International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.8 (2016), pp.145-152
http://dx.doi.org/10.14257/ijmue.2016.11.8.16

Research on Cloud Task Scheduling Algorithm of Particle Swarm Optimization Algorithm Based on Fission Mechanism

Youwei Shao

School of Applied Electronics, Chongqing College of Electronic Engineering, Chongqing, China
E-mail:

Abstract

Task scheduling and resource scheduling are the core issue in cloud computing. Pointing at the premature problem in the scheduling algorithm of particle swarm, we propose a scheduling algorithm of cloud task particle swarm based on "fission" mechanism in this paper. The particle in traditional particle swarm algorithm gets "fission" by the new algorithm in appropriate place, to get more kinds of the particles, contributing to the particle swarm diversity, avoiding premature convergence of the swarm. As the experimental result shows that, the algorithm in this paper has faster scheduling efficiency than the FIFO and the PSO, also solves the premature problem in PSO.

***Keywords**: cloud computing, particle swarm algorithm, fission, premature*

1. Introduction

The appearance of cloud computing brought a tremendous change in IT industry development ,users no longer have to spend much on expensive hardware, neither on the problems as storage capacity, software upgrades, software maintenance *etc.*, the cloud computing SP will setup servers in cloud to offer all the service the users need [1]. As cloud computing has several certain commercial properties, and needs to meet the demands of different users, so the task scheduling issue in cloud computing environment is a quite complicated but important problem. Currently, most cloud environment are based on the Hadoop architecture, and its scheduling algorithm includes the FIFO (First Input First Output),computing capacity scheduling algorithm and fair scheduling algorithm[2-4],which can hardly meet the demands on the cloud computing task scheduling issue. Therefore, academic circles have carried on the optimization and improvement, and achieved certain results, especially to apply the bionic algorithm into the Hadoop-based cloud platform, which makes good sense. Goyal optimized the task scheduling issue in cloud computing with genetic algorithm, reducing the scheduling time effectively [5]. Gsior improved the ant colony optimization and applied it into the cloud environment, which also has good results in increasing the task scheduling efficiency [6].Gupta applied particle swarm optimization (PSO) into the task scheduling issue in cloud environment, compared with the above two methods, the PSO alleviates the jobTracker load because of less steps [7], but causes "premature" and local optimum in the task allocation process, which influences the task scheduling effect seriously, making it an urgent problem [8].

The purpose of this paper is to improve the cloud task scheduling problem in PSO, to achieve better effect.

2. Scheduling Algorithm Based on "Fission" Mechanism

2.1. General Idea of Algorithm Design

There are two major deficiencies in traditional PSO:

One is to cause "premature " in the optimization, which is because of insufficient velocity update capability of each particle in late particle swarm optimization, making particles gather tightly in some place and unable to search globally more detailed and in greater extent. In terms of the swarm diversity, the diversity is insufficient at this time, there is little difference between the particles, unable to promote the development of the particle swarm.

The others is that the particle iterative update is in the synchronous mode, all the particles need to update synchronously in each iteration to get information of other particles, the next iteration never executes until the previous one finished, which means iteration of all the particles must synchronously calculate the exchanging information between particles(self best location and global best location),which will cause particles losing their independence largely, information of the best particle can not be fully shared, the convergence is slow.

Therefore, pointing at the above two shortages in PSO, we introduce "fission" into PSO in this paper. "Fission" can maintain the swarm diversity effectively, avoiding premature convergence in the algorithm.

2.2. Traditional PSO

Suppose there are m particles in the swarm, the dimension of the searching space is D, then the velocity and location information of the particle i at time t are as formula (1) and (2) show:

$$v_i^t = \{v_{i1}^t, v_{i2}^t, \cdots v_{id}^t\} \tag{1}$$

$$c_i^t = \{c_{i1}^t, c_{i2}^t, \cdots c_{id}^t\} \tag{2}$$

In which, v_i^t indicates velocity of the i-th particle at time t, c_i^t indicates location information of the i-th particle at time t, they both have component of d.

According to the value of fitness function, the self best location and global best location of particle i at time t are judged as formula (3) and (4) show:

$$\bar{c}_i^t = \{\bar{c}_{i1}^t, \bar{c}_{i2}^t, \cdots \bar{c}_{id}^t\} \tag{3}$$

$$\bar{g}_i^t = \{\bar{g}_{i1}^t, \bar{g}_{i2}^t, \cdots \bar{g}_{id}^t\} \tag{4}$$

The velocity updating formula of particle i at time t+1 is as follow:

$$v_{id}^{t+1} = \pi v_{id}^t + \kappa_1 D_1(\bar{c}_{id}^t - c_{id}^t) + \kappa_2 D_2(\bar{g}_{id}^t - c_{id}^t) \tag{5}$$

The location updating formula of particle i at time t+1 is as follow:

$$c_{id}^{t+1} = c_{id}^t + v_{id}^{t+1} \tag{6}$$

In the formulas, κ_1, κ_2 indicate learning factors, D_1, D_2 are random numbers between (0,1). π is inertia weight, which plays an important role in global and local searching capacity of the coordinated particle, a larger π helps the particle in global searching,and a smaller π helps in local searching.

2.3. Fission

In traditional PSO, it uses formula (5) and (6) to iterative update the velocity and location information of the particle, making it approach the optimal location gradually, to get the optimal solution finally. If $\kappa_1 = 0$, as the particle only considers about "global ", the convergence rate may be fast but easily trapped into local optimum, causing the "premature" problem. If $\kappa_2 = 0$, as the particle only considers about the self "inheriting", there will be no information communication between particles, the information sharing is zero, which causes little chance to obtain solutions, also losing the meaning of PSO.

In traditional PSO, the particle considers about both the current self best location and global best location, however, when the distance between the self best location and global best location of a particle is too long, and the difference of their fitness function is quite little, it proves that the current self best location and global best location are both excellent solutions, the velocity updating formula of standard algorithm will lead to the losing of the excellent solution. In order to avoid getting trapped into local optimum and maintain the swarm diversity, the particle can be fission to two parts at this time, one part moves towards the direction of the global best location, and the others is towards the self best location, which can maintain the excellent solutions in the searching effectively, increasing the swarm diversity, so to solve the "premature" problem fundamentally. The two particles in "fission" uses the following new velocity updating formula:

$$v_{id}^{t+1} = \pi v_{id}^{t} + \kappa_1 D_1 (\bar{c}_{id}^{t} - c_{id}^{t}) \tag{7}$$

$$v_{id}^{t+1} = \pi v_{id}^{t} + \kappa_2 D_2 (\bar{g}_{id}^{t} - c_{id}^{t}) \tag{8}$$

Particle using formula (7) to update velocity will move towards the direction of the self best location, and particle using formula (8) will move towards global best location.

The processing steps of the "fission"-based PSO are as follows:

Step 1, initialize the velocity and location of each particle. If the location of the particle at time t is c_i^t, then the self-best location is $\bar{c}_i^t$.The global best location $\bar{g}_i^t$ can be obtained with fitness function.

Step 2, judge the relationship between the difference of the distance d_1-between $\bar{c}_i^t$ and $\bar{g}_i^t$-and the fitness function and the threshold value.

When the distance is too long and the difference of the fitness function is small, it proves that there is also an excellent solution far away from the current global best location. In order to avoid being trapped into local optimum and maintain the swarm diversity, the particle can be "fission" into two parts, one is moving towards the direction of the global best location, and the other one is towards the self best location.

η_1 is the threshold value limiting the distance d_1 between $\bar{c}_i^t$ and $\bar{g}_i^t$, η_2 is the threshold value limiting the difference between the fitness functions, we choose η_2 to be the fitness function tolerant error.

When $d_1 > \eta_1$ and the difference is less than η_2, the particle is "fission" into two parts. They will respectively move towards the global best location and self best location with the different velocity updating formula (7) and (8).In other situations, we choose the updating formula (5) in standard PSO.

Step 3, update the location of the particle with formula (6).

Step 4, judge whether fitting the end condition or not. If the difference of the fitness functions is less than the tolerant error or reaching the maximum iterative times, the iteration will be terminated and output the best solution. Otherwise, repeating step 2 and step 3.

2.4. Building the Cloud Task Scheduling Algorithm

2.4.1. Set Up the Objective Function and Fitness Function

In this paper, we set up a objective function besides the fitness function, which is used to guide the moving track of the subtask, and be the standards to judge the self best location and the global best location of the subtask, the larger the value of the objective function is, the better the location will be. The fitness function is used to control the loading balancing situation of the cloud resource pools, as the standards to judge whether the whole task scheduling process terminates or not, the smaller the value is, the more balancing the loading between resource nodes will be.

Finally, the objective function and fitness function are as formula (9) and (10) show:

$$F_{m,i} = \varphi \times nf_{i,map} + \vartheta \times \min_{1 \le q \le w-1} (s1_{i,q}) \tag{9}$$

$$F_{r.j} = \varphi \times nf_{i,reduce} + \vartheta \times \min_{1 \le q \le w-1} (s2_{j,q}) \tag{10}$$

In which, φ and ϑ indicate weighting factor, $f_{i,map}$ is the number of the free Map task slot on the Slave resource node the nearest to the subtask Map, $f_{i,reduce}$ is the number of the free Reduce task slot on the Slave resource node the nearest to the subtask Reduce, $s1_{i,q}$ is the distance between the i-th subtask Map and the q-th Slaves resource node, $s2_{j,q}$ is the distance between the j-th subtask Reduce and the q-th Slaves resource node.

2.4.2. Designing the Distance-Calculating Formula

As the location information of the subtask iterative updates in the space range, the location information at some moment may be not the same as the location information of the Slaves resource node. Therefore, in this paper, we estimate the merit of current location of the subtask according to the distance information from the location to the nearest Slaves resource node recorded in the table. The distance can be obtained with the location information of the i-th subtask Map and the q-th Slaves resource node, calculating with the following formula (11):

$$s1_{i,q} = \sqrt{\sum_{d=1}^{D} (c_{tm_i,d} - locS_{q,d})^2} \tag{11}$$

Similarly, calculating the distance with the location information of the j-th subtask Reduce and the q-th Slaves resource node, we can get formula (12):

$$s2_{i,q} = \sqrt{\sum_{d=1}^{D} (c_{tr_{ji},d} - locS_{q,d})^2} \tag{12}$$

3. Experiment and Analysis

In order to test the effectiveness of the cloud task particle swarm scheduling algorithm based on “fission” mechanism proposed in this paper, we carried out the following researches. The platform is PC, with Intel Celer 2.50Hz Dual-core CPU, 4G RAM, Windows 7 OS, the cloud computing simulation environment is the CloudSim Platform.

In experiment, we build a cloud with 10 resource nodes, the resource allocations are shown in Table 1.

Table 1. Resource Allocations of the Cloud Nodes in Experiment

	CPU	Memory	Bandwidth
Node 1	Dual-core 2.0GHz	2.0GB	100M/s
Node 2	Dual-core 2.0GHz	1.0GB	100M/s
Node 3	Dual-core 1.0GHz	2.0GB	100M/s
Node 4	Dual-core 1.0GHz	1.0GB	100M/s
Node 5	Single-core 2.0GHz	2.0GB	100M/s
Node 6	Single-core 2.0GHz	1.0GB	100M/s
Node 7	Single-core 1.0GHz	2.0GB	100M/s
Node 8	Single-core 1.0GHz	2.0GB	100M/s
Node 9	Single-core 1.0GHz	1.0GB	100M/s
Node 10	Single-core 1.0GHz	1.0GB	100M/s

In experiment, we set up 4 scheduling task, the resource requirements are shown in Table 2.

Table 2. Task Resource Requirements in Experiment

	CPU	Memory	Bandwidth
Task 1	2.0GHz	0.4GB	20M/s
Task 2	2.0GHz	0.2GB	20M/s
Task 3	1.0GHz	0.2GB	20M/s
Task 4	1.0GHz	0.1GB	20M/s

In experiment, we choose the FIFO and traditional PSO as the comparison algorithm of the algorithm in this paper. The major parameter configuration of the improved PSO in this paper is set up as follows: $\pi = 0.8$, $\kappa_1 = 1.1$, $\kappa_2 = 1.1$, $D_1 = 0.6$, $D_2 = 0.6$, $\varphi = 0.25$, $\vartheta = 0.75$.

Firstly, comparing the difference of scheduling time between the algorithm in this paper and FIFO and traditional PSO, according to the tasks in Table 2, respectively execute scheduling task 1, simultaneously schedule task 1, task 2, simultaneously schedule task 1, task 2, task 3, simultaneously schedule task 1, task 2, task 3, task 4, the difference between the three algorithms is shown in Figure 1.

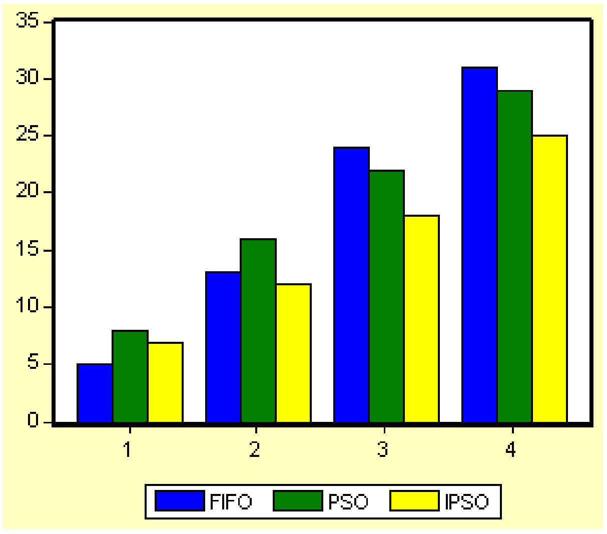

Figure 1. Comparison of Scheduling Time in Three Algorithms

In Figure 1, it can be seen that as the number of the scheduling tasks increases, the algorithm in this paper (Improved PSO) shows its advantage gradually, it costs less and less time than the other two algorithm.

Next, we need to focus on testing after the improvement in this paper, whether the IPSO avoids the premature problem in PSO or not. If premature appears, the iteration of the scheduling algorithm will terminate untimely. The changing of fitness value with the iteration times in IPSO and PSO is shown in Figure 2.

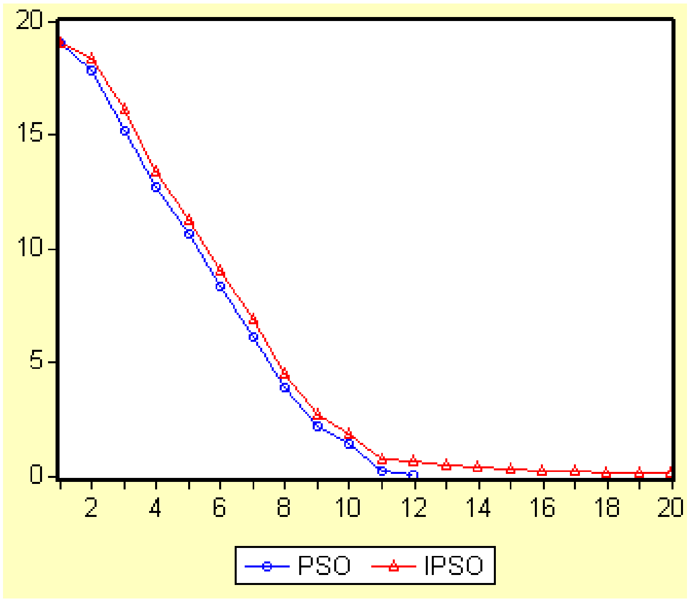

Figure 2. Comparison of Premature in PSO and IPSO

In Figure 2, Y-axis indicates the specific value of fitness function, X-axis indicates the iteration times. It can be seen in Figure 2, PSO terminates in 12 times, which means premature appearing. IPSO improved by the “fission” in this paper iterates 20 times, avoiding the premature problem successfully.

4. Conclusion

Particle swarm optimization is widely used in the cloud computing task scheduling, but the iteration is easy to be premature convergence. Pointing at this problem, we introduce “fission” in this paper. In traditional PSO framework, we “fission” the particle in appropriate place, promoting the swarm diversity, so to avoid the premature in iterative algorithm effectively. As the experimental result shows that, the optimization in this paper not only avoids the premature problem successfully, but has certain advantages in scheduling time as well.

References

[1] K. A. Saranu and J. Suresh, “Intensified scheduling algorithm for virtual machine tasks in cloud computing”, Advances in Intelligent Systems and Computing, vol. 3, no. 25, **(2015)**, pp. 283-290.
[2] B. Saurabh, S. Santwana and D. Madhabananda, “A multi-objective cat swarm optimization algorithm for workflow scheduling in cloud computing environment”, Advances in Intelligent Systems and Computing, vol. 1, no. 3, **(2015)**, pp. 73-84.
[3] A. C. Bartlett, A. A. Andales, M. Arabi and T. A. Bauder, “A smartphone app to extend use of a cloud-based irrigation scheduling tool”, Computers and Electronics in Agriculture, vol. 111, no. 3, **(2015)**, pp. 127-130.
[4] C. Divya and K. Bijendra, “An analysis of the load scheduling algorithms in the cloud computing environment: A survey”, 2014 9th International Conference on Industrial and Information Systems (ICIIS), Gwalior, India, **(2014)**, pp. 15-17.

[5] G. Sudhir, B. Seerna and S. Bhupinder, "Experimental comparison of three scheduling algorithms for energy efficiency in cloud computing", 2014 IEEE International Conference on Cloud Computing in Emerging Markets, Bangalore,, India, **(2014)**, pp. 15-17

[6] G. Jakub and S. Franciszek, "A decentralized multi-agent approach to job scheduling in cloud environment", Advances in Intelligent Systems and Computing, vol. 32, no. 2, **(2015)**, pp. 403-414.

[7] G. Punit and G. S. Prakash, "Load and fault aware Honey Bee scheduling algorithm for cloud infrastructure", Advances in Intelligent Systems and Computing, vol. 328, no. 8, **(2015)**, pp. 135-143.

[8] B. K. Reddy and F. Paul, "Load scheduling in a cloud based massive video-storage environment", 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Lomânia, **(2014)**, pp. 22-25

Author

Youwei Shao, Associate Professor, Chongqing College of Electronic Engineering. Born in 1979, Mr. Shao graduated and got master degree from Chongqing University, and his main research interests are computational intelligence and Cloud Computing.