International Journal of Grid Distribution Computing
Vol. 8, No. 3, (2015), pp.111-118
http://dx.doi.org/10.14257/ijgdc.2015.8.3.11

Virtual Resource Allocation based on Improved Particle Swarm Optimization in Cloud Computing Environment

Youwei Shao

School of Applied Electronics, Chongqing College of Electronic Engineering, Chongqing, China
cqshaoyouwei@126.com

Abstract

A major challenge facing cloud computing is virtual resource allocation with dynamic characteristics. Evaluation of a resource allocation strategy using a single aspect can no longer meet the real world demands. We resolve this issue from the perspectives of users and resource providers using a particle swarm algorithm for resource allocation. With this algorithm, we establish an allocation model using the shortest task completion time and the lowest cost as the constraints. The fast convergence rate of the particle swarm algorithm is then used to find the optimal solution for resource allocation. The velocity weight of each particle is self-adaptively adjusted based on the fitness value of each particle, resulting in an improvement in the global optimization and convergence capabilities. Finally, a simulation with the CloudSim platform shows that this algorithm can take into account the completion time and cost, which ensures the minimum cost in the shortest possible time to complete the task to improve resource utilization.

Keywords: *Cloud computing, Resource allocation, Particle swarm algorithm, CloudSim platform*

1. Introduction

As a business computing model, cloud computing is an extension of various technologies including distributed processing, parallel processing and grid computing. Thus, cloud computing represents a new stage in the development of parallel computing technology [1]. Cloud computing faces many problems, including resource allocation, that have yet to be solved. In cloud computing, the efficiency of resource allocation directly affects the performance of the entire cloud computing environment [2]. Because task scheduling in a cloud computing environment is an NP-complete problem, the development of a heuristic intelligent algorithm is an important direction in this field. In a previous study [3], a modified particle swarm optimisation (MPSO) algorithm was applied for task scheduling in a cloud computing environment with the introduction of dynamic multi-group collaboration and a reverse of the flight of a mutation particle to coordinate the global search and local search, resulting in improved resource use. Another study proposed a resource allocation model based on ant colony optimisation that introduced the concept of entropy into the model to measure the uncertainty of the cloud resource [4]. In addition, a traditional genetic algorithm has been integrated into the task scheduling model in the cloud computing environment to improve the quality of service and the fitness function; however, it is common to encounter issues such as local optimisation [5]. In the present study, we resolve the disadvantages of the particle swarm optimisation (PSO) algorithm, describing an improved particle swarm optimisation (IPSO) algorithm capable of identifying an optimal solution for the cloud resource scheduling problem using the fast convergence rate of the particle swarm algorithm. The velocity weight of each particle self-adaptively adjusts based on the fitness value of each

ISSN: 2005-4262 IJGDC

particle, resulting in improved global optimisation and convergence. Finally, the validity of the IPSO algorithm is verified through simulations.

2. The Cloud Computing Resource Scheduling Model

2.1. The DAG Scheduling Model

$G=(V,E)$ is a set of directed acyclic graphs (DAG), in which V is a collection of computing tasks v and E is a set of edges e representing the relationship of the precedence constraint between the tasks. To maintain generality, it is assumed that graph G has a node without any precursor task v_{start} as the starting node of the schedule and a node without any successor task v_{end} as the ending node of the schedule [6-8]. The weight of node v_i is denoted as cl_{v_i}, which represents the computational loading of task v_i. Suppose there are m different virtual machines VMs in the cloud, denoted as set M, and that the computing ability of the virtual machine m_j is ca_{m_j}. Each task can be executed on different virtual machines, where $t(v_i,m_j)$ represents the execution time of task v_i on virtual machine m_j. Thus,

$$t(v_i,m_j)=\frac{cl_{v_i}}{ca_{m_j}}. \qquad (1)$$

If the execution mode of a task is non-pre-emptive, the average execution time for task v_i is

$$\overline{T}_{v_i}=\sum_{j=1}^{m}\frac{t(v_i,m_j)}{m}. \qquad (2)$$

The weight of the directed edge $\langle v_i,v_k\rangle$ is denoted as ct_{v_i,v_k}, giving the communication time between task v_i and task v_k. If v_i and v_k are executed in the same virtual machine, then $ct_{v_i,v_k}=0$. The scheduling priority p_{v_i} of task v_i depends on the reverse recursion of DAG, *i.e.*, starting from the node v_{end},

$$p_{v_i}=\overline{T}_{v_i}+\max_{v_k\in succ(v_i)}\left(ct_{v_i,v_k}+p_{v_k}\right), \qquad (3)$$

where $succ(v_i)$ is the successor set of task v_i, and the value of p_{v_k} is the priority of the direct successor of task v_i. Because the priority is calculated in reverse, the priority of the ending node is

$$p_{v_{end}}=\overline{T}_{v_{end}}, \qquad (4)$$

where $EST(v_i,m_j)$, $EFT(v_i,m_j)$, and $LST(v_i,m_j)$ represent the earliest starting time, the earliest completion time, and the latest completion time of task v_i executed on virtual machine m_j. For starting node v_{start},

$$EST(v_{entry},m_j)=0. \qquad (5)$$

For the other tasks, EST, EFT, and LST are calculated according to

$$EST(v_i,m_j)=\max\left\{time(j),\max_{v_l\in pred(v_i)}\left(AST(v_l)+ct_{v_l,v_i}\right)\right\}, \qquad (6)$$

$$EFT(v_i,m_j)=t(v_i,m_j)+EST(v_i,m_j), \qquad (7)$$

and

$$LST\left(v_i, m_j\right) = \min_{v_k \in succ(v_i)} \left(AST\left(v_k\right) - ct_{v_i, v_k} \right), \tag{8}$$

where $pred(v_i)$ is the collection of the direct precursors of task v_i, $time(j)$ is the readiness time of virtual machine m_j, and $succ(v_i)$ is the collection of the direct successors of task v_i. The actual starting time and actual ending time of the execution of task v_i on virtual machine m_j are $AST(v_i, m_j)$ and $AFT(v_i, m_j)$, respectively. The values might be different from $EST(v_i, m_j)$ and $EFT(v_i, m_j)$, which is related to the readiness time of the resource. The maximum completion time is the actual completion time of the ending node according to

$$MS = AFT\left(v_{end}\right), \tag{9}$$

where MS is the execution time of the entire graph DAG. By calculating EST, EFT, and LST, the critical path of the entire graph can be obtained, which is the scheduling order of the critical tasks in the entire resource allocation schedule.

2.2. The Resource Scheduling Model in the Cloud Computing Environment

For cloud computing service providers, computing resources, such as virtual machines, have different computing powers and payment modes. The cost primarily depends on the computing power of the CPU, the memory size, and the bandwidth. Using the processing capability of the CPU as an index, a linear model can be selected to evaluate the cost, where $ca_{m_{slow}}$ refers to the computing ability of virtual machine m_{slow} with the slowest CPU. If Vc_{base} denotes the base cost of virtual machine m_{slow}, the cost of task v_i executed on the virtual machine m_j is

$$c\left(v_i, m_j\right) = \delta \times t\left(v_i, m_j\right) \times Vc_{base} \times \frac{ca_{m_j}}{ca_{m_{slow}}}, \tag{10}$$

where δ is a random variable used to generate virtual machines with different processing capabilities and costs. The total cost is

$$C = \sum_{j \in select} c\left(v_i, m_j\right). \tag{11}$$

For each task $v_i \in V$, the constraint function for both the minimum execution time and cost is

$$\min\left\{\omega \times T(i, j) + (1 - \omega) \times C(i, j)\right\} \tag{12}$$

$$s.t. \begin{cases} \omega \in [0,1], m_j \in M \\ T(i, j) = \dfrac{t\left(v_i, m_j\right) - t_{\min}}{t_{\max} - t_{\min}} \\ C(i, j) = \dfrac{c\left(v_i, m_j\right) - t_{\min}}{t_{\max} - t_{\min}} \end{cases}, \tag{13}$$

where ω is a weighting factor used to measure the user preferences, i.e., the weight of the execution time and cost and $T(i, j)$ and $C(i, j)$ are the normalised time and cost, respectively. Thus, $t_{\min(\max)}$ and $c_{\min(\max)}$ refer to the minimum (maximum) execution time and minimum (maximum) cost at any point of the execution scheduling process. Obviously, this is an NP-hard problem.

3. Virtual Resource Scheduling Based on the Improved Particle Swarm Algorithm

3.1. Improved Particle Swarm Optimisation (Ipso)

In the PSO algorithm, $X_i = [x_{i,1}, x_{i,2}, ..., x_{i,d}]$ and $V_i = [v_{i,1}, v_{i,2}, ..., v_{i,d}]$ represent the position and velocity of the particles, respectively. In addition, $pbest_i$ and $gbest_i$ represent the historical best position of an individual particle and the group, respectively. The updating methods for the velocity and position of the particles are [9-11]

$$\begin{aligned} V_{i+1} &= \omega * V_i + c_1 * rand * (pbest_i - X_i) \\ &\quad + c_2 * rand * (gbest_i - X_i) \end{aligned} \tag{14}$$

and

$$X_{i+1} = X_i + V_i , \tag{15}$$

where k is the current iteration number, c_1 and c_2 are the acceleration factors, ω is the inertial weight, and r_1 and r_2 are the random numbers in the range of 0 to 1.

The value of the velocity weight ω plays a role in balancing the global optimisation capability and the local optimisation capability. Therefore, we propose an IPSO that self-adaptively adjusts the value of the velocity weight ω when updating the position and velocity of each particle. The detailed procedure is as follows:

(1) Calculate the average fitness value of the particles in the swarm.

(2) Extract particles with fitness values greater than the average fitness value and calculate their average fitness value f_{av1}. The maximum value of the velocity weight is assigned to particles with fitness values greater than f_{av1}.

(3) Extract the particles with fitness values less than the average fitness value and calculate their average fitness value f_{av2}. The minimum value of the velocity weight is assigned to particles with fitness values less than f_{av2}.

(4) According to the self-adaptive strategy, the values of the velocity weights with linear variation between the maximum and minimum velocity weights with f_{av1} and f_{av2} are assigned to particles with fitness values between f_{av1} and f_{av2}. The updated IPSO equation for the position is

$$\begin{aligned} V_i &= \omega_i * V_i + c_1 * rand * (pbest_i - X_i) + \\ &\quad c_2 * rand * (gbest_i - X_i) \end{aligned} \tag{16}$$

$$\omega_i = \begin{cases} \omega_{max} & f_i > f_{av1} \\ \omega_{max} - \dfrac{(\omega_{max} - \omega_{min})(f_{av1} - f_i)}{f_{av1} - f_{av2}} & f_{av2} \le f_i \le f_{av1} \\ \omega_{min} & f_i < f_{av2} \end{cases} , \tag{17}$$

where ω_{min}, ω_{max}, and ω_i refer to the weight value with the minimum velocity, the weight value with the maximum velocity, and the weight value with the velocity of the current particle, respectively . The IPSO procedure is shown in Figure 1.

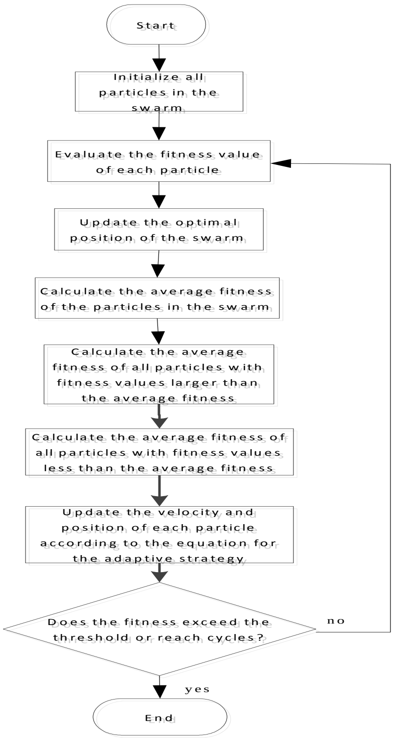

Figure 1. Flowchart of the Ipso Algorithm

3.2. Ipso-Based Virtual Resource Scheduling

Resource allocation models (12) and (13) are based on the time and cost constraints. They are integrated with IPSO algorithm such that the resulting resource scheduling algorithm in the cloud computing environment can be described as follows:

1. Set the values of the machine parameters and the weight of the nodes and edges in the DAG ;

3. Calculate the priority p_{v_i} for all of the nodes $v_i \in V$

4. The tasks v_i are sorted in descending order according to their priority

5. For each $v_i \in V$, do

6. For each $m_i \in M$, do

7. Calculate equation (12) based on the constraint condition (13)

8. End

9. End

10. For the nodes $v_i \in V$ ready for scheduling, do

11. The task is assigned to a virtual machine m_j for execution based on the IPSO algorithm

12. End

3. Simulation Results and Analysis

CloudSim is a simulation software platform for cloud computing that was jointly launched by the University of Melbourne, Australia and Gridbus. Our simulation was performed on this platform. By rewriting the bindCloudletToVm method, an algorithm for scheduling different tasks was developed such that the task scheduling algorithm in the cloud computing environment based on the IPSO algorithm could be tested and compared with the results for task scheduling based on the PSO algorithm. We used the Windows 7 operating system with a 2.50 GHz Intel Core i5-2450M processor and 4 GB of memory. The number of resource nodes in the cloud computing system was 10. To make the results of the IPSO algorithm more convincing, we compared them to the results from the PSO algorithm using the same experimental conditions.

The curves for the optimal solution variations of the PSO and IPSO algorithms are shown in Figure 2. The IPSO algorithm converged faster than the PSO algorithm. After 100 iterations, the IPSO obtained the optimal solution for resource scheduling in the cloud computing system, whereas the PSO algorithm required 250 iterations. This indicates that the introduction of a chaotic operation to the PSO algorithm ensures the diversity of the individuals in the particle swarm and prevents the emergence of local optimisations, leading to a better cloud resource scheduling solution.

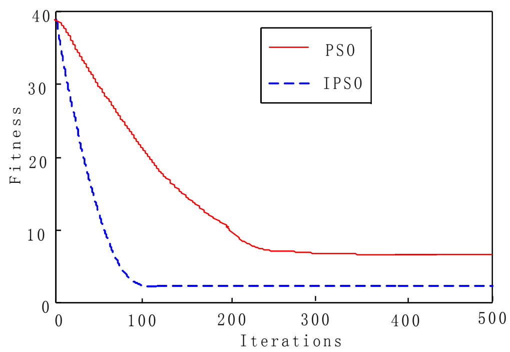

Figure 2. Comparison of the Convergence Performance for the IPSO and PSO Algorithms

The curves for task completion time variation with 200 tasks randomly assigned to 10 resource nodes are shown in Figure 3. When the number of nodes was increased, the competition for resources among the tasks was weakened. The task completion time was reduced for both the PSO and the IPSO algorithms, while the task completion time of the IPSO algorithm was relatively short. The comparison showed that the IPSO algorithm had a definitive advantage.

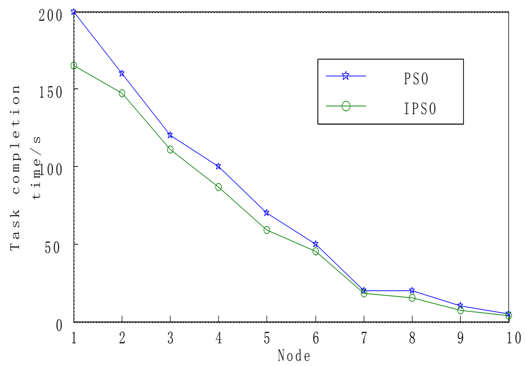

Figure 3. The Curve for Task Completion Time

The costs of the PSO and IPSO algorithms for 10 nodes are shown in Figure 4. The costs vary for different nodes, primarily because of their differences in processing capability. The IPSO algorithm had a better balance for the costs of the various nodes compared to the PSO algorithm.

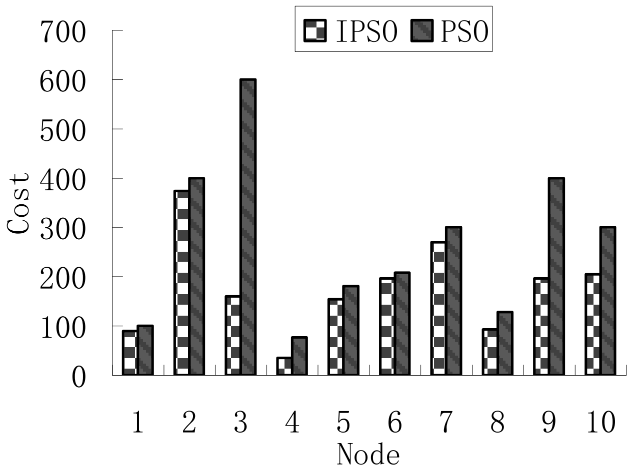

Figure 4. Load Distribution on Different Nodes

4. Conclusion

We proposed a resource allocation strategy for a cloud computing environment based on IPSO scheduling with dual constraints for time and cost. A particle swarm algorithm was introduced for scheduling resource allocation. Simulations showed that this algorithm could quickly and accurately allocate resources on virtual machines with a reduced total time for task scheduling in the cloud environment. In terms of the cost, the IPSO algorithm is obviously superior to the traditional PSO algorithm.

References

[1] R. Patel R and S. Patel, "Survey on Resource Allocation Strategies in Cloud Computing", International Journal of Engineering Research and Technology. ESRSA Publications, **(2013)**, pp.11-14.

[2] M. A. Arfeen, K. Pawlikowski and A. Willig, "A Framework for Resource Allocation Strategies in Cloud computing Environment", Computer Software and Application Conference Workshops (COMPSACW), **(2011)**, pp.261-266.

[3] H. Yuan H, C. Li and M. Du, "Optimal Virtual Machine Resources Scheduling Based on Improved Particle Swarm Optimization in Cloud Computing", Journal of Software, vol. 9, **(2014)**, pp.705-708.

[4] L. Wang and L. Ai, "Task Scheduling Policy Based on Ant Colony Optimization in Cloud Computing Environment", LISS 2012, **(2013)**, pp. 953-957, Springer Berlin Heidelberg.

[5] K. Zhu, H. Song and L. Liu, "Hybrid Genetic Algorithm for Cloud Computing Applications", IEEE Asia- Pacific Services Computing Conference, **(2011)**, pp. 182-187.

[6] R. Buyya, R. Ranjan and R. N. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities", HPCS, IEEE Press, pp. 1-11. 2009, New York, USA.

[7] J. T. Tsai, J. C. Fang and J. H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm", Computers & Operations Research, vol. 40, **(2013)**, pp. 3045-3055.

[8] Z. Xiao, W. Song and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment", IEEE Transactions on Parallel and Distributed Systems, vol. 24, **(2013)**, pp. 1107-1117.

[9] R. Poli, J. Kennedy and T. Blackwell, "Particle swarm optimization", Swarm intelligence, vol. 1, **(2007)**, pp. 33-57.

[10] X. Bo, Q. Guan and K. Chen, "Multi-Agent Coalition Formation Based on Quantum-behaved Particle Swarm Optimization", Journal of Information and Computational Science, **(2010)**, pp. 1059-1064.

[11] A. Engelbrecht, "Particle swarm optimization", Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion. ACM, **(2014)**, pp.381-406.

Author

Shao Youwei, 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.

