14 data centers distributed across the globe. Essentially, it is a federated, heterogeneous cloud system. There are also many other open source systems including Eucalyptus, Open Nebula, and Reservoirs [9].

Visualization, which abstracts the coupling between hardware and software, is one of the most vital mechanisms in cloud computing. Rather than simply hide the internal details of a physical system, visualization actually uses an interfacing abstraction of real hardware resources or subsystems as a way to map the virtual resource to the actual resource; logical resources are abstracted from their underlying physical resources in order to improve agility, flexibility, and cost-reduction. Visualization is typically classified into six categories [10]: Full visualization, hardware assisted visualization, visualization, operating system level visualization, application level visualization, and network visualization.

For practical proposes, we have chosen to focus on full visualization, which can be treated as the mapping of a single physical resource to multiple logical representations, e.g., virtual machines (VMs). In the entire cloud computing environment, VMs can be dynamically created, expanded, shrunk, and moved as needed. Therefore, visualization is extremely well-suited to the dynamic cloud infrastructure by virtue of highly efficient features such as sharing capability, manageability, and isolation. Visualization has several benefits for enabling cloud computing [11, 12] including the following.

1) Functional execution isolation: The hypervisor handles the protection among VMs and, therefore, among the applications on different VMs. Users can be granted privileges within their VM without compromising isolation or host integrity.
2) A customized environment: Visualization enables the provisioning of highly specialized and customized environments that contain specific-purpose operating systems, libraries, and run-time execution environments. Visualization also offers functional isolation therefore enabling multiple views over the same physical hardware.
3) Easier management: Customized run-time environments can be started up, migrated, and shut down in a very flexible manner depending on the needs of whoever provides the underling hardware.
4) Consistency of legacy applications: VMs help to preserve binary compatibility in the run-time environments for legacy applications.
5) Testing and debugging parallel applications: Leveraging virtualized environments as a full distributed system may be emulated within a single physical host.
6) Enhancing reliability: Hypervisors and the live-migration capabilities can enhance the reliability of hosted virtualized applications, making them independent of the reliability of the underlying hardware in a seamless and transparent manner.

Nevertheless, the multi-tenancy nature of cloud computing along with its higher consolidation level constitutes several challenges yet to be properly mitigated. An increased level of sharing physical resources among multiple software components and applications to be hosted on behalf of different customers makes it exceedingly difficult to provide stable and predictable performance levels across the board. Indeed, VMs can be executed concurrently in a virtualized platform, simultaneously competing for physical resources that are scheduled by an underlying hypervisor. VMs and activities/tasks within VMs adjust to a hierarchical scheduling view where time can be partitioned among VMs. Within VMs, the processor is further granted tasks or threads according to the guest OS specific scheduling policy.

There have been numerous, valuable contributions to the literature in this regard. Emeneker et al., for example, proposed an image caching mechanism to reduce the overhead of loading disk images in a virtual machine [13]. Fallenbeck et al. used a dynamic approach to create a virtual cluster to manage the conflict between parallel and serial jobs [14]. Sadhasivam et al. [15] proposed a scheduling heuristics algorithm that can be incorporated at the data center level to select ideal hosts for VM creation; this implementation can be further extended to the host level by using an inter-VM scheduler for adaptive load balancing in the cloud. Gupta et al. presented the rid scheduling algorithm EDF-BF based on a QoS constraint, the CPU speed of the clusters [16], so that it can be employed by a grid scheduler. Liu et al. proposed a grid-based scheduling algorithm to manage load imbalance based on Min-Min in Grid [17]. Kiyarazm et al. presented an algorithm for task scheduling and load balancing in multi-processor systems based on the PSO method that can minimize the maximum span and the average utilization of all processors in an optimal manner [18]. All that being said, researchers have yet to address the VM performance problem in terms of managing dynamic

independent tasks. In this study, we modeled the dynamic independent task scheduling problem into a Markov decision processes (MDP) and built a PIS algorithm to optimize the task scheduling and load balancing.

The article aims to provide an approach of dynamic independent task scheduling for cloud resource scheduling, with contributions including:

1) A new scheduling model is designed for dynamic independent task scheduling.
2) A state-of-the-art approach to cloud scheduling with PIS algorithm.

3. Dynamic Independent Task Scheduling Approach

In this section, we model the dynamic independent task scheduling problem as an MDP problem. Typically, cloud computing architecture is categorized into three layers: The application layer, platform layer, and infrastructure layer (from top to bottom) [19]. We focus on user-oriented scheduling within the platform layer to provide the computing service for the service layer Users submit tasks to the cloud computing environment and receive executed results through the interface. The hypervisor creates the system description in two respects: The state of the user tasks and the state of the VMs. As a result, user tasks can be sent to the proper VM to obtain the required cloud computing resources via the dispatch of the hypervisor.

3.1 Formulation into Markov decision process

Both task scheduling and MDP in cloud computing have no aftereffect characteristics in the application, so we dynamically schedule the cloud task through the iterative and incremental updated scheduling strategy. We formulate the scheduling procedure as an MDP consisting of the tuple $(S, A, f, P_I, g, \gamma)$.The main symbols used in this paper are listed in **Table 1**.

Table 1. Symbol legend

Symbols	Definitions
S	A set of states
A	A set of actions
$f(s_{t+1} \mid s_t, a_t)$	The transition probability density from current state s_t to next state s_{t+1} while action a_t is taken
$P_I(S_0)$	The probability of initial states s_0
$g(s_t, a_t, s_{t+1})$	An immediate reward for transition from current state s_t to next state s_{t+1} by taking action a_t
γ	The discount factor for future rewards
π	The policy
$R(h)$	The discounted sum of future rewards
$Q_\pi(s,a)$	The conditional expected cumulative rewards of taking action a in state s under the policy π
π^*	The optimal policy
e_i	The ith Cloud task

Policy is the core of reinforcement learning (RL), which defines the learning agent's way of behaving at a given time *t*. Stochastic policy incorporates exploratory actions, exploration is usually required for securing an optimal policy in the learning process. We used the stochastic

policy in this study, which maps states distributed over the action space and represents the conditional probability density of taking action a_t in state s_t, as in:

$$a_{t+1} \sim \pi(a_t | s_t) \tag{1}$$

The dynamics procedures of the MDP are as follows. Initially, the agent starts from a randomly selected state s_1 following the initial state probability density $P_I(S_0)$ and chooses an action a_1 based on the policy π. The agent then makes a transition following the dynamics of the environment $P(s_2 | s_1, a_1)$. The transition is repeated T times to obtain a trajectory, which is denoted as $h=[s_1, a_1, \ldots, s_T, a_T]$. The return (i.e., the discounted sum of future rewards) along h is given by:

$$R(h) = \sum_{t=1}^{T} \gamma^{t-1} g(s_t, a_t, s_{t+1}) \tag{2}$$

Policy iteration is a popular and well-researched approach to RL. The key idea is to determine policies based on value function. The state-action value function $Q_\pi(s,a)$ for the policy π is defined as the conditional expected cumulative rewards of taking action a in state s under the policy π:

$$\begin{aligned} Q_\pi(s,a) &= E_{\pi,P_T}\left\{\sum_{t=1}^{\infty} \gamma^{t-1} g(s_t, a_t, s_{t+1}) \middle| s_1 = s, a_1 = a\right\} \\ &= E_{\pi,P_T}\{R(h)\} \end{aligned} \tag{3}$$

where $E\pi$ and P_T denote the conditional expectation over the $\{s_t, a_t\}_{t=1}^{\infty}$ following the policy $\pi(a_t|s_t)$ and transition $P_T(s_{t+1} | S_t, a_t)$ starting from $s_1 = s$ and $a_1 = a$. The optimal state-action value function is $Q_\pi^*(s,a) = \max_\pi Q_\pi(s,a)$. The goal of reinforcement learning is to find the optimal policy π^* that maximizes the state-action value function:

$$\pi^*(a|s) = \arg\max_a Q_\pi^*(s,a) \tag{4}$$

Since the optimal state-action value function is unknown, it is necessary to evaluate $\hat{Q}_{\pi l}(s,a)$ for the policy π and then update the policy based on the current evaluated value $\hat{Q}_{\pi l}(s,a)$.

3.2 PIS algorithm

In a practical cloud computing environment, task scheduling is always a large-scale problem. It is impossible to directly calculate the state-action value function for all the possible state-action pairs, thus the approximation of the state-action value function is required. We employed an efficient approach to evaluate the value function, i.e., Least Squares Policy Iteration (LSPI) [20]. The policy update rule is given by:

$$\pi_{l+1}(a|s) = \frac{\exp(\hat{Q}_{\pi l}(s,a)/\tau)}{\sum_a \exp(\hat{Q}_{\pi l}(s,a)/\tau)} \tag{5}$$

where τ is a positive parameter determining the randomness of the new policy $\pi_{l+1}(a|s)$.

Finally, the algorithm can be presented through the following steps:

1) Initialize policy $\pi_1(a|s)$.

2) Policy evaluation: Compute the state-action value function $\hat{Q}_{\pi l}(s,a)$ for the current policy $\pi_l(a|s)$ based on LSPI.

3) Policy improvement: Update the stochastic policy via Eq. (5).

4) Repeat steps (2) and (3) until the policy converges.

In order to monitor the situation of the entire environment, we designed the state including the task information S_e and the virtual machine information S_{vm}. The set of tasks is $E=\{e_0,e_1,...,e_{n-1}\}$, where the e_i denotes the ith cloud task, and the number of the tasks is N. More specifically, the state space of our tasks is expressed by five features, $S_e=(ID_e,RR_e,STA_e,DATA_e,PIR_e,PRI_e)$, where ID_e is the task identification number waiting to be assigned to some VM.

RR_e: The required resource for each task, $RR_e=\{RC_e^{(0)},RC_e^{(1)},...,RC_e^{(k-1)}\}$, where RC_k is the description of the k-th resource.

STA_e: The description of the task.

$STA_e=\{Free_e,Allo_e,Sche_e,Wait_e,Exec_e,Comp_e\}$.

$DATA_e$: The relative data, $DATA_e=\{C_e,I_e,O_e\}$, where C_e is the computational amount; I_e is the input data; and O_e is the output data.

PIR_e: The pair of task and its running virtual machine, $PIR_e=\{ID_e,ID_{vm}\}$.

PRI_e: The priority of the tasks, which is binary value *0* or *1* represented as a vector $(PRI_0,PRI_1,...,PRI_K)$.

The VM information can be described somewhat differently, as well. The set of the VM is defined as $VM=(vm_0,vm_1,...,vm_{M-1})$, where vm_j indicates the jth VM, and the number of the VM is M. Our VM state space is expressed by three features: $S_{vm}=(ID_{vm},TCAP_{vm},FCAP_{vm},DATA_{vm})$, where ID_{vm} is the VM identification number and $TCAP_{vm}$ is the total service ability that the VM can provide, which can be represented as $TCAP=\{TC_0,TC_1,...,TC_K\}$, where K is the resource type number and TC_k is the service ability of each resource (e.g., VM). Additionally, $FCAP_{vm}$ is the available resource of the VM, $FCAP_{vm}=\{FC_0,FC_1,...,FC_K\}$. $DATA_{vm}$ is the relative data of the VM, $\{IB_{vm},OB_{vm}\}$, where vm is the vector to represent the status of all K VMs such as $(vm_0,...,vm_K)$.

In our model, the action of the hypervisor in the cloud is to assign tasks to the VM at the current step. The action is calculated from the task scheduling policy of the hypervisor implemented by the PIS algorithm $a\leftarrow\pi^*(s)$, where a is the action represented as $ID_e\rightarrow ID_{vm}$, π^* is the optimal policy for the task scheduling, and S is the state in the MDP model. Task scheduling is naturally formulated as an MDP.

The goal of task scheduling is to maximize workload throughput. We define the reward function by the entire time necessary to complete the task by degree. The reward function is:

$$R(h)=\sum_{i=1}^{K}\gamma^{i-1}g(s_i,a_i,s_{i+1}) \tag{6}$$

The function $g(s_i,a_i,s_{i+1})$ is denoted as $g(s_i,a_i,s_{i+1})=(\alpha_1 t_{ex}+\alpha_2 t_{wa})^{-\rho^{PRI_{e_i}}}$, where a_1 and a_2 are the coefficients, t_{ex} is the task execution time, and t_{wa} is the task waiting time. ρ is the priority parameter and PRI_{e_i} is the state identification of the *i*-th task e_i.

4. Experimental Results and Analysis

4.1 Experimental settings

In this experiment, we adopted MIPS (Million Instruction Per Second) to represent the computing ability of the VMs and the load of tasks. We used some parameter settings of the cloud simulator *cloudsim*, as shown in **Table 2** [21]. The computation workload of the task is from 10,000 to 50,000 *MIPS*. For PIS implementation, we initially set the task scheduling policy as uniform distribution, i.e., with the same probability of choosing each machine. In the LSPI algorithm, the approximated value function $\hat{Q}_\pi(s,a)$ is expressed as a linear parametric combination of the basis function $\hat{Q}_\pi(s,a|\omega)=\omega^T\phi(s,a)$, where $\varphi(s,a)$ is the k = 15 dimensional Gaussian basis function vector:

$$\phi(s,a)=(\phi_1(s,a),\phi_2(s,a),...,\phi_k(s,a))^T \quad , \phi_i(s,a)=\exp(-\frac{\left\|s-s_{c_i}\right\|_2+\left\|a-a_{c_i}\right\|_2}{2\sigma^2}) \tag{7}$$

The $\sigma=0.5$ and $\{(s_{c_i},a_{c_i})\}_{i=1}^{k}$ were randomly generated from the trajectory sample as the centers of the Gaussian kernel. Note that the hyper parameters k and σ were chosen based on our preliminary experiments, and may have affected the performance. The parameter of τ in Eq. (1) was set to 0.9, and the discount factor γ to 0.95. The maximum number of policy update iterations was set to 100.

Table 2. Cloud simulator parameters

Type	Parameters	Value
Datacenter	Number of datacenters Number of hosts	1 5
Virtual Machine (VM)	Total number of VMs Number of PE per VM VM memory(RAM)	24 1 2048(MB)
Cloud task	Bandwidth Type of manager Total number of tasks Priority level of tasks VM Length of task	1000bit Time shared 50 1/0 10000-50000MI
Parameters setting of IGA	Number of initial individuals P_c P_m	250 0.2 0.005

4.2 Same VMs and different loads

In the first part of the experiment, we used a total of 5 heterogeneous VMs to 5 hosts with a set of 10-50 different tasks; the results are depicted in **Fig. 2** and **Fig. 3**. The whole completion time of tasks in PIS was shorter than that of other algorithms. In effect, the PIS scheduling strategy can be used to enhance resource efficiency in cloud computing feasibly and effectively.

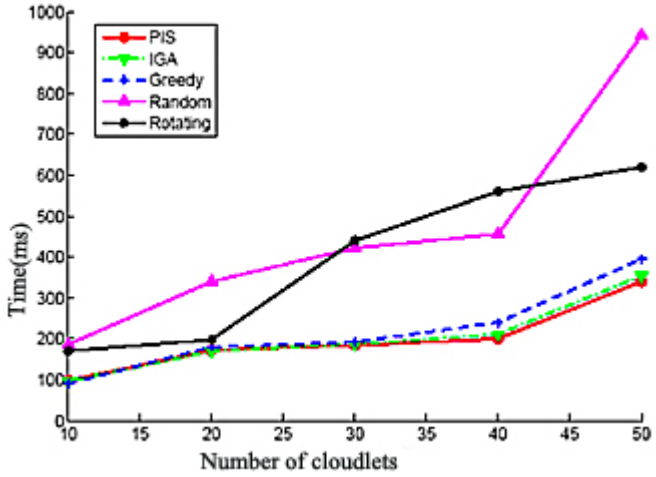

Fig. 2. Whole completion time under the same VMs and various loads

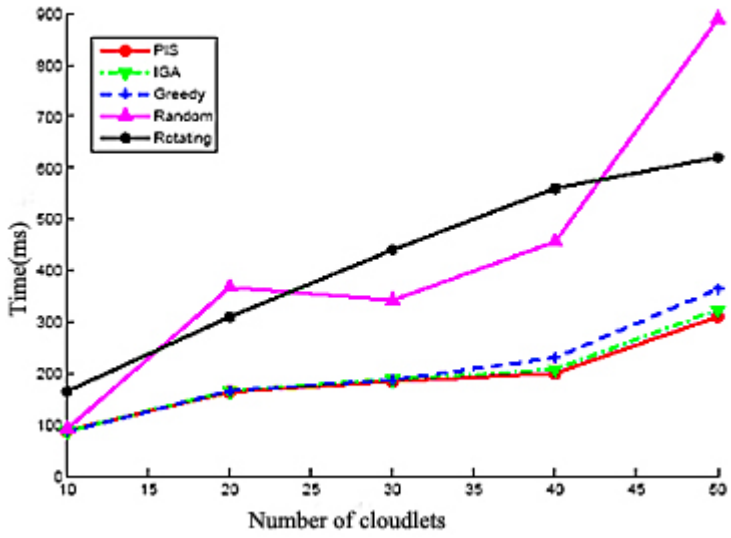

Fig. 3. Completion time of priority task under the same VMs and various loads

4.3 Same tasks and different VMs

As the number of VMs often changes in a real cloud environment, we adjusted the number and allowed each to perform differently. The results included the completion time of these tasks executed by different algorithms, average completion time, and standard deviation. As shown in **Fig. 4**, the priority task was shorter than that in other algorithms; the completion time of the whole task was similar to the completion time of the priority task. In effect, PIS

outperformed the other algorithms under various conditions.

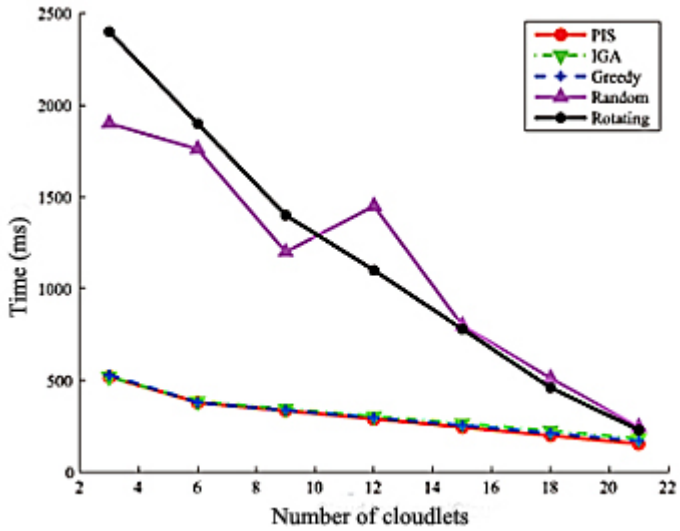

Fig. 4. Completion time of priority task under the same load and various VMs

4.4 Extremely unbalanced load

Taking into account the variation of different loads and VMs, we expanded the load and computing power to repeat the experiment. Tasks are quite different, and the performance of the VMs is also totally different. As shown in **Fig. 5**, the completion time of the whole task and priority were shorter than other algorithms. Our scheduling approach also proved robust in efficiency regardless of variations in workload pattern. The operating efficiency of the PIS algorithm is better than other algorithms in a complex cloud environment.

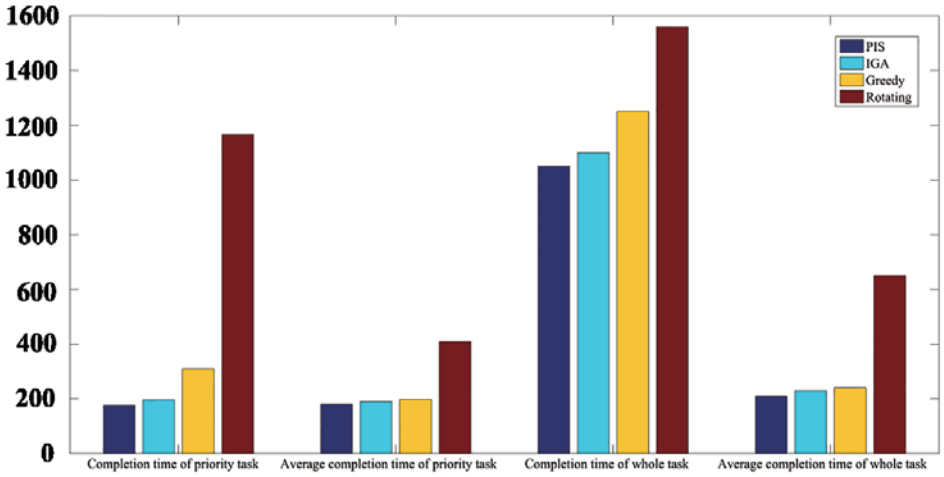

Fig. 5. Completion time of priority task under extremely unbalanced load and various VMs

4.5 Comparison among GA in iteration times and convergence

In the following experiments, we compared PIS with standard GA in two aspects: Iteration time and variance of maximum fitness function. Newer mechanisms worked faster (about

50%) by number of iterations, as shown in **Fig. 6**. The variance of the maximum fitness was smaller by about 80%. These observations suggest that PIS has better convergence and robustness in searching for feasible/optimum solutions with a reasonable timeframe and number of iterations. The PIS scheduling algorithm also outperformed the standard GA in regards to convergence. Accordingly, the scheduling strategy based on PIS is well-suited to resource scheduling in a practical cloud computing environment.

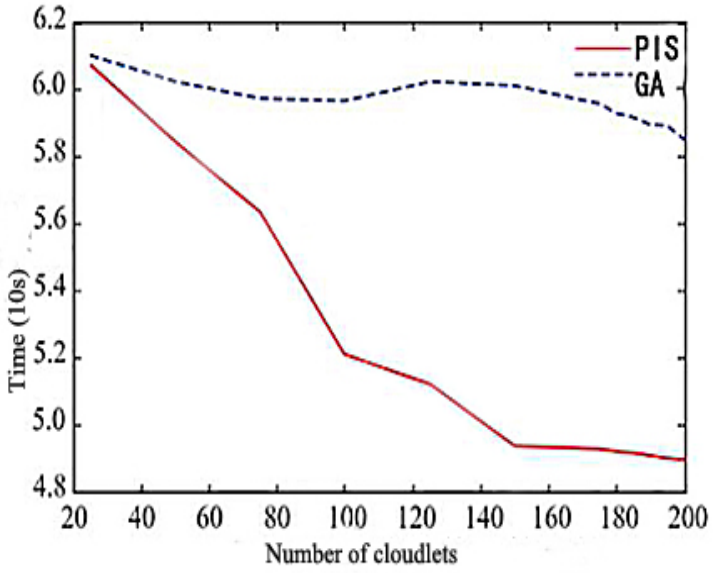

Fig. 6. PIS compared with GA in iteration times and convergence

5. Conclusion

In this paper, we present a dynamic task scheduling approach called the PIS strategy. Experimental results showed that PIS enables the user to solve the dynamic resource scheduling problem in a typical cloud computing environment. Our contributions include a valid model to describe the dynamic changes of both computing facilities (such as hardware updating) and request task queuing, as well as establishing the PIS which globally optimizes the independent task scheduling scheme for maximizing the utility of cloud computing resources. We demonstrated that our method can secure optimal scheduling effects compared to similar, pre-existing methods. In the future, we plan to address the more challenging topic of scheduling tasks by relevance.

References

[1] Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Computing Surveys*, vol.47, no.4, pp.47-63, July 2015. Article(CrossRefLink)

[2] João Nuno Silva, Luís Veiga, and Paulo Ferreira, "Heuristic for resources allocation on utility computing infrastructures," in *Proc. of the 6th international workshop on Middleware for grid computing* (MGC '08). ACM, New York, NY, USA, Article 9, pp.1-6, 2008. Article(CrossRefLink)

[3] Tarek Hagras and Jan Janecek, "A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems," *Parallel Computing*, vol.31, no.7, pp.653–670, 2005. Article(CrossRefLink)
[4] Meikang Qiu, Minyi Guo, Meiqin Liu, Chun Jason Xue, Laurence Tian-ruo Yang, and Edwin Hsing-Mean Sha, "Loop scheduling and bank type assignment for heterogeneous multi-bank memory," *Parallel Distrib.Comput.*, vol.69, no.6, pp.546–558, 2009. Article(CrossRefLink)
[5] Meikang Qiu and Edwin H. M. Sha, "Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol.14, no.2, pp.25–30, April 2009. Article(CrossRefLink)
[6] Stephen T. Heumann, Vikram S. Adve, and Shengjie Wang, "The tasks with effects model for safe concurrency," in *Proc. of the 18th ACM SIGPLAN symposium on Principles and practice of parallel programming* (PPoPP '13). ACM, New York, NY, USA, vol.48, no.8, pp.239-250, August 2013. Article(CrossRefLink)
[7] Ahn Y, Cheng A M K, Baek J, et al., "An auto-scaling mechanism for virtual resources to support mobile, pervasive, real-time healthcare applications in Cloud Computing[J]," *Network, IEEE*, 2013, vol.27, no.5,pp. 62-68, October 2013. Article(CrossRefLink)
[8] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia, "A view of Cloud Computing," *Commun. ACM*, vol.53, no.4, pp.50–58, April 2010. Article(CrossRefLink)
[9] Arutyun I. Avetisyan, Roy H. Campbell, Indranil Gupta, Michael T.Heath, Steven Y. Ko, Gregory R. Ganger, Michael A. Kozuch, David R.O'Hallaron, Marcel Kunze, Thomas T. Kwan, Kevin Lai, Martha Lyons, Dejan S. Milojicic, Hing Yan Lee, Yeng Chai Soh, Ng Kwang Ming,Jing-Yuan Luke, and Han Namgoong, "Open Cirrus: A Global Cloud Computing Testbed," *IEEE Computer*, vol.43, no.3, pp.35–43, April 2010. Article(CrossRefLink)
[10] Benny Rochwerger, David Breitgand, Amir Epstein, David Hadas, Irit Loy, Kenneth Nagin, Johan Tordsson, Carmelo Ragusa, Massimo Villari, Stuart Clayman, Eliezer Levy, Alessandro Maraschini, Philippe Massonet, Henar Muoz, and Giovanni Toffetti, "Reservoir - When One Cloud Is Not Enough," *IEEE Computer*, vol.44, no.3, pp.44–51, March 2011. Article(CrossRefLink)
[11] Marisol Garcia valls (Marisol Garca-valls), Tommaso Cucinotta, and Chenyang Lu, "Challenges in real-time virtualization and predictable Cloud Computing," *Journal of Systems Architecture*, vol.60, no.9, pp.726-740, October 2014. Article(CrossRefLink)
[12] Wei Huang, Jiuxing Liu, Blent Abali, and Dhabaleswar K. Panda, "A case for high performance computing with virtual machines," in *Proc. of International Conference on Supercomputing. ACM*, pp.125–134, 2006. Article(CrossRefLink)
[13] Joshua E. Simons and Jeffrey Buell, "Virtualizing high performance computing," *ACM SIGOPS Operating Systems Review*, vol.44, no.4, pp.136–145, 2010. Article(CrossRefLink)
[14] Hyung Won Choi, Hukeun Kwak, Andrew Sohn, and Kyusik Chung, "Autonomous learning for efficient resource utilization of dynamic VM migration," in *Proc. of the 22nd annual international conference on Supercomputing* (ICS '08). ACM, New York, NY, USA, pp.185-194, 2008. Article(CrossRefLink)
[15] Niels Fallenbeck, Hans-Joachim Picht, Matthew Smith, and Bernd Freisleben, "Xen and the Art of Cluster Scheduling," in *Proc. of Virtualization Technology in Distributed Computing, 2006*, pp.4-4, November 17-17, 2006. Article(CrossRefLink)
[16] R. Jayarani, Rajarathinam Vasanth Ram, Sudha Sadhasivam, and N. Na-gaveni, "Design and Implementation of an Efficient Two-level Scheduler for Cloud Computing Environment," in *Proc. of Advances in Recent Technologies in Communication and Computing*, pp.585–586, May 17-20, 2010. Article(CrossRefLink)
[17] A. Gupta, S. Kohli and S. Jha, "Performance of EDF-BF algorithm under QoS constraint in grid heterogeneous environment," in *Proc. of Information Systems and Computer Networks (ISCON), 2013 International Conference on* Mathura, pp. 170-172, March 9-10, 20113. Article(CrossRefLink)

[18] Juefu Liu and Peng Liu, “The research of load imbalance based on Min-Min in grid,” in *Proc. of International Conference on Computer Design and Applications*, pp.41-44, June 25-27, 2010. Article(CrossRefLink)
[19] OmidReza Kiyarazm, M-Hossein Moeinzadeh, and Sarah Sharifian-R, “A New Method for Scheduling Load Balancing in Multi-processor Systems Based on PSO,” in *Proc. of International Conference on Intelligent Systems, Modelling and Simulation, 2011 Second International Conference on. IEEE*, pp.71-76, January 25-27, 2011. Article(CrossRefLink)
[20] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia, “A view of Cloud Computing,” *Communications of the ACM*, vol.53, no.4, pp.50-58, April, 2010. Article(CrossRefLink)
[21] X. Xu, D. Hu and X. Lu, “Kernel-Based Least Squares Policy Iteration for Reinforcement Learning,” in *Proc. of IEEE Transactions on Neural Networks*, vol. 18, no.4, pp.973–992, July 9, 2007. Article(CrossRefLink)
[22] S. Santra and K. Mali, "A new approach to survey on load balancing in VM in Cloud Computing: Using CloudSim," *Computer, Communication and Control (IC4)*, in *Proc. of 2015 International Conference on*, Indore, 2015, pp. 1-5, September 10-12, 2015. Article(CrossRefLink)

Bin Hu is a teacher in College of Information science and Technology, Bohai University, and is also a Ph.D. student in School of Computer and Communication Engineering, University of Science and Technology Beijing. His research interests include the theory and application of Cloud Computing.

Ning Xie received the M. E., and Ph.D degrees from Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan, in 2009 and 2012 respectively. In 2012, he was appointed as a Research Associate in the same institute. From 2014, he is an assistant professor in the School of Software Engineering, Tongji University. His research interests include the theory and application of machine.

Tingting Zhao obtained her Ph.D. degree from Tokyo Institute of Technology, Japan. She is the Associate professor of Tianjin University of Science & Technology, China. She has published more than 15 papers in leading international journals or conferences of machine learning domain, including Neural Computation, Neural Networks, NIPS2011, ECML/ PKDD 2014 and IJCAI 2015. Her research interests mainly include machine learning, robot control, pattern recognition etc.

Xiaotong Zhang received his Ph.D. degree from University of Science and Technology Beijing in 2000. As national university students were sent to the Lehigh University in Computer Science and Engineering in computer networks. Now, he is an professor in University of Science and Technology Beijing. His research interests include the theory and application of cloud computing and wireless sensor networks.