TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 07/09 pp660-667
Volume 21, Number 6, December 2016

Virtual Machine-Based Task Scheduling Algorithm in a Cloud Computing Environment

Zhifeng Zhong*, Kun Chen, Xiaojun Zhai, and Shuange Zhou

Abstract: Virtualization technology has been widely used to virtualize single server into multiple servers, which not only creates an operating environment for a virtual machine-based cloud computing platform but also potentially improves its efficiency. Currently, most task scheduling-based algorithms used in cloud computing environments are slow to convergence or easily fall into a local optimum. This paper introduces a Greedy Particle Swarm Optimization (G&PSO) based algorithm to solve the task scheduling problem. It uses a greedy algorithm to quickly solve the initial particle value of a particle swarm optimization algorithm derived from a virtual machine-based cloud platform. The archived experimental results show that the algorithm exhibits better performance such as a faster convergence rate, stronger local and global search capabilities, and a more balanced workload on each virtual machine. Therefore, the G&PSO algorithm demonstrates improved virtual machine efficiency and resource utilization compared with the traditional particle swarm optimization algorithm.

Key words: cloud computing; virtual machine; G&PSO algorithm

1 Introduction

Along with the development of grid computing, high-performance storage transmission technology, the WEB2.0, and virtualization technology, cloud computing has become a popular commercial technology and uses virtualization technology to provide users with an infrastructure, a platform, and software services from the data center[1, 2].

The principle of virtualization technology is to virtualize computer hardware to run multiple independent operating systems in the same hardware environment. Consequently, each operating system can run multiple applications simultaneously in independent physical spaces, which significantly improves the efficiency of the cloud computing platform[3]. Server virtualization technology is one of the key technologies in the virtualization technology family. In this technology, a single physical machine can be instantiated into multiple virtual machines, and the remaining computing resources of each physical machine can be mapped and virtualized into a new virtual machine for other users[4–6]. In general, the actual utilization of the physical server is only 7% to 12%. Therefore, running multiple virtual servers on a single server would not only reduce the overall business cost but also greatly improve the utilization of the servers[7, 8]. In fact, the greatest potential of virtualization is to integrate x86 servers into a single private cloud with multiple independent virtual servers to provide greater utilization efficiency of the available resources[9, 10].

As cloud computing environments need to scale to a large number of users and tasks, designing a scheduling algorithm that can efficiently distribute the

• Zhifeng Zhong, Kun Chen, and Shuange Zhou are with College of Computer and Information Engineering, Hubei University, Wuhan 430062, China. E-mail: fzhong@hubu.edu.cn; ckjack2006@163.com; 1224242749@qq.com.
• Xiaojun Zhai is with Department of Electronics, Computing and Mathematics, University of Derby, Derby DE22 1GB, UK. E-mail: x.zhai@derby.ac.uk.
* To whom correspondence should be addressed.
Manuscript received: 2016-07-14; revised: 2016-08-18; accepted: 2016-10-03

tasks and resources becomes a key point for research. Current research uses probabilistic techniques, e.g., greedy algorithm, genetic algorithm, the Particle Swarm Optimization algorithm (PSO), and the Ant Colony (ACO) algorithm. In Ref. [11], a greedy algorithm was used to schedule tasks to improve the overall quality of the cloud computing service. Similarly, genetic algorithm[12] and genetic ACO algorithm[13] have also been used to schedule cloud computing tasks to reduce completion time and cost. In Ref. [14], virtual machine-based particle swarm and Tabu Search (TS) algorithms were introduced to enhance population diversity in order to avoid a particle being prematurely trapped in local optimization and to eventually improve the scheduling performance of the virtual machine tasks in a cloud environment and reduce the task execution time. However, there are still opportunities to improve the existing algorithms. For example, although the genetic algorithm has a fast random global search ability, its implementation is complicated and easily falls into local optimum. Despite its stronger global search ability, the search direction of the PSO algorithm is difficult to control and exhibits different convergence rates during the initial and later stages, which means that the result of the optimization is difficult to predict and control[15]. However, compared with the genetic algorithm, PSO has a fast convergence rate, better optimization performance, and is easy to implement[16]. Conversely, the greedy algorithm has a better local search ability. This algorithm attempts to access the local optimal solution. However, it often just gives an approximate solution. Therefore, the PSO algorithm does not have strong global search capabilities[17]. Although the ACO algorithm has better optimization ability, at the initial stage, it lacks information sources and its convergence rate is slower compared with other algorithms[18].

Existing task-scheduling algorithms aimed at the cloud platform are achieved by large-scale server clusters and virtual machine clusters. However, these algorithms do not consider the requirements of medium- and small-sized enterprises that use only a single server to build their own cloud platform to cope with growing business requirements. Therefore, a highly-efficient virtual machine task-scheduling algorithm is required to improve the overall efficiency and operation cost of such a cloud platform. Based on this problem, this paper introduces a Greedy Particle Swarm Optimization (G&PSO) based task-scheduling algorithm for virtual machine based on a cloud computing platform. The major advantage of the proposed algorithm is that it has strong local and global search abilities, along with a fast convergence rate. The archived experimental results show that the proposed algorithm has a faster convergence rate at earlier stage iterations and stronger local search capability during later periods. This means that it outperforms the traditional PSO algorithm with better global optimization performance and overcomes its shortcoming with greater randomness. Within a cloud environment deployed by a single server, using the proposed algorithm will not only reduce the total task completion time but also will balance the system load and improve the efficiency of task scheduling and resource utilization of the cloud computing platform.

2 Cloud Computing Task Scheduling Problem

The essence of the resolution of the cloud computing task-scheduling problem is to set up a scheduling policy. Based on this, suitable mapping relationship can be established between application tasks and computing resources in order to achieve reasonable distribution and efficient execution of application tasks using the limited computing resources[19]. In this paper, the proposed cloud computing task-scheduling algorithm virtualizes a single server into multiple virtual machines, then assigns T independent tasks to M heterogeneous virtual machines for execution (i.e., one task can not be run on two virtual machines, each virtual machine can only handle one task at one time and each has different properties), thus minimizing the time required to complete all the tasks[20]. In order to simplify the simulation process, this paper will ignore the memory and other resource requirements of the tasks. Moreover, the execution time of each task is only related to the size of the task and the property of the virtual machine. The task set is represented as $\mathrm{TS} = \{t_1, t_2, \ldots, t_n\}$, and the task size is expressed as MI (Million Instructions). The performance of the virtual machines is represented as Millions of Instructions Per Second (MIPS). The expected execution time of task TS_i run on virtual machine VMS_j can be expressed as an ETC matrix[21]:

$$\mathbf{ETC}(i\times j) = \begin{bmatrix} \mathrm{ETC}(11) & \mathrm{ETC}(12) & \cdots & \mathrm{ETC}(1j) \\ \mathrm{ETC}(21) & \mathrm{ETC}(22) & \cdots & \mathrm{ETC}(2j) \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{ETC}(i1) & \mathrm{ETC}(i2) & \cdots & \mathrm{ETC}(ij) \end{bmatrix} \tag{1}$$

where $\mathrm{ETC}(ij) = \mathrm{MI}_{\mathrm{TS}_i}/\mathrm{MIPS}_{\mathrm{VMS}_i}$, $i \in \{1, 2, \ldots, T\}$, $j \in \{1, 2, \ldots, M\}$, T is the number of tasks, M is the number of virtual machines, and the load on the virtual machine VMS_j is the total execution time of tasks, which is expressed as

$$\mathrm{Load}_{\mathrm{VMS}_j} = \sum \mathrm{ETC}(ij) \tag{2}$$

The function of the system load balancing degree is defined as

$$\mathrm{Load}_{\mathrm{level}} = \frac{\min_{1 \leqslant j \leqslant M} \mathrm{Load}_{\mathrm{VMS}_j}}{\max_{1 \leqslant j \leqslant M} \mathrm{Load}_{\mathrm{VMS}_j}} \tag{3}$$

In this function, $\min_{1 \leqslant j \leqslant M} \mathrm{Load}_{\mathrm{VMS}_i}$ is the minimum time for all the virtual machines to complete all the tasks above, and $\max_{1 \leqslant j \leqslant M} \mathrm{Load}_{\mathrm{VMS}_j}$ is the maximum time for all the virtual machines to complete all the tasks above. So the function is the ratio of the minimum load to the maximum load. The following conclusions can be drawn from Formula (3):

(1) $\max_{1 \leqslant j \leqslant M} \mathrm{Load}_{\mathrm{VMS}_i} = 0$ means the tasks do not yet start to schedule.

(2) $\mathrm{Load}_{\mathrm{level}} = 0$ and $\max_{1 \leqslant j \leqslant M} \mathrm{Load}_{\mathrm{VMS}_j} =!0$ mean there are idle virtual machines.

(3) $\mathrm{Load}_{\mathrm{level}} = 1$ means that the maximum load is equal to the minimum load, and that the load balance is the best, i.e., the closer to 1, the better.

3 Design of the Cloud Computing Task-Scheduling Optimization Algorithm

3.1 G&PSO

The PSO algorithm was first proposed by Eberhart and Kennedy in 1995, and its basic concept is based on a study of birds foraging behavior[22]. Thus, PSO algorithm was inspired by the behavioral traits of biological group and subsequently has been used to solve and optimize problems.

Important scheduling goals in a cloud computing environment are the reduction of total completion time[23] and balance of the system load[24]. The proposed algorithm first uses a greedy algorithm to quickly find the initial solution G_{ov} and the expected total completion time G_{ct}, then initializes the global optimal solution gbest of the PSO algorithm by G_{ov} and uses $1/G_{\mathrm{ct}}$ as the updating threshold for the best position of particle swarm.

3.2 Encoding and decoding of particles

The direct encoding mode is also adopted in this paper; each particle's position represents a feasible task allocation scheme and the length of the particle depends on the number of tasks. Assume that T (the number of tasks) is 10 and M (the number of available virtual machines) is 5, then each particle from the set {5, 3, 2, 5, 2, 1, 3, 2, 4, 3} corresponds to a feasible task allocation scheme, thus the particles can be encoded. In this allocation scheme, tasks (1, 4) are allocated to the fifth virtual machine, tasks (3, 5) are allocated to the second virtual machine, tasks (2, 7, 10) are allocated to the third virtual machine and task 6 is allocated to the first virtual machine, thus the particles can be decoded.

3.3 Initialization of particles

S, T, and M denote the size of particle swarm, and the number of tasks, the number of virtual machines, respectively. The location of the i-th particle is thus represented as

$$P_i = \{P_{i1}, P_{i2}, \ldots, P_{in}\},\ 1 \leqslant n \leqslant T, 1 \leqslant i \leqslant S,$$

where P_{ij} represents the i-th task that is assigned to run on the j-th virtual machine and $1 \leqslant P_{ij} \leqslant M$. Speed $V_i = \{V_{i1}, V_{i2}, \ldots, V_{in}\}$ $(1 \leqslant n \leqslant T, 1 \leqslant i \leqslant S)$ and V_{ij} must meet the condition of $1 \leqslant V_{ij} \leqslant M$. The initial position of the particle is a random integer selected from $[1, M]$ and the speed of the particle is a random integer selected from $[-(M-1), (M-1)]$. The best position that the entire group has experienced (gbest) is initialized with G_{ov}.

3.4 Fitness function

A fitness function[25] is used to evaluate the merits of the particle positions. As the total task completion time is the key parameter for task-scheduling in cloud computing, the inverse of the total task completion time is used to represent the fitness function. The fitness function is defined as

$$\mathrm{fitness}(i) = \frac{1}{\mathrm{SFT}_i}, 1 \leqslant i \leqslant S \tag{4}$$

$$\mathrm{SFT} = \max_{1 \leqslant m \leqslant M} \left(\sum_{n=1}^{K} \mathrm{VM}(m, n) \right) \tag{5}$$

In Formula (4), SFT_i represents the time needed to complete the task-scheduling for allocating the task at the i-th particle. In Formula (5), SFT represents the time needed to complete all the tasks; $\mathrm{VM}(m, n)$ represents the time for the n-th task to run on the m-th virtual machine, and K is the number of tasks distributed to this virtual machine. Each iteration selects the particle with a larger fitness value, and one of these values is used as the globally optimal solution, which means that adopting this particle' task allocation scheme results in the shortest completion time.

3.5 Update of particles' velocity and position

In the traditional PSO algorithm, only if the particle's current position has a better fitness value than the best recorded position will the best position be replaced by the current position. The best position that the i-th particle has experienced is denoted as $\text{pbest} = (\text{pbest}_{i1}, \text{pbest}_{i2}, \ldots, \text{pbest}_{in})$. In the whole particle swarm, the best position that all particles have experienced is recorded as $\text{gbest}_i = (\text{gbest}_{i1}, \text{gbest}_{i2}, \ldots, \text{gbest}_{in})$. In this formula, n represents the best location of the particle experience, in the range of the total number of tasks ($1 \leqslant n \leqslant T$). For each iteration, the value of the particle's fitness function can be calculated using Formulas (4) and (5). The value of particle's current fitness function is denoted as $f(p_i(t))$, and during the next iteration, the value is denoted as $f(p_i(t+1))$.

$$\text{pbest}_i(t+1) = \begin{cases} \text{pbest}_i(t), \text{if } f(p_i(t+1) \leqslant f(\text{pbest}_i(t)); \\ p_i(t+1), \text{if } f(p_i(t+1)\ f(\text{pbest}_i(t)) \end{cases} \tag{6}$$

$$f(\max(\text{pbest}(t)) = \text{getMax}(f(\text{pbest}_1(t)), f(\text{pbest}_2(t)), \ldots, f(\text{pbest}_s(t))) \tag{7}$$

$$\text{gbest}(t) = \begin{cases} \max(\text{pbest}(t)), \text{if } f(\max(\text{pbest}(t)) > f(\text{gbest}); \\ \text{gbest}, \qquad \text{else} \end{cases} \tag{8}$$

In this paper's algorithm, during the each iteration, if the particle's current position has a better fitness value than the last position, the position will be updated. In the particle swarm, the particle owning the best fitness value, when its fitness value is better than $1/G_{\text{ct}}$ corresponding to the scheduling scheme G_{ov}, which is calculated by the greedy algorithm, its position will be updated. When the above conditions are met, the particle's velocity and position will be updated.

$$v_i(t+1) = \omega \times v_i(t) + c_1 \times \text{Rand}() \times (\text{pbest}_i(t) - p_i(t)) + c_2 \times \text{Rand}() \times (\text{gbest}(t) - p_i(t)) \tag{9}$$

$$p_i(t+1) = p_i(t) + v_i(t) \tag{10}$$

In the above formulas, t represents the number of iterations; ω is the inertia weight; c_1 and c_2 are learning factors, and generally $c_1 = c_2 = 2$. Rand() is a random value within [0, 1]. During the process of iteration, the position of the particle is limited to a specific range ($1 \leqslant p_i(t) \leqslant M$), at the same time, pbest and gbest are also updated accordingly, and finally gbest is output as the globally optimal solution.

3.6 Process of G&PSO

The specific steps of the proposed G&PSO algorithm are as follows:

Step 1 Initialization of the particle swarm. The position and velocity of the particles are first initialized, and the greedy algorithm is used to quickly obtain the initial solution G_{ov} (i.e., a feasible task allocation solution) and the expected total task completion time G_{ct}; then, the best position gbest that the particles experience with G_{ov} is initialized.

Greedy procedure: The procedure starts from index row 0 of the ETC matrix; it tries to allocate tasks to the virtual machine from the last column of each row in the ETC matrix. If the choice made is better than the others, then the assignment is finished; otherwise, the task is assigned to the virtual machine that makes the current result optimal. Moreover, if there are multiple allocation plans available, then the task is assigned to the virtual machine that has the least tasks, thus achieving simple load balancing[26].

Step 2 Calculate each particle's fitness function value using Formulas (4) and (5).

Step 3 Update the optimal. Update the individual and group optimal based on Formulas (6)–(8):

(1) Compare the value of the particle's fitness function to its individual optimal pbest, if the value of the particle's fitness function is better than pbest, then replace the value of pbest with the current position of the particle.

(2) Compare the particle's fitness function value to its group optimal gbest, if the fitness function value of the particle is better than that of the initial solution calculated by the greedy algorithm, then reset the value of gbest with the particle's current position.

Step 4 Update the speed and position of the particle using Formulas (9) and (10) respectively.

Step 5 Stop conditions. The loop will return to Step 2 until the stop conditions are met.

The flowchart of the proposed G&PSO algorithm is shown in Fig. 1. The flowchart of the Greedy algorithms is shown in Fig. 2.

4 Simulation and Analysis

To validate the feasibility and performance of the G&PSO algorithm in terms of scheduling ability in a cloud, we used the cloud computing simulation platform Cloudsim[27], and extended the DataCenterBroker class of the platform by adding a method to implement

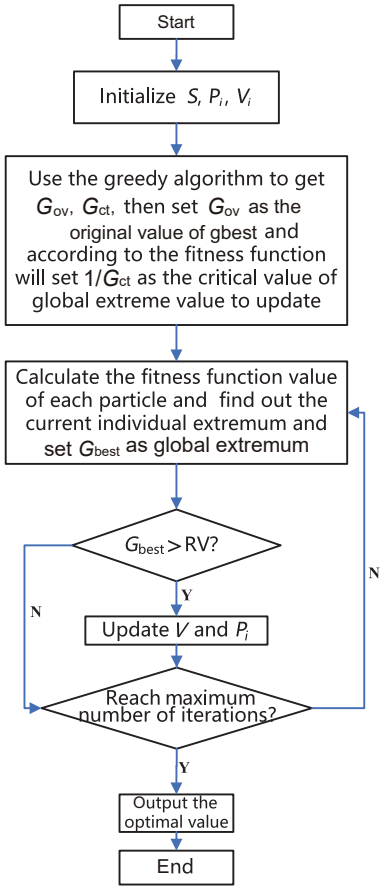


Fig. 1 Proposed G&PSO algorithm.

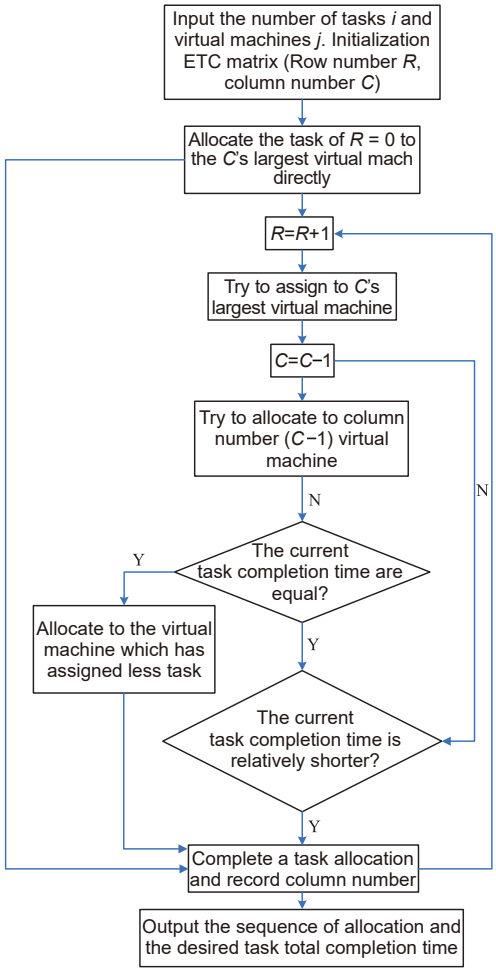


Fig. 2 Proposed greedy algorithm.

greedy particle swarm algorithm class. Eclipse 4.3 IDE was used for the implementation. The computer architecture and operating system of the cloud data center were x86 and Linux, respectively, where each virtual machine had a 1.2 GHz CPU, 2 GB RAM, and 100 GB hard drive, and all virtual machines were set to Xen. The proposed experiments were performed at four different scheduling scales: (1) 5 virtual machines with 50 tasks; (2) 5 virtual machines with 500 tasks; (3) 10 virtual machines with 50 tasks; and (4) 10 virtual machines with 500 tasks. The traditional PSO algorithm and the proposed G&PSO algorithm were applied to each scheduling scale. The main parameters of both algorithms are shown in Table 1.

Table 1 Main parameters of the algorithm.

Name	Value
Population size (s)	100
Number of virtual machines (VM)	5
Performance of virtual machines (MIPS)	{500, 600, 700, 800, 900}
Number of tasks	50, 500
Length range of task (MI)	[500, 2000]
Inertial factor (W)	0.9
Learning factor (c_1)	2
Learning factor (c_2)	2
Maximum number of iterations	200

Figures 3 and 4 show the total completion times for 5 virtual machines with 50 and 500 tasks respectively. As can be seen from Figs. 3 and 4, when using the proposed G&PSO algorithm, the total completion time for the assigned tasks was 10 s less than using the PSO algorithm. In addition, the proposed G&PSO algorithm had less iteration, a faster convergence speed, and less randomness in the processes of optimization for small-or-large scale task scheduling. Figure 3 shows that although the total task completion time of the proposed G&PSO algorithm when scheduling a large-scale task is longer than the PSO algorithm at the initial stage of iteration, the proposed algorithm has a shorter total task completion time. It also has a stronger ability for local searching, which means that it has, to some extent,

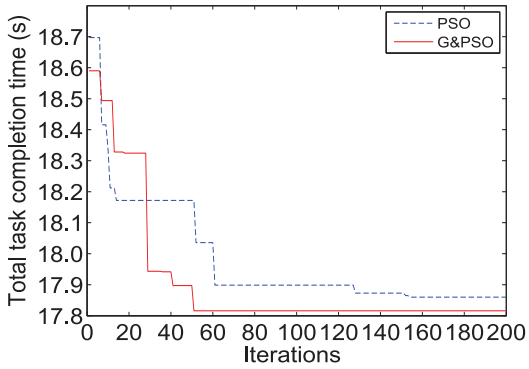


Fig. 3 Completion time vs. iteration with 50 tasks.

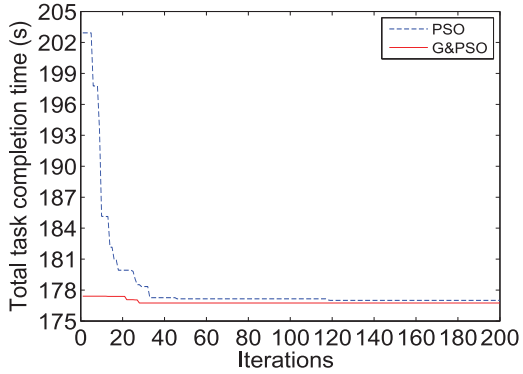


Fig. 4 Completion time vs. iteration with 500 tasks.

overcome the shortcoming of the PSO algorithm with its insufficient local search ability. Compared with the PSO algorithm, when scheduling a large-scale task, the proposed algorithm shows a stronger ability within the optimization process and has a better scheduling effect.

The number of tasks assigned to each virtual machine is shown in Fig. 5 (50 tasks) and Fig. 6 (500 tasks). In terms of utilization of the virtual machine resource, as shown in Figs. 5 and 6, when performing large-or-small scale task scheduling, the number of tasks assigned to each virtual machine is closer to the mean value when using the G&PSO algorithm. This results in improved

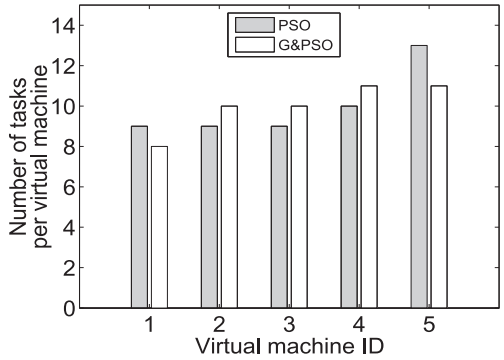


Fig. 5 5 virtual machines with 50 tasks.

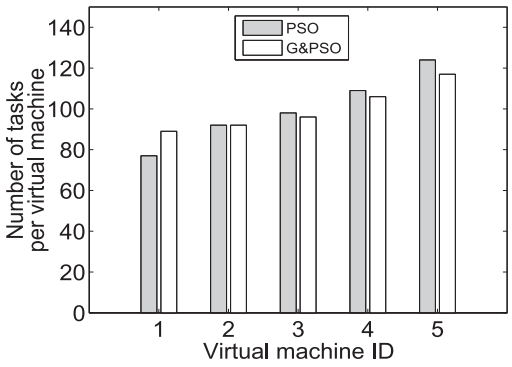


Fig. 6 5 virtual machines with 500 tasks.

utilization of system resources and avoids a workload overload on the virtual machines.

Figure 7 shows that G&PSO algorithm has better load balancing performance compared with the PSO algorithm. Generally, using more virtual machines does not mean obtaining a better result, as configuring each virtual machine often consumes more system resources and eventually leads to a decrease in the overall system performance. Due to the limitation of the physical hardware in the general host and network bandwidth, the number of virtual machines assigned to a single host should be set to no more than 10 to achieve the best system performance.

To further verify the performance of the proposed G&PSO algorithm of load balancing in virtual machines, the number of virtual machines was increased from 5 to 10 and their processing capabilities updated to {500, 600, 700, 800, 900, 1000, 550, 650, 750, 850}. The number of tasks remained unchanged, and those tasks are assigned to each virtual machine shown in Figs. 8 and 9.

In Figs. 8 and 9, the simulation results for both large-scale and small-scale task schedulings are shown. When using the proposed algorithm the number of tasks

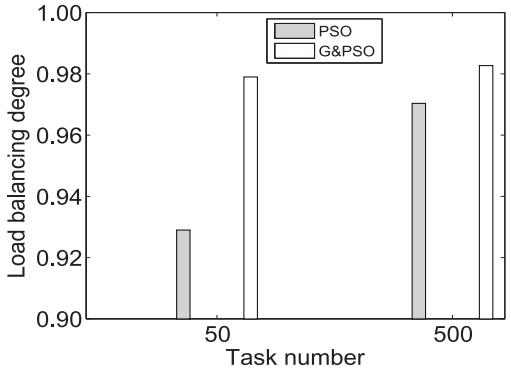


Fig. 7 Systems load balancing degree.

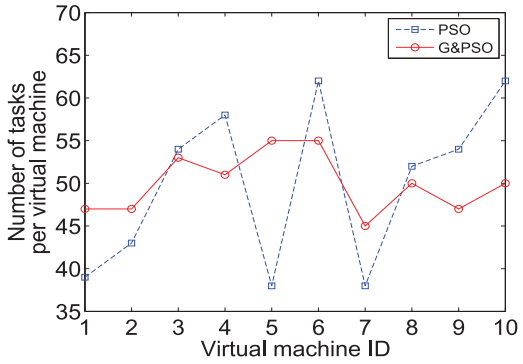


Fig. 8 10 virtual machines with 500 tasks.

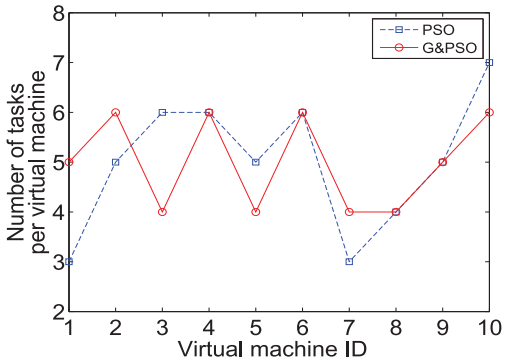


Fig. 9 10 virtual machines with 50 tasks.

assigned to each virtual machine is still closer to the mean value and the system load is still balanced.

In conclusion, the proposed G&PSO algorithm achieves the goals of shorter task completion time and a more balanced virtual machine load; the comprehensive efficiency of the cloud computing platform has therefore been improved.

5 Conclusion

This paper aimed to solve the task-scheduling problems of virtual machines on a cloud platform, and the G&PSO algorithm was proposed to reduce the overall completion time and balance the workload in each virtual machine. Compared with the traditional PSO algorithm, the G&PSO algorithm has a faster convergence rate in the early stage of iteration, a stronger local search capability during the later period of iteration, better global optimization performance, and overcomes the shortcoming of the traditional algorithm with less randomness. On a cloud platform simulated by Cloudsim (Data center disposes one server), the proposed algorithm not only reduces the total task completion time, but also balances the system load and improves the comprehensive efficiency of the entire cloud platform. Although only the size of tasks and processing capacity of the virtual machines were considered when estimating the task completion time, there are more factors to be considered in real applications, such as the effects of bandwidth and data transmission.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., Above the clouds: A Berkeley view of cloud computing, Technical Report No. UCB/EECS-2009-28, University of California at Berkeley, USA, 2009.

[2] A. Matsunaga, M. Tsugawa, and J. Fortes, CloudBLAST: Combining MapReduce and virtualization on distributed resources for bioinformatics applications, in *IEEE Fourth International Conference on Escience*, 2008, pp. 222–229.

[3] R. Uhlig, G. Neiger, D. Rodgers, A. L. Santoni, F. C. M. Martins, A. V. Anderson, S. M. Bennett, A. Kagi, F. H. Leung, and L. Smith, Intel virtualization technology, *Computer*, vol. 38, no. 5, pp. 48–56, 2005.

[4] J. E. Smith and R. Nair, The architecture of virtual machines, *Computer*, vol. 38, no. 5, pp. 32–38, 2005.

[5] L. X. Shi, Utility maximization model of virtual machine scheduling in cloud environment (in Chinese), *Journal of Computers*, vol. 36, no. 2, pp. 252–262, 2013.

[6] J. Daniels, Server virtualization architecture and implementation, *Crossroads*, vol. 16, pp. 8–12, 2009.

[7] H. Liu, A measurement study of server utilization in public clouds, in *IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, 2011, pp. 435–442.

[8] H. González-Vélez and M. Kontagora, Performance evaluation of MapReduce using full virtualisation on a departmental cloud, *International Journal of Applied Mathematics & Computer Science*, vol. 21, no. 2, pp. 275–284, 2011.

[9] Y. Dong and Z. Zhou, X86-based system virtual machine development and application, *Computer Engineering,* vol. 32, no. 13, pp. 71–73, 2006.

[10] B. Sotomayor, S. R. Montero, and I. Foster, Virtual infrastructure management in private and hybrid clouds, *IEEE Internet Computing*, vol. 13, no. 5, pp. 14–22, 2009.

[11] Q. Kang, H. He, and J. Wei. An effective iterated greedy algorithm for reliability-oriented task allocation in distributed computing systems, *Journal of Parallel & Distributed Computing*, vol. 73, no. 8, pp. 1106–1115, 2013.

[12] S. Kaur and A. Verma, An efficient approach to genetic algorithm for task scheduling in cloud computing environment, *International Journal of Information Technology & Computer Science*, vol. 4, no. 10, pp. 74–79, 2012.

[13] Y. Zhang, I. L. Fang, and T. Zhou, Task scheduling algorithm based on genetic ant colony algorithm in cloud computing environment, (in Chinese), *Computer*

Engineering & Applications, vol. 50, no. 6, pp. 51–55, 2014.

[14] D. Hu, J. Hu, and X. Yu. Virtual machine task scheduling algorithm based on pso in cloud computing environment (in Chinese), *Computer Measurement & Control*, vol. 22, no. 4, pp. 1189–1192, 2014.

[15] M. Jiang, Y. P. Luo, and S. Y. Yang, Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm, *Information Processing Letters*, vol. 102, no. 1, pp. 8–16, 2007.

[16] D. Liu, K. C. Tan, C. K. Goh, and W. K. Ho, A multi-objective memetic algorithm based on particle swarm optimization, *IEEE Transactions on Systems Man & Cybernetics—Part B Cybernetics*, vol. 37, no. 1, pp. 42–50, 2007.

[17] C. Y. Liu, C. M. Zou, and P. Wu, A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing, in *Proc.13th Int.Distributed Computing and Applications to Business, Engineering and Science (DCABES), International Symposium on IEEE*, 2014, pp. 68–72.

[18] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, Cloud task scheduling based on load balancing ant colony optimization, in *2011 6th Annual ChinaGrid Conference*, 2011.

[19] P. Wang, Research on task scheduling strategy in cloud computing environment, (in Chinese), *Computer & Modernization*, no. 7, pp. 22–25, 2013.

[20] M. Stillwell, F. Vivien, and H. Casanova, Virtual machine resource allocation for service hosting on heterogeneous distributed platforms, *IEEE International Parallel & Distributed Processing Symposium*, vol. 19, pp. 786–797, 2012.

[21] S. Ali, H. J. Siegel, M. Maheswarand, D. Hensgen, and S. Ali, Representing task and machine heterogeneities for heterogeneous computing systems, *Tamkang Journal of Science & Engineering*, vol. 3, no. 3, pp. 19–25, 2003.

[22] I. C.Trelea, The particle swarm optimization algorithm: Convergence analysis and parameter selection, *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[23] S. Yi, D. Kondo, and A. Andrzejak, Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud, in *IEEE Int. Cloud Computing Conf.*, 2010, pp. 236–243.

[24] N. J. Kansal and I. Chana, Cloud load balancing techniques: A step towards green computing, *International Journal of Computer Science Issues*, vol. 9, no. 1, pp. 238–246, 2012.

[25] X. Li, Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function, *Lecture Notes in Computer Science*, vol. 3102, pp. 117–128, 2004.

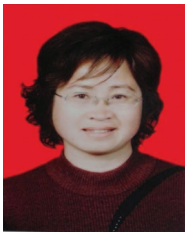
[26] B. Hayes, Cloud computing, *Communications of the ACM*, vol. 51, no. 1, pp. 47–68, 2008.

[27] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Software Practice & Experience*, vol. 41, no. 1, pp. 23–50, 2011.

Zhifeng Zhong received the PhD degree in communication and information system from Wuhan University in 2007. He is currently an associate professor working at the Department of Computer and Communication Engineering, Hubei University, China. His research interests include radar system and signal processing, photovoltaic power generation, and system integration.

Xiaojun Zhai received the PhD degree from the University of Hertfordshire, UK, in 2013. He is currently a lecturer at the College of Engineering and Technology, University of Derby. His research interests mainly include the design and implementation of the digital image and signal processing algorithms, custom computing using FPGAs, embedded systems and hardware/software co-design. He is a member of British Computer Society and Fellow of the Higher Education Academy.

Kun Chen is currently a master student at the Department of Computer and Communication Engineering, Hubei University. He received the BS degree from Hubei University, China, in 2012. His research interests are signal processing and system integration.



Shuange Zhou received the PhD degree in communication and information system from Harbin Engineering University in 2003. She is currently a professor working at the Department of Computer and Communication Engineering, Hubei University, China. Her research interests include distributed computing, database technology, and fault-tolerant.