International Journal of Computer Science and Software Engineering (IJCSSE), Volume 5, Issue 3, March 2016
ISSN (Online): 2409-4285 www.IJCSSE.org Page: 28-35

An Improved Genetic-based Approach to Task Scheduling in Inter-Cloud Environment

Huan Ma[1] and Miao Zhang[2]

[1, 2] School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, 10083, China

[1]*523691298@qq.com*, [2]*zhangmiao_lyy@163.com*

ABSTRACT

With the development of cloud computing, the number of cloud computing service providers has arisen rapidly. The research of task scheduling in cloud computing environment nearly enters a mature stage. But when there is a sharp increase in the amount of user tasks, a single cloud provider cannot meet user's needs. This phenomenon prompted the generation of Inter-cloud, and the task scheduling in which gradually gets everyone's attention. In this paper, we improve the genetic algorithm by adopting Gene Space Balance Strategy (GSBS), which optimizes the generation of initial population. On the basis of improved algorithm, we propose the multi-objective optimization task scheduling method in Inter-cloud. The scheduling goal is to minimize the completion time and cost. We can complete task scheduling according to the different QoS requirements of users. By performing simulation on Cloud-Sim, we demonstrate the effectiveness of improved algorithm. At the same time, we compare the scheduling results of single-cloud and Inter-cloud.

Keywords: *Genetic Algorithm, Task Scheduling, Inter-cloud, Gene Space Balance Strategy.*

1. INTRODUCTION

Since Google first proposed the conception of "cloud computing" in 2006, cloud computing develops rapidly in recent years. There launched a large number of cloud computing centers all over the world, for example, Amazon Elastic Compute Cloud, EC2, Amazon Simple Storage Service, S3 and Google App Engine, etc. Cloud computing centers provide elastic resources to users which saves users' expense. At the same time, by task scheduling and resource sharing, cloud computing improves the utilization of physical servers. Thus achieving a win-win situation between users and cloud computing centers.

Task scheduling is one of the main research problems in cloud computing [1]. In distributed heterogeneous computing environment, tasks best mapped to appropriate machines to perform has been proved to be a NP-hard optimization problem. Heuristic algorithm, such as genetic algorithm (GA), ant colony optimization algorithm (ACO) and particle swarm optimization algorithm (PSO), is a good solution to this kind of problem. Reference [2] utilizes GA to solve the problem of slow convergence speed of ACO algorithm. It meanwhile reduces the average completion time of tasks and improves the utilization of resources. The work in [3] adopts a double-fitness GA, which not only shortens the total completion time and also reduces the average completion time of tasks. At the same time, with the increasing size of the cloud computing center, energy consumption has become a huge problem to be considered for cloud service providers. The energy-conscious task scheduling strategy is put forward in [4], [5]. It observably reduces the energy consumption without significantly affecting the performance. Another important scheduling target is load balancing. On the premise of meeting users' needs, reference [6] and [7] achieve load balancing among virtual machines (VMs), thus improving the resource utilization.

At present, the development of cloud computing center is close to mature. Each cloud service provider can individually provide services to users. But the resource of a single cloud is limited. When the current choosing cloud cannot satisfy the user's need, we need to resort to other cloud providers. This phenomenon leads the generation of Inter-cloud [8], and since IEEE sets up Inter-cloud working group (ICWG) in 2012, researches about Inter-cloud have gradually emerged. The present researches are mainly focus on Inter-cloud architecture [9], interoperability [10], and communication mode [11], [12]. Inter-cloud unites multiple cloud providers together and provides services to users by means of the collaboration between clouds as well as taking full advantage of each cloud's resources. One of the problems we need to solve in Inter-cloud is task scheduling. Considering the profit of cloud service provider and the cost users need to pay, reference [13] proposes a reputation-guided genetic algorithm to solve

the independent task scheduling in Inter-cloud. Reference [14] uses genetic algorithm to schedule impatient tasks in Inter-cloud, which is proved to be effective by observing mapping time and the completion time of tasks.

Based on Inter-cloud architecture and the existing scheduling researches in single cloud and Inter-cloud, we consider the task scheduling across clouds. Due to geographically distribution of each cloud, there exists communication delay. In addition, the resource in one cloud is limited and the resource price is different. In this paper, on the basis of taking account of the effect of distance on completion time of tasks and execution cost, we propose an improved genetic algorithm to schedule independent tasks, which can adjust scheduling target according to different QoS requirement of users.

The rest of this paper is organized as follows. Section 2 describes the task scheduling problem in Inter-cloud environment. The improved genetic algorithm is introduced in section 3. In section 4, we perform simulation on Cloud-Sim and verify the effectiveness of improved algorithm. The paper is concluded in section 5.

2. PROBLEM DESCRIPTION

The services cloud computing provided include storage, computing and networking. In this paper, we mainly focus on scheduling computing tasks. Task scheduling architecture in Inter-cloud is shown in Fig.1. There are a lot of physical resources in each cloud datacenter. They have been virtualized to form a virtual resource pool. In order to realize Inter-cloud task scheduling, cloud providers should communicate in a uniform standard and the real-time resource status can be informed by cloud coordinator to form the unified descripted resource directory. When a user sends a request to perform independent tasks, first we check the resource directory. Then tasks are assigned to VMs according to scheduling algorithm. Next, we allocate the VMs to appropriate datacenter according to user's QoS requirement. We propose the task scheduling algorithm in the case of barrier-free communication between cloud providers and normalized service description.

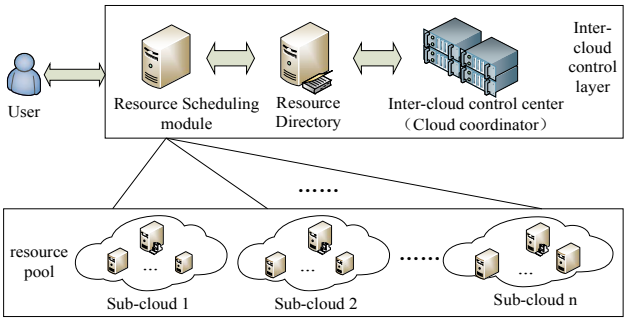


Fig. 1. Inter-cloud task scheduling architecture

Before we propose the scheduling algorithm, we make the following assumptions,

1) The user tasks are independent;
2) The network links from user to different cloud datacenters are stable and no congestion;
3) Tasks can be executed immediately after arriving at datacenter.
4) VMs won't fail, or they can be restored immediately.

In this paper, the number of tasks is denoted by TN. The task set is . For each task , there are three properties, which are described in Table I. VN is used to represent the number of virtual machine types. The virtual machine set is . The properties of each virtual machine, , are described in Table II. For calculation convenience, we assume that one datacenter denotes a sub-cloud. The number of cloud datacenter is DN. Table III records the properties of each datacenter.

Table 1: Task Property

1.1.1.1 Property Name	*1.1.1.2 Description*
task_length	The task length
tXlocation	The x location of task
tYlocation	The y location of task

Table 2: Virtual Machine Property

1.1.1.3 Property Name	*1.1.1.4 Description*
size	VM storage size
ram	VM memory size
bw	VM bandwidth
mips	Processing speed of process element
pesNumber	The number of processing element in VM

Table 3: Datacenter Property

1.1.1.5 Property Name	*1.1.1.6 Description*
costPerSecond	CPU usage cost per second
costPerMem	Memory cost per unit
costPerStorage	Storage cost per unit
costPerBw	Bandwidth cost per unit
dcXlocation	The x location of datacenter
dcYlocation	The y location of datacenter
trans_price	The transmission charge per unit distance
trans_time	The transmission time per unit distance

3. TASK SCHEDULING ALGORITHM

Inter-cloud task scheduling requires us to find the optimal one in a large number of feasible solutions. Several heuristic algorithms have been used to solve this problem, including the commonly used genetic algorithm, ACO algorithm and PSO algorithm. Among them, genetic algorithm has the advantage of global optimization, which is in line with the scheduling model in this paper. We make appropriate improvements on genetic algorithm and propose a task scheduling method with multi-objective optimization.

The flowchart of task scheduling algorithm in this paper is shown in Fig.2. First, we choose an appropriate encoding schema and generate initial population. We calculate the fitness value of each individual and record the best one. Then we judge whether the termination condition is reached. If not, selection, crossover and mutation operation will be iteratively proceeded until the condition reached. Next, we will introduce each step in detail.

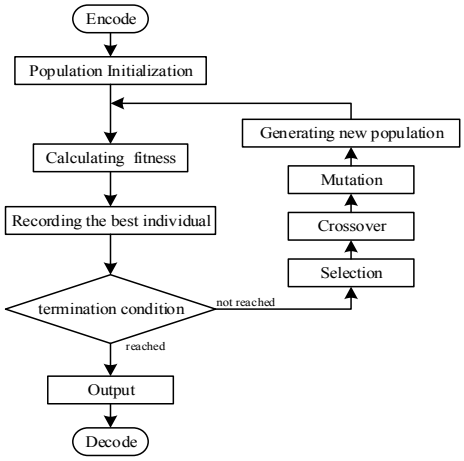


Fig. 2. The flowchart of scheduling algorithm

3.1 Fitness Function

One of the important step in genetic algorithm is calculating fitness value, which is used to evaluate user experience. Individuals with higher fitness value have a greater probability to be inherited to the next generation. In users' perspective, they expect short completion time and low cost to perform tasks. In this paper, we choose both two as the QoS evaluation criteria to conduct multi-objective optimization. The cloud executing tasks may be a little bit further to user. Thus we use transmission cost to quantify the impact of long distance brought to the total cost and we assume that transmission time is proportional to the distance. The total cost user need to undertake includes the transmission cost and the usage charges of the VMs. While the total completion time includes the transmission time and execution time of tasks.

The transmission and execution cost are respectively calculated by formula (1), (2).

$$transPrice_{T_i \to DC_j} = trans\_price_{DC_j} \times \sqrt{(dcXlocation_{DC_j} - tXlocation_{T_i})^2 + (dcYlocation_{DC_j} - tYlocation_{T_i})^2} \quad (T_i \in T, DC_j \in DC, 0 \le i \le TN-1, 0 \le j \le DN-1) \tag{1}$$

$$processPrice_{T_i \to V_k \to DC_j} = (task\_length_{T_i} / (mips_{V_j} \times pesNumber_{V_j})) \times costPerSecond_{DC_k} + size_{T_i} \times costPerStorage_{DC_k} + ram_{V_j} \times costPerMem_{DC_k} + bw_{V_j} \times costPerBw_{DC_k} \quad (T_i \in T, V_k \in V, DC_j \in DC, 0 \le i \le TN-1, 0 \le k \le VN-1, 0 \le j \le DN-1) \tag{2}$$

Then, the total cost is,

$$cost = \sum_{i=0}^{TN-1} (transPrice_{T_i \to DC_j} + processPrice_{T_i \to V_k \to DC_j}) \tag{3}$$

The transmission and execution time of a task are respectively calculated by formula (4), (5),

$$transTime_{T_i \to DC_j} = trans\_time_{DC_j} \times \sqrt{(dcXlocation_{DC_j} - tXlocation_{T_i})^2 + (dcYlocation_{DC_j} - tYlocation_{T_i})^2} \quad (T_i \in T, DC_j \in DC, 0 \le i \le TN-1, 0 \le j \le DN-1) \tag{4}$$

$$processTime_{T_i \to V_k} = task\_length_{T_i} / (mips_{V_k} \times pesNumber_{V_k}) \quad (T_i \in T, V_k \in V, 0 \le i \le TN-1, 0 \le k \le VN-1) \tag{5}$$

The completion time of a set of tasks is,

$$time = \max_{i=0}^{TN-1}(transTime_{T_i \to DC_j} + processTime_{T_i \to V_k}) \quad (6)$$

As previously mentioned, individuals with higher fitness value have a greater probability to be inherited to the next generation. Therefore, in order to make low-cost and short-time, the fitness value should be inversely proportional to both of them. Firstly, we normalize cost and time. The normalization formulas are shown in (7) and (8).

$$normalizedCost = cost / maxCost \quad (7)$$

$$normalizedTime = time / maxTime \quad (8)$$

In which, and are respectively the maximum cost and maximum time of current population. The fitness function in this paper is,

$$fitness = alpha \times \frac{1}{normalizedCost} + (1 - alpha) \times \frac{1}{normalizedTime} \quad (9)$$

In the formula above, alpha is user preference factor. When alpha is close to 0, the set of tasks is impatient, which means they should be executed in time. While alpha is close to 1, user requires the execution fee as little as possible. By adjusting the value of alpha, we can meet the QoS requirements of different users to reach multi-objective optimization purposes.

3.2 Encoding

Encoding means to denote the possible solution in the form of chromosome. In this paper, the chromosome represents scheduling result, the length of which is the number of tasks plus that of virtual machines. It is expressed in formula (10).

$$chromosomeLength = TN + VN \quad (10)$$

The former TN genes represent the VM number that each task assigned to. The remaining VN genes denote the datacenter number that each VM assigned to. Thus, we complete the task scheduling from task to VM and VM to datacenter. For example, the chromosome shown in Fig.3, task T_0 is allocated to VM V_4 and task T_{TN-1} is allocated to VM V_{20}. VM V_0 is assigned to datacenter DC_5 and VM V_{VN-1} is assigned to datacenter DC_0.

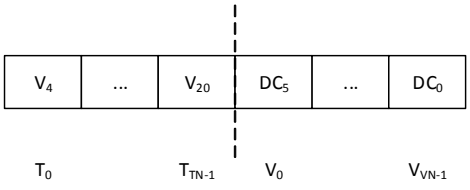


Fig. 3.chromosome

3.3 Initial Population

We set the size of population in advance, denoted as popSize. Population initialization is to produce popSize feasible solutions which are expressed in the form of chromosomes. The initial population is,

$$population = (CH_0, CH_1, ..., CH_{popSize-1}) \quad (11)$$

CH_0 , CH_1 , … , $CH_{popSize-1}$ are respectively represent the individual in population.

In traditional genetic algorithm, the initial population is generated randomly, which is easy and simple to achieve. But the gene is not enough diversity. We adopt GSBS [15] to replace random generation, which improves the gene diversity and ensures high evolution efficiency. The GSBS is described in Table IV.

Table 4: Gene Space Balance Strategy

Space Balance Strategy
Input: the population size, popSize
Output: initial population
1. while the size of chromosomes in population is smaller than popSize
2. generate a chromosome randomly
3. update count matrix (including, VM count matrix: the number of each VM appearing in each gene-position; datacenter count matrix: the number of each datacenter appearing in each gene-position)
4. for each one in count matrix
5. if the value is greater than the average value of the corresponding gene appearing in the gene-position, then
6. mutate this gene to other values which appear the least
7. end
8. end

3.4 Genetic Algorithm Operations

The operations in genetic algorithm include selection, crossover and mutation. Then we will introduce each operation in detail.

Selection operation is to select popSize chromosomes from the current population to form a new one. In this paper, we use roulette-wheel selection policy. It means to perform popSize times selection. Each time we choose one individual from the current population. The probability of an individual to be chosen is,

$$p(CH_i) = \frac{fitness_{CH_i}}{\sum_{i=0}^{popSize-1} fitness_{CH_i}} \quad (12)$$

fitness represents the fitness value of an individual, which is calculated by formula (9).

The roulette-wheel selection policy ensures that individuals with higher fitness value have greater probability to be inherited to the next generation. In

addition, we make the one with highest fitness value, which is referred as elite individual, directly inherit to the next generation. Thereby speeding up the convergence rate of GA.

Crossover operation is the main method to generate new individual. We choose single-point crossover in this paper. It refers to randomly choose several couples of chromosomes according to the crossover probability, p_c. Then the two in each couple cross with each other at the cross-point to produce two new individuals. In order to accelerate the running speed of genetic algorithm, we assume that if the fitness values of descendants after crossover are smaller than parent chromosomes, we do not update the population. The crossover operation is shown in Fig.4.

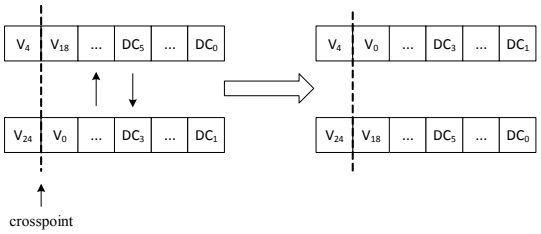


Fig. 4. Crossover operation

4. SIMULATION AND ANALYSIS

In this section, we perform simulations of our algorithm on Cloud-Sim. The run-time environment is Windows 7 with a Core Duo CPU E7400, 2.27Hz, and 5GB RAM. The parameters are set as follows.

Table 5: Task parameter setting

1.1.1.7 Name	*1.1.1.8 Value*
TN	200
task_length	10000 – 50000 mi
tXlocation	Fixed to 0
tYlocation	Fixed to 0

Table 6: VM parameter setting

1.1.1.9 Name	*1.1.1.10 Value*
VN	40
size	8000 – 15000 MB
ram	1024 – 2048 MB
bw	1000 – 2000 bps
mips	400 – 1000 mips
pesNumber	1 – 3

Table 7: Datacenter parameter setting

1.1.1.11 Name	*1.1.1.12 Value*
DN	10
costPerSecond	3 – 5 yuan/s
costPerMem	0.1 – 0.2 yuan/MB
costPerStorage	0.002 – 0.003 yuan/MB
costPerBw	0.1 – 0.2 yuan/bps
dcXlocation	-500 – 500 m
dcYlocation	-500 – 500 m
trans_price	0.1 – 0.2 yuan/m
trans_time	0.01 – 0.03 yuan/s

Table 8: Genetic algorithm parameters

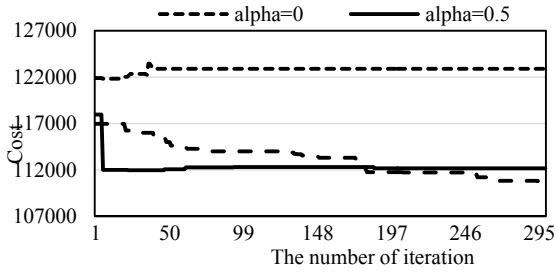
1.1.1.13 Name	*1.1.1.14 Value*
MaxGen	300
popSize	40
chromosomeLength	240
pc	0.8
pm	0.01

In this paper, we can adjust scheduling result according to user's QoS requirement. The preference factor is alpha. When the value of alpha changes from 0 to 1, the requirement of user changes from minimum completion time to minimum cost. The multi-objective scheduling result of improved algorithm is shown in Fig.5. In which, (a) is the cost curve when alpha is different value. While (b) is the completion time curve. We can see from Fig.5, when the value of alpha is 0, the algorithm optimizes the completion time and makes it as short as possible. The completion time significantly decreases versus to the iteration number. On the contrary, the total cost decreases by optimization when the value of alpha is 1. Thus, we achieve the purpose of multi-objective optimization.

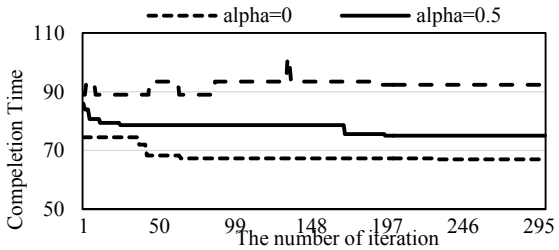
Traditional genetic algorithm randomly generate initial population, which may lead to limited gene diversity. We adopt GSBS instead. GA_Random and GA_GSBS respectively represent the improved algorithm and tradition GA. The comparison of two algorithms is shown in Fig.6. We make the two algorithms respectively execute 10 times and denote the scheduling result. (a) is the cost changing curve when alpha equals 1. And (b) is the completion time changing curve when alpha is 0. We can learn that the scheduling result of our improved algorithm is better than traditional GA. In addition, in order to compare the convergence speed of two algorithms, we assume that if the result has no significant changes in 50 consecutive generations, the algorithm ends. (c) and (d) are the comparison of iteration number respectively when alpha equals 1 and 0. We can think that the convergence speed of improved algorithm is faster.

We have already mentioned that for calculation convenience, a cloud service provider has only one datacenter. In this paper, there are 10 cloud providers in

total. Fig.7 shows the comparison between Inter-cloud scheduling and single-cloud scheduling. In which, (a) and (b) are the scheduling result when task number is 200. We can see that the Inter-cloud scheduling result is neither the best nor the worst. (c) and (d) show the result of Inter-cloud scheduling and single-cloud scheduling under different task number. (c) displays the cost value of tasks executing in different cloud service providers when task number is 200, 500 and 1000. While (b) is the completion time curve. We can conclude that, the Inter-cloud scheduling is not always the best strategy compared with sing-cloud scheduling. According to the figures, the execution cost of Inter-cloud scheduling is higher than single-cloud when task number increases. But the completion time is relatively shorter. The reason may be that the experiment in this paper is conducted in environment without pressure. All the datacenter has no other tasks to execute. The transmission fee of single-cloud scheduling is cheaper than that of Inter-cloud scheduling. Thus lead to total cost cheaper. In real world, a cloud service provider must conduct several user's tasks at the same time. A user may have to wait for some time before tasks to be executed. In this case, single-cloud may be not so efficient. As a result, in practice, users can choose single-cloud or Inter-cloud to perform tasks according to load condition of different cloud and link status.

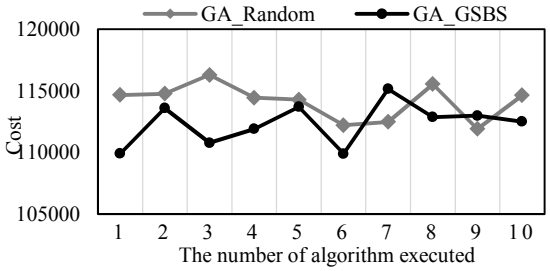


(a) The cost curve of different alpha values

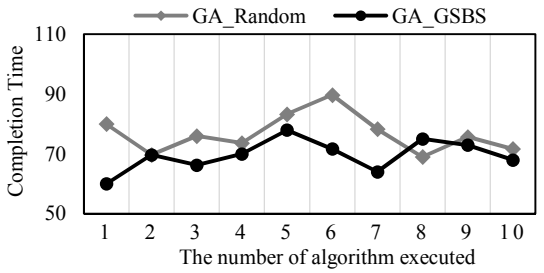


(b) The time curve of different alpha values

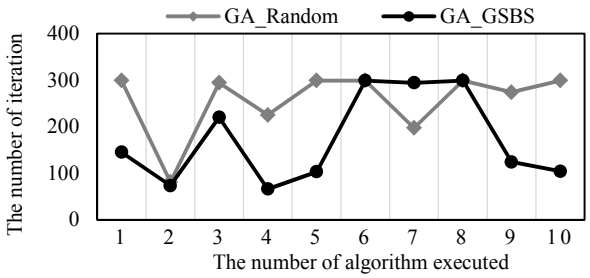
Fig. 5. Multi-objective Optimization Simulation



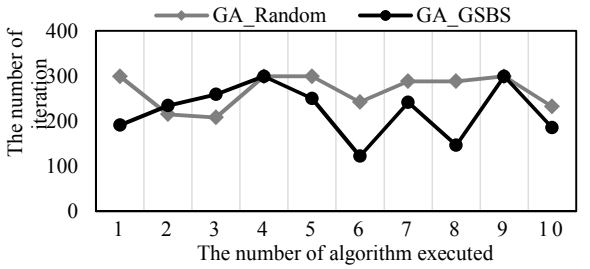
(a) alpha=1, the comparison of cost optimization result



(b) alpha=0, the comparison of completion time optimization result

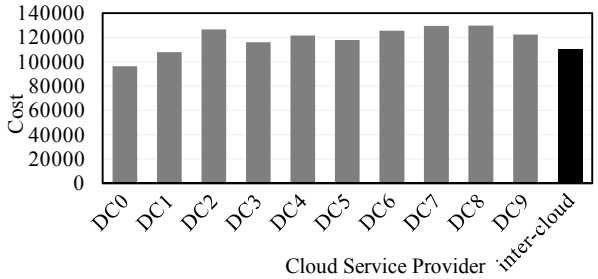


(c) alpha=1, the iteration number comparison



(d) alpha=0, the iteration number comparison

Fig. 6. Algorithm comparison



(a) alpha=1

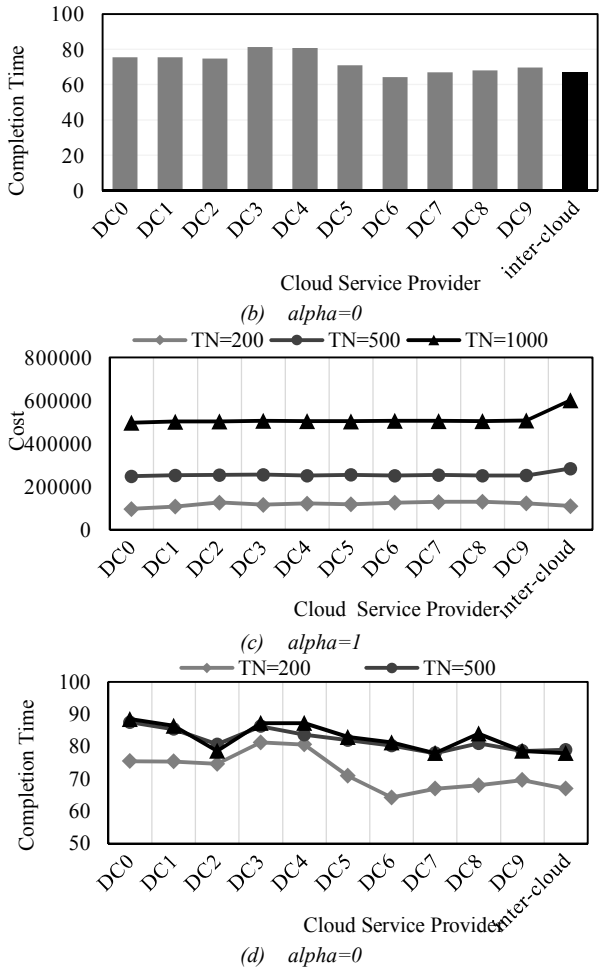


Fig. 7. The comparison of Inter-cloud scheduling and single-cloud scheduling

5. CONCLUSION

In this paper, we firstly describe the Inter-cloud task scheduling problem and make assumptions on the scheduling environment. Then we put forward our scheduling method, which is based on genetic algorithm with gene space balance strategy. The scheduling target is total cost and completion time of tasks. We set user preference factor to the two targets. According to user's different QoS requirement, we can perform multi-objective optimization in Inter-cloud. At last, we simulate the algorithm on Cloud-Sim. The result signifies that the performance of improved algorithm is better than that of traditional GA. In addition, we compare the result of Inter-cloud scheduling and single-cloud scheduling. Inter-cloud scheduling is not always the best. Users can choose the optimal way to conduct tasks. In future work, we will further study the energy problem and consider the dynamic task scheduling in Inter-cloud.

REFERENCES

[1] Q. Chen, Q Deng. Cloud computing and its key techniques. Journal of Computer Applications, vol29, no.9, 2009, pp.2565.

[2] Y.G. Wang, R.L. Han. Study on cloud computing task schedule strategy based on MACO algorithm. Computer Measurement & Control, vol.19, no.5, 2011, pp.1203-1204.

[3] J.F. Li, J. Peng. Task scheduling algorithm based on improved genetic algorithm in cloud computing environment. Jisuanji Yingyong/ Journal of Computer Applications, vol.31, no.1, 2011, pp.184-186.

[4] S.K. Garg, C.S. Yeo, A. A, et al. Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. Journal of Parallel and Distributed Computing, vol.71, no.6, 2011, pp.732-749.

[5] Y.C. Lee, A.Y. Zomaya. Energy efficient utilization of resources in cloud computing systems[J]. The Journal of Supercomputing, vol.60, no.2, 2012, pp.268-280.

[6] Y Fang, F Wang, and J Ge. A task scheduling algorithm based on load balancing in cloud computing[M]//Web Information Systems and Mining. Springer Berlin Heidelberg, 2010, pp.271-277.

[7] P.V. Krishna. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing, vol.13, no.5, 2013, pp.2292-2303.

[8] D. Bernstein. Intercloud – The future cloud of clouds. Journal of the Institute of Telecommunications Professionals, vol.7, no.3, 2013, pp.16-20.

[9] Y. Demchenko, C. Ngo, and C. Laat, et al. Intercloud architecture framework for heterogeneous cloud based infrastructure services provisioning on-demand//Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on. IEEE, 2013, pp.777-784.

[10] Y. Demchenko, C. Ngo, C. Laat, et al. Federated Access Control in Heterogeneous Intercloud Environment: Basic Models and Architecture Patterns//Cloud Engineering (IC2E), 2014 IEEE International Conference on. IEEE, 2014, pp.439-445.

[11] D. Bernstein, D. Vij. Intercloud directory and exchange protocol detail using XMPP and RDF//Services (SERVICES-1), 2010 6th World Congress on. IEEE, 2010. pp.431-438.

[12] J. Lloret, M. Garcia, J. Tomas, et al. Architecture and protocol for intercloud communication. Information Sciences, vol.258, 2014 pp.434-451.

[13] F. Pop, V. Cristea, N. Bessis, et al. Reputation guided genetic scheduling algorithm for independent tasks in inter-clouds environments//Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on. IEEE, 2013, pp.772-776.

[14] N.A. Mehdi, A. Mamat, H. Ibrahim, et al. Impatient task mapping in elastic cloud using genetic algorithm. Journal of Computer Science, vol.7, no.6, 2011, pp.877.

[15] J.J Hu, C.J. Tang, L. Duan, et al. The strategy for diversifying initial population of gene expression

programming. CHINESE JOURNAL OF COMPUTERS-CHINESE EDITION-, vol.30, no.2, 2007, pp.305.