Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017
DOI: 10.15625/vap.2017.0003

ACCELERATE THE CLOUD TASK SCHEDULING WITH ANT COLONY OPTIMIZATION

Le Thanh Cong[1], Nguyen The Xuan Ly[2], Nguyen Tran Quoc Vinh[1]

[1] Informatics Faculty, University of Education, The University of Danang, Vietnam

[2] Information Technology Faculty, University of Science and Technology, The University of Danang, Vietnam

ltcong@dce.udn.vn, ntxly@dut.udn.vn, ntquocvinh@gmail.com

ABSTRACT: *Cloud computing is a type of parallel and distributed system consisting of interconnected physical and virtual machines. One of the fundamental issues in this environment is related to job scheduling. This work is a nondeterministic polynomial time-hard optimization problem, and many meta-heuristic algorithms have been proposed to solve it. A good task scheduler should adapt its planning strategy to the dynamic environment and the types of assignments. In this paper, the modified ant colony optimization to deal with cloud task scheduling is offered and compared to different algorithms such as First Come First Serve or Round Robin. The main goal of adjustment is to enhance the performance of ant colony optimization algorithm and apply it to minimizing the makespan of a given task set in Clouds. The improved algorithm is going to be deployed in the CloudSim toolkit where Round Robin pre-installed. The experimental results show that the suggested approach outperforms the preset one.*

Keywords: *Cloud Computing, Scheduling, Ant Colony Optimization.*

I. INTRODUCTION

Cloud computing has changed the way to store data and process the user tasks by exploiting strongly power of modern servers, dedicated applications as well as high network resources between them. Moreover, with the support of virtualization technology, cloud computing providers allow customers to choose services in a flexible manner by renting
virtual machines (VMs) [1]. Due to the using of hundreds or thousands of VMs, the problem is how to allocate tasks to computing resources in clouds [2]. Therefore, the finding of an effective algorithm to resolve the scheduling matter is necessary in clouds.

A good task scheduler should make suitable its planning strategy to the variable environment and the types of assignments [3]. This leads to the consideration of Ant Colony Optimization (ACO) which is a dynamic task scheduling
algorithm. This algorithm simulates the behavior of real ant colonies. Due to the natural characteristics and chemical properties, each ant while seeking food always leave pheromone trail along the way. This helps ants to get in touch with others. Furthermore, they often go on the path with denser pheromone. This attribute is emulated as positive feedback in the ACO algorithm. Besides, at the very beginning, ants will take the routes randomly. Thus, ACO algorithm is also a random search algorithm. These days, ACO is applied to solve nondeterministic polynomial time (NP)-hard problems such as travelling salesman problem, graph coloring problem, vehicle routing problem and scheduling problem [4, 5]. In this paper, the modified ACO to deal with cloud task scheduling is offered and compared to different algorithms such as First Come First Serve (FCFS) or Round Robin (RR). The main goal of adjustment is to enhance the performance of ACO algorithm and apply it to minimizing the makespan of a given task set in Clouds. The improved algorithm is going to be deployed in the CloudSim toolkit where RR pre-installed. Experimental results show that the suggested approach outperforms the preset one. The remaining of paper is arranged as follows. The next (second) section summarizes background and the related work. The enhanced ACO algorithm is going to depicted in the third section. The expected results while executing in the simulator are expressed in the fourth section. Finally, the fifth section contains our conclusions of this paper.

II. BACKGROUND AND RELATED WORK

A. Background

Refer to [6, 7], in a cloud system, the way which process a heavy workload has been changed. Local computers do not take the responsibility to handle significant lifting while running applications. The computer networks that composed the cloud resolve them instead. This leads to the decrease of hardware and software requirements in the customer site. Consumers just only interact with providers via Web browsers to give the task set.

The main power of cloud computing is supplied by the collection of datacenters installed with hundreds to thousands of servers to provide computing environment. According to [7], the combination of five essential features, three service models and four deployment models creates the fully cloud model. We should find out a little bit about them to see the benefits that bring by clouds. Firstly, looking at the features.

On-demand self-service: no needed to interact between consumers and service providers while taking computing capabilities such as server time and network storage.

Broad network access: users could access the network computing services no matter what they are using fixed or portable devices.

Resource pooling: basing on consumer demands, the provider's computing resources are pooled to serve multi–clients with flexible physical and virtual resources.

Rapid elasticity: computational resources and operations could provide for user demands elastically and rapidly. In some cases, they are even scaled out and released to scale in automatically. To the customers, the available provision of capabilities often occurs to be limitless and could be bought in any quantity at any time.

Measured service: the utilization of resources (storage, central processing units, bandwidth, etc.) could be monitored, controlled, and reported transparently for both providers and consumers.

As a result, reduction of costs, universal access, updated software, choice of applications, potential of green economy and flexibility are obvious profits on customer site while employing cloud computing. Next, explaining together why cloud service providers could give us the good things. The cloud platform is created from service and deployment models in which service ones include Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) whereas four types of deployment ones offer users the flexible selections. They are described briefly in the following figures.

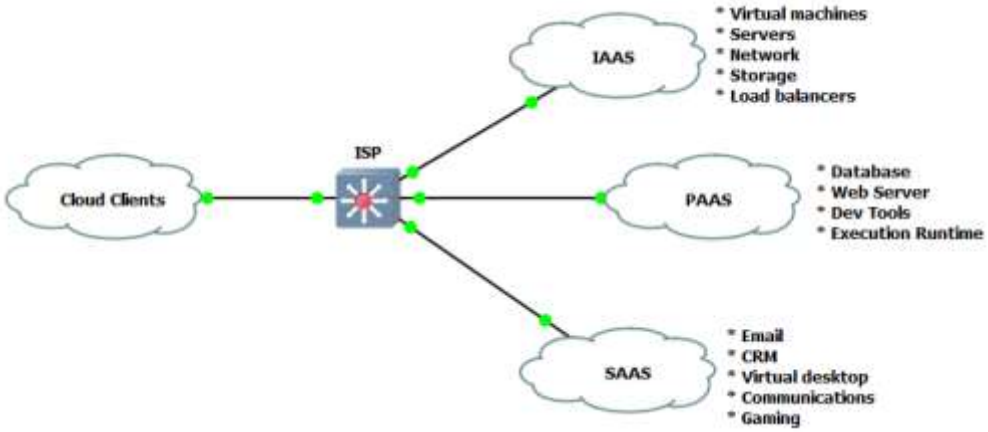

Figure 1. Cloud Service Models

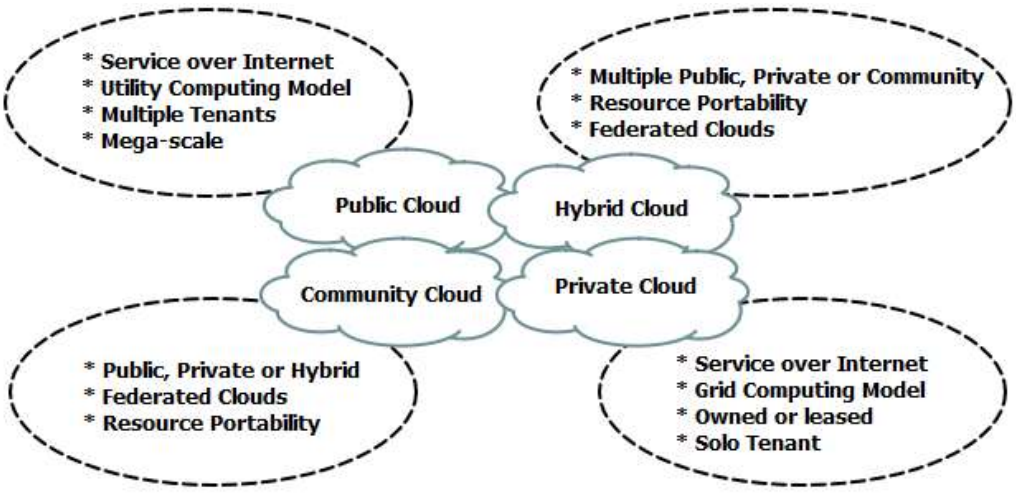

Figure 2. Cloud Deployment Models

Clearly, the burden of handling is concentrated on clouds where hundreds to thousands of virtual/ physical servers are connected. How to scheduling/load balancing to make jobs done in the best way? This means that the cloud task scheduling must fulfill the following requirements:

- Process the tasks as much as possible in one unit of time
- Maximum utilization of resources such as central processing units (CPUs), storage and bandwidth
- Avoid "deadlock"
- Finish the user jobs as soon as possible.

These requests would be addressed and given solution in the next sections.

B. Related Work

At present, the most widely used job scheduling in clouds is RR due to the simplicity and justice in time-sharing. This planning mechanism also pre-setup in the CloudSim as well. Its operations are recapped in the figure 3.

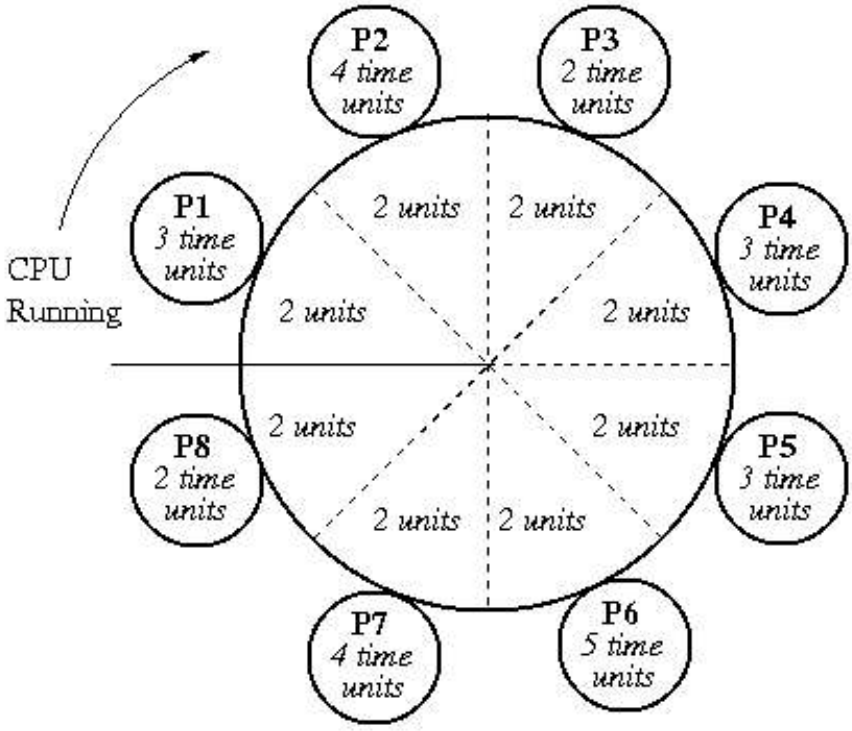

Figure 3. Round Robin operations

However, RR scheduling/ load balancing is the static one based on the fixed time quantum. Consequently, it is not enough for variable systems and reducing power consumption as clouds [8]. On the other hand, the ACO-based algorithm is the candidate to deal with dynamic environments and power saving in clouds [8].

In recent years, some researchers proposed the ACO-based approaches as the load balancing in the cloud or grid computing systems. However, their ACO methods were simplified in some parameters or result set. For example, Kumar Nishant *et al.* offered a way where the ants only updated a single result set continuously in the process, rather than updating their own result set [9]. Another study, the ACO solution of Ratan Mishra1 *et al.* just focused to maximize or minimize performance parameters, such as CPU load and memory capacity while plaining the chemical property of ants (pheromone) [10]. The mentioned drawbacks were repaired by implementing a quite complicated ACO-based scheduling with 10 ants [11]. This does not reflect the average size of ant colony in the reality (from 4000 to 8000 ants at age 3-4 years) [12]. Alternatively, the smaller colonies took more time in searching, with less achievements than the larger ones [13].

In the third section, we suggest the enhanced ACO algorithm which covers most of parameters in the ant moving procedure and the medium number of ants in practice. Then, applying it to speed up the cloud task scheduling.

III. ENHANCED ACO ALGORITHM

Mathematically, the Cloud job scheduling is the NP-hard problem in the graph G = (N, E) where N is the number of VMs and E is the Tasks – VMs connections. The following figure shows an example of task scheduling.

The ACO-based task scheduling uses a heuristic method to expect execution time of the task (i) on VM (j). In addition, the algorithm has been appended the short-term memo to satisfy constraint which avoid visiting a VM more than once per ACO process and minimize time of the assigned couplings (task and VM). Besides, pheromone updating rule is the essential of ant colonies because pheromone evaporates on all routes and new pheromone is deposited by all ants on visited paths. Its value is proportional to the solution quality built by the ants. The probabilistic transition rule is called random proportional. The above mentions are expressed in the following equations. Some of them are referred to [14] and adjusted to suit our algorithm.

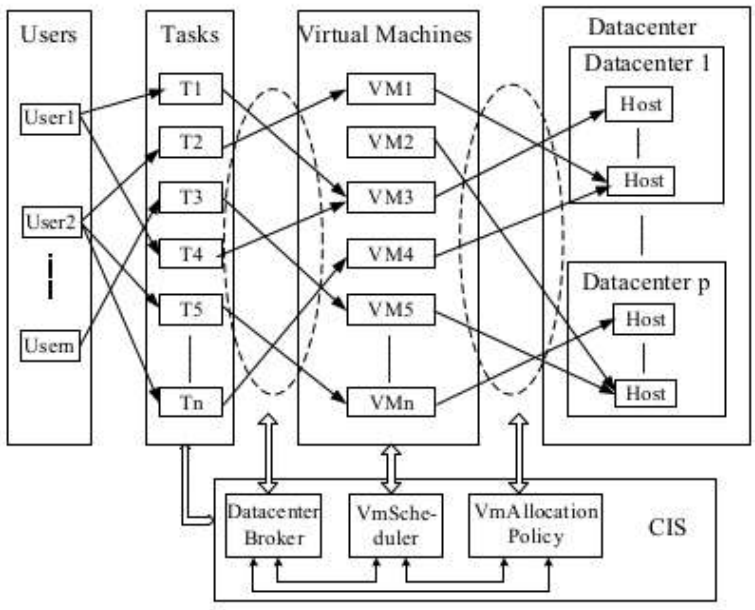

Figure 4. Cloud job scheduling example

- Equation (1): The k-ant chooses VM(j) for next task T(i)

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ji}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{s\in allowed_k}[\tau_{is}(t)]^{\alpha}.[\eta_{ij}]^{\beta}} & if\ j \in allowed_k \\ 0, otherwise \end{cases} \quad (Eq.1)$$

Where:

- $\tau_{ji}(t)$ displays the pheromone at the t time between cloudlet(i) and VM(j);
- $allowed_k$ is the list for short-term memory;
- α and β are accumulation of information and inspired information;
- $\eta_{ij} = \frac{1}{d_{ij}}$, a heuristic function and d_{ij} is execution time and transfer time of the task (i) on VM(j).

A. Improving the heuristic function

In term of basic ACO, the heuristic function is the same at the very beginning. This could make ants start the search blindly; thus, a large number of routes should be generated and the convergence would be slow. How to solve that situation? The expected execution time and transfer time d_{ij} has been enhanced as follows:

$$d_{ij} = \frac{Task\_Length_i}{numPe_j * MIPS_j\_VM_j} + DataSize * Metric\_VM_j \quad (Eq.2)$$

Where:

- $Task\_Length_i$ is the length of tasks which has been submitted by VM(j)
- $numPe_j$ is the number of processors in VM(j)
- $Metric\_VM_j = \frac{1}{Bandwidth\ of\ VM(j)}$
- $MIPS_j\_VM_j$ is the power of each processor in VM(j) based on Million Instructions Per Second

When an ant completes its tour, the local pheromone will be updated because each ant k lays a quantity of pheromone $\Delta\tau_{ij}^{k}(t)$ which is calculated by the Equation (3) as follows:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L^k(t)} & if\ (i,j) \in\ T^k(t) \\ 0 & if\ (i,j) \notin\ T^k(t) \end{cases} \quad (Eq.3)$$

Where:

- Q is an adaptive parameter
- $L^k(t)$ is the expected makespan of this tour found by Equation (4) when $T^k(t)$ is done by ant k at iteration t.

$$L^k(t) = \arg max\ _{j \in IJ} \{sum_{i \in IJ}(d_{ij})\} \quad (Eq.4)$$

- $\tau_{ji}(t)$ can become the intensity of VM (j) pheromone at time t in the equation below.

$$\tau_{ji}(t) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (Eq.5)$$

- With the trail decay ρ in which 0 < ρ < 1, we have $\Delta\tau_{ij}(t)$ by applying the equation (6):

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau\,{}_{ji}^{k} \quad (Eq.6)$$

An elite ant enforces pheromone on the edges which is the best tour found from the beginning of the trial T^+ when all ants complete their paths. $\frac{Q}{L^+}$ is a quantity and L^+ is the best tour's length. Those are so called global pheromone update followed by the equation (7):

$$\tau_{ij}(t) = \tau_{ij}(t) + \frac{Q}{L^+}\ if\ (i,j) \in T^+ \quad (Eq.7)$$

Extracting from the above equations, we create the Pseudo code of ACO-based task scheduling below:

Input: Cloudlets list (Tasks) and list of VMs
Output: the best solution for task scheduling on VMs
Step:
1. *Initial: α, β, ρ, Q, the numbers of ants, max iteration and max trial time*
2. *While iteration < iteration $_{max}$*
3. *For each ant*
4. *For each task*
5. *For each virtual machine*
6. *Calculate the probability by (1)*
7. *Choose the VM for next Task*
8. *Calculate the Length of Tasks*
9. *End for*
10. *Update local Pheromone by Equation (5)(6)*
11. *For each Virtual Machine*
12. *Adjust local pheromone if VM is selected*
13. *End for*
14. *End for*
15. *End for*
16. *Find the best solution in this iteration*
17. *iteration ++*
18. *Update global pheromone by Equation (7)*
19. *End while*
20. *Find the best solution for task by Length*

In the next part, the deployment of ACO algorithm would be operate and show us the achievements.

IV. IMPLEMENTATIONS AND RESULTS

A. ACO-based algorithm with CloudSim. Why?

In accordance with [15], the CloudSim toolkit has overcome some simulators such as GridSim, SimGrid, OptorSim GangSim and Open Cirrus in terms of cost and reflection ability of Cloud multi-layer service abstractions (SaaS, PaaS, and IaaS). The figure below shows the architecture of CloudSim. Its layered architecture consists of simulation engine, cloud services, configuration and user source code.

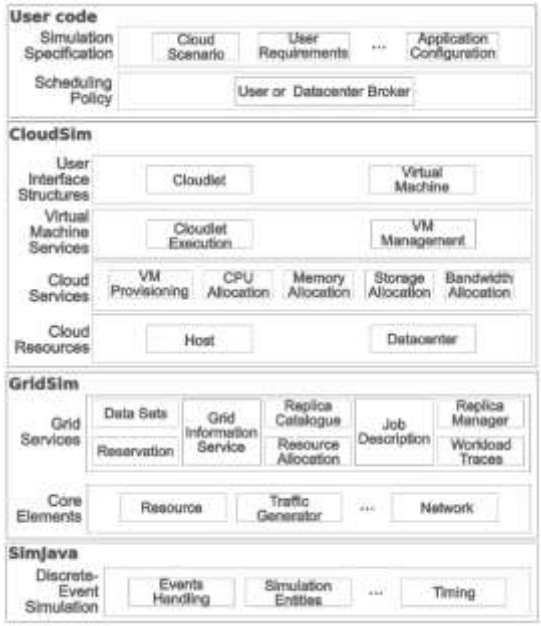

Figure 5. CloudSim architecture

While modelling and simulation of cloud environments as well as managing the dedicated interfaces, e.g. memory, storage, bandwidth and VMs, we could receive the support from the CloudSim layer. The monitoring of dynamic system state, administer of running applications and hosts to VMs are also provided. A cloud service provider can exploit the help from this layer to implement customized plans to know the efficiency of different policies in VM provisioning. The user code layer allows programmers to configure basic parameters such as the application types, Datacenter/ Host/ VM quantities as well as executing their applications and scheduling policies. In other words, it enables the capability of adding new features beside the basic ones. Thus, researchers could implement tests relied on specific scenarios and configurations. And now, our ACO-based algorithm is ready to launch with CloudSim.

B. Experimental Results

The following tables display the input parameters for CloudSim and our improved ACO-based task scheduling before performing the RR and ACO algorithms.

Table 1. CloudSim input parameters

Type	Parameters	Value
Datacenter	Datacenter quantities	10
	Host quantities	2 – 6
	Manager type	Time sharing
Virtual Machine	VM quantities	50
	Processors per VM	2 – 8
	RAM	256 – 4096
	Bandwidth	500 – 1000
Cloudlet	Cloudlet quantities	100 – 500 or 200 – 600
	Task length	1000 - 20000

In the ACO algorithm, the pheromone evaporation rate (ρ) is the key factor which impacts the global finding and convergence velocity. For this reason, we will preserve almost the ACO input parameters except ρ shown in Table 2.

Table 2. ACO input parameters

Parameters	Value
α	3
β	2
ρ	0.01 or 0.02
Q	8
Ant quantities	4000
Max iteration	8
Max trial time	100

For each number of cloudlets, we would run the test for five times and take the mean consequences of them. After finishing all operations, we have acquired the expected results in the figure 6. To note that these are the medium of five runs for each cloudlet quantities.

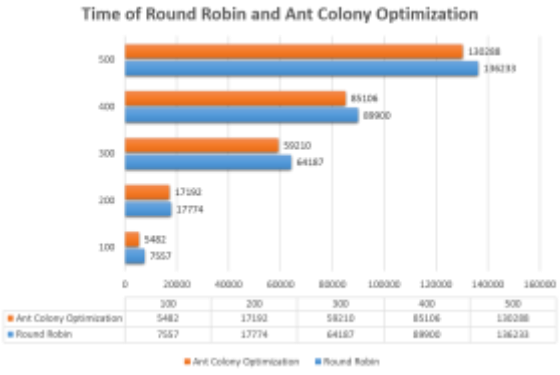

Figure 6a. ρ = 0.01

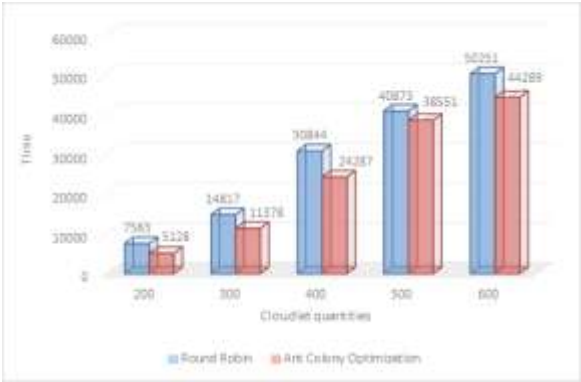

Figure 6b. ρ = 0.02

Figure 6. Performance of ACO vs. RR (Lower is better)

Clearly, RR is slower than the offered ACO algorithm in two cases (ρ = 0.01 or 0.02). Besides, the bigger pheromone evaporation rate (ρ=0.02) acquired the better results than smaller one (ρ = 0.01). Therefore, we would take a look at figure 6b and summarize the percentage of acceleration in time in the Table 3 as follows:

Table 3. Percentage of acceleration (ρ = 0.02)

ACO	RR	Acceleration (%)
5128	7563	32.2
11378	14817	23.21
24287	30844	21.26
38551	40873	5.68
44289	50251	11.86
Average acceleration (%)		18.84

Obviously, the cloud job scheduling has been accelerated by the enhanced ACO algorithm while comparing with the RR one. In the final section, we would conclude from this study that the offer algorithm should be applied to innovate the efficiency of cloud task scheduling.

V. CONCLUSIONS

In recent years, there has been a significant impact of biosciences on mathematics and computer technologies, leading to the genesis of a new science, technology, which uses biological principles to improve technology and information processes.

The ACO algorithm is attributed to techno-biology since they are based on the self-organization mechanisms for social insects. Within 10 years since its first proposal, the ACO algorithms have become an important field of theory of optimization.

While implementing ACO-based task scheduling algorithm with the number of jobs verified from hundreds to thousands, we have received the valuable results. The experimental ones demonstrate that the ACO algorithm accelerates the cloud task scheduling in term of comparing with RR. Furthermore, this helps to improve optimization problems such as task execution time, task transferring time and task execution cost. The virtual ants select the shortest routes by a probability rule based on the pheromone values and heuristic method for solving specific problems. The outcomes in this report are especially appropriate for dynamical systems with non-stationary parameters such as telecommunication or computer networks.

REFERENCES

[1] Fangzhe C., Ren J., and Viswanathan R., “Optimal Resource Allocation in Clouds”, The 3rd International Conference on Cloud Computing, Florida, USA, pp. 418-425, 2010.

[2] Qiyi H. and Tinglei H., “An Optimistic Job Scheduling Strategy based on QoS for Cloud Computing”, The IEEE International Conference on Intelligent Computing and Integrated Systems, Guilin, China, pp. 673-675, 2010.

[3] Gao K., Wang Q., and Xi L., “Reduct Algorithm Based Execution Times Prediction in Knowledge Discovery Cloud Computing Environment,” the International Arab Journal of Information Technology, vol. 11, no. 3, pp. 268-275, 2014.

[4] Dorigo M., Birattari M., and Stutzel T., "Ant Colony Optimization", IEEE Computational Intelligence Magazine, vol. 1, no. 4, pp. 28-39, 2006.

[5] Gao Y., Guan H., Qi Z., Hou Y., and Liu L., "A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing", Journal of Computer and System Sciences, vol. 79, no. 8, pp. 1230-1242, 2013.

[6] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities", *CoRR*, (abs/0808.3558), 2008.

[7] Peter Mell and Tim Grance, "The NIST Definition of Cloud Computing", National Institute of Standards and Technology, 2009.

[8] Pragya Matre, Sanjay Silakari and Uday Chourasia, "Ant Colony Optimization (ACO) Based Dynamic VM Consolidation for Energy Efficient Cloud Computing", International Journal of Computer Science and Information Security (IJCSIS), Vol. 14, No. 8, 2016.

[9] Kumar Nishant, Pratik Sharma, Vishal Krishna,. Chhavi Gupta and Kuwar Pratap Singh, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization", International Conference on Computer Modelling and Simulation (UKSim), Cambridge, UK, pp. 3-8, 2012.

[10] Ratan Mishra1 and Anant Jaiswal, "Ant colony optimization: A solution of load balancing in cloud", International Journal Web Semantic Technology, Vol. 3, No. 2, pp. 33-50, 2012.

[11] Arabi E. keshk, Ashraf B. El-Sisi, Medhat A. Tawfeek, "Cloud Task Scheduling for Load Balancing based on Intelligent Strategy", International Journal Intelligent Systems and Applications, No. 5, pp. 25-36, 2014.

[12] Deborah M. Gordon, Alan W. Kulig, "Founding, Foraging, and Fighting: Colony Size and The Spatial Distribution of Havester Ant Nests", Ecology, pp. 2393-2409, 1996.

[13] Tatiana P. Flanagan, Kenneth Letendre, William R. Burnside, G. Matthew Fricke, Melanie E. Moses, "Quantifying the Effect of Colony Size and Food Distribution on Harvester Ant Foraging", PloS ONE, Vol. 7, Issue 7, e39427, 2012.

[14] Tan Zhi and Zhang Hui, "An Improved Ant Colony Routing Algorithm for WSNs", Journal of Sensors, 2015.

[15] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose and Rajkumar Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Journal of Software and Experience, Vol. 41, pp. 33-50, 2011.

Lê Thành Công, Nguyễn Thế Xuân Lý, Nguyễn Trần Quốc Vinh

TÓM TẮT: *Điện toán đám mây là một dạng hệ thống song song và phân tán bao gồm các máy chủ vật lý và ảo được kết nối với nhau. Một trong những vấn đề cơ bản trong môi trường này liên quan đến lập lịch xử lý nhiệm vụ. Công việc này là một bài toán tối ưu thuật toán bất định trong thời gian đa thức, và nhiều thuật toán meta-heuristic đã được đề xuất để giải quyết vấn đề này. Một lịch xử lý nhiệm vụ được xem là tốt nếu thích ứng được với môi trường biến động và các loại công việc của điện toán đám mây. Trong bài báo này, thuật toán đàn kiến điều chỉnh được đưa ra để cải tiến việc lập lịch trên điện toán đám mây và so sánh với các thuật toán khác như First Come First Serve hoặc Round Robin. Mục đích chính của việc điều chỉnh là tăng cường hiệu năng của thuật toán đàn kiến và áp dụng nó để giảm thiểu thời gian xử lý của các tác vụ trong hệ thống đám mây. Thuật toán đàn kiến cải tiến sẽ được triển khai trong bộ công cụ CloudSim với Round Robin đã được cài sẵn. Kết quả thực nghiệm cho thấy thuật toán đề xuất tốt hơn so với thuật toán cài sẵn.*

Từ khóa: *Điện toán đám mây, Lập lịch, Thuật toán đàn kiến.*