An auto-scaling VM game approach for multi-tier application with Particle swarm optimization algorithm in Cloud computing

Khiet Bui Thanh
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology
Ho Chi Minh City, Vietnam
khietbt@tdmu.edu.vn

Lam Mai Hoang Xuan
Posts and Telecoms Institute of Technology
Ho Chi Minh City, Vietnam
maihoangxuanlam@gmail.com

Chien Nguyen Khac
Department of Mathematics - Informatics Ho Chi Minh City People's Police University
Ho Chi Minh City, Vietnam
nkchienster@gmail.com

Hung Ho Dac
Faculty of Engineering Technology
Thu Dau Mot University Binh Duong Provine, Vietnam
hunghd@tdmu.edu.vn

Vu Pham Tran
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology
Ho Chi Minh City, Vietnam
ptvu@hcmut.edu.vn

Hung Tran Cong
Training and Science Technology Department
Posts and Telecoms Institute of Technology
Ho Chi Minh City, Vietnam
conghung@ptithcm.edu.vn

***Abstract* —Cloud computing allows customers to scale their application to their needs. However, the problem of determining the amount of resources to be leased while still ensuring the quality of services and low cost is a big challenge. In this study, we focused on modeling the problem of auto-scaling virtual machines for multi-tier applications based on game theory. The strategy for auto-scaling resources is based on the Nash equilibrium and QoS parameter of the virtual machines. In the cloud computing environment, there is a need for scalability and high user responsiveness, so we design the algorithm - PSOVM to solve this problem based on PSO algorithm. Metaheuristic algorithms can find the near-optimal results in acceptable time.**

Keywords—autoscaling; PSO; cloud computing;

I. INTRODUCTION

Cloud computing is an increasingly popular technology because of the advantages it provides customers with the flexibility to deploy applications in a flexible way, simplifying the process of hiring and releasing resources. Today's technology services are largely based on the resources, operating mechanisms as well as the storage, distribution and processing of information in the cloud. Instead of using one or more PMs, users can use virtualization resources through the internet environment in the underlying service model (IaaS) or use the underlying service (PaaS) consisting of APIs for developing applications on a specific technology platform or software services (SaaS) - most of which are provided as a web-based application and accessed from far distances.

Cloud computing environment allows customers to dynamically scale up/down their application [1]. Specifically, in the IaaS infrastructure service, the cloud must be able to dynamically scale up/scale down resources assigned to the application at different times to improve performance, maintain system stability [2]. While cloud computing allows applications to adapt to changing needs, it is not an easy task to make the right decision. This work needs a system that scales the resources to the workload handled by the application. Accordingly, autoscaling of resources can be adjusted horizontally or vertically. In the horizontal scaling, the resource unit is the virtual machine replica. In contrast, the vertical scaling involves changing the resources assigned to a running virtual machine, for example, increasing (or decreasing) the allocated CPU power or memory. The most common operating systems do not allow on-the-fly changes (no restart) on the machine that it runs (even if it's a VM); for this reason, most cloud providers only provide horizontal expansion.

Autoscaler decisions must ensure the desire among the stakeholders involved. For customers, they want low cost of service, while service providers want to maximize their profit. Resource pricing models may include virtual machine types, cost per unit of time (per minute, per hour). The autoscaler must also ensure that the functionality of the application is properly implemented by maintaining QoS. QoS usually depends on two types of service level agreement (SLA): SLA application is a contract between clients (the application owner) and the end user; and SLA resources are provided by the vendor and the customer. Both types of SLAs are often mixed to satisfy SLA applications, vendors need to adhere to SLA resources. However, identifying the right amount of resources for rental and meeting the required service level (SLA) level while keeping the overall low cost are a big challenge. There are many algorithms for automatically coordinating resources that have been developed but no algorithms are suitable for all applications [3-5]. In the cloud environment, customers and

service providers often have different requirements and may be in conflict with others. Therefore, autoscaling of resources on cloud computing is a big challenge. The solution to this problem is often based on the specific characteristics of each problem, from algorithms such as exhaustive algorithm, deterministic algorithm [6] or metaheuristic [7-9]. In practice, almost all algorithms define exhaustive algorithms rather well . However, exhaustive algorithms are not effective in distributed data environments since they are not suitable for scheduling problems in the extensibility environment. [10]. At the same time, cloud computing is a distributed data environment that requires scalability and high user responsiveness so that it can address the problem of dynamically adjusting virtual machines on the fly. Metaheuristic algorithm for cloud computing is feasible although metaheuristic algorithms can give near-optimal results in acceptable time.

In this study, we propose a solution to resources autoscaling to ensure quality of objectives and the cost of renting resources based on game theory and metaheuristic algorithms in particular to optimize the Particle Swarm Optimizarion (PSO) to find an optimal or near optimal optimization solution based on Nash equilibrium. Following sections are presented as follows. Part II is the related work. The autoscaling method VM is presented in Part III. Part IV shows the method of autoscaling the virtual machine based on PSO. Part V is empirical to evaluate the method proposed. Finally, the conclusion and direction for further development are presented in Section VI.

II. RELATED WORKS

Each autoscaling method has been designed with specific objectives, focusing on application architecture or cloud provider and offering different tuning capabilities. It is possible to base on the technical specifications and applied theoretical foundations, which can be divided automatically in such directions as: (i) threshold-based, (ii) control theory, (iii) reinforcement learning, (iv) queue theory and (v) time series analysis. The autoscaling process mainly involves the analysis and planning phases of the MAPE loop. Queue theory and time series analysis are useful in the analysis phase to estimate current demand or future demand of the application. Threshold-based rules, reinforcement learning, and control theory can be used in the planning phase to determine corrective action, and they can be combined with an earlier related analysis, for example, time series analysis.

Ruiqing Chi et al. proposed a method of automatic resource scaling for multi-tier web applications based on game theory [11]. The model ensures QoS requirements and at the same time minimizes the cost of resource use. Based on game theory, they apply the Nash equilibrium concept, propose an evolutionary algorithm, and use the Layered Queuing Network (LQN) to predict correct application behavior across a range workload and resource allocation. They resolve resource allocation issues for multiple-tiered web applications in the cloud.

Huang et al. have proposed an autoscaling algorithm [5] which can make web applications that not only meet the needs of customers but also use less resources, improve resource utilization and reduce deployment costs. Designing automated virtual machine tuning mechanism that automatically adjusts the resources of the application is the best way to predict resources by using queue theory. Compared with traditional methods of using peak load resources, this method can improve the efficiency of resource utilization and service quality assurance.

Wei-Hua Ba et al. [3] propose a method for evaluating the performance of an application on the cloud. Based on the average response time, average waiting time, and other performance indicators to access the traffic density usage of each server, as well as the configuration server cluster, in a heterogeneous data center. Using queue theory, they build a complex queue model consisting of two serial queue systems and represent as an analytical model to evaluate the performance of heterogeneous data centers. The complex queue model provides high precision results for each performance measure and enables heterogeneous data center heterogeneity analysis.

Massimo Ficco et al. propose a meta-heuristic approach for cloud resource allocation based on bio-inspired modeling ecosystems [12]. Accordingly, classic game theory to optimize resource allocation strategies ensure the goals of the service provider as well as the requirements of the customer. Evolutionary algorithms based on coral reef structure and coral reproduction show a very interesting behavior to simulate the continuous requirements of resources, trigger the change process size, scaling-up, scaling-down and migrating. It also exploits the dynamics of competition among strategic users (with their SLAs) and service providers (with the goal of maximizing revenue) to converge optimal solutions which can meet the clear benefits of the stakeholder involved. Experiments show that the combination of bio-based and game-based approaches not only achieves a satisfactory solution of adaptability and elasticity but also can lead to significant performance improvements. On time convergence, the problem of scale under the huge clouds with a lot of machines and virtual machines are reallocated.

Yang Guo et al. proposed the autoscaling algorithm for VM hosting applications on the cloud [4]. The goal is to minimize the number of PMs that are stored by encapsulating virtual machines into PMs, while virtual machines are automatically tuned to meet the changing requirements. The Shadow algorithm uses a specially designed virtual queue system to automatically create an optimal solution for automatic VM tuning and VM-to-PM packaging. Suggested algorithms run continuously without having to re-solve the underlying optimization problem "from scratch" and autoscalingto changes in application demand.

III. AUTOSCALING VM APPROACH

A. System model

Clouds provide resources for applications. The MAPE-K loop monitoring module continuously monitors the resource usage. The monitor collects data on resource usage status and passes it to the autoscaler then issues commands to

autoscalingresources - specifically, increasing/decreasing the number of virtual machines served. In order to access QoS for each customer's multi-tier application, we used the Layer Queuing Network (LQN)[13, 14]. This is an analytical model based on Queueing Networks (QNs). This model is well suited to model complex software applications that have the hardware resources that these software entities run on to combine different levels of detail in the model. The LQN performance model can easily be integrated with the Software Development Life Cycle (SDLC). LQNs are ideal for representing inter-actions and complexity of multi-tier applications.

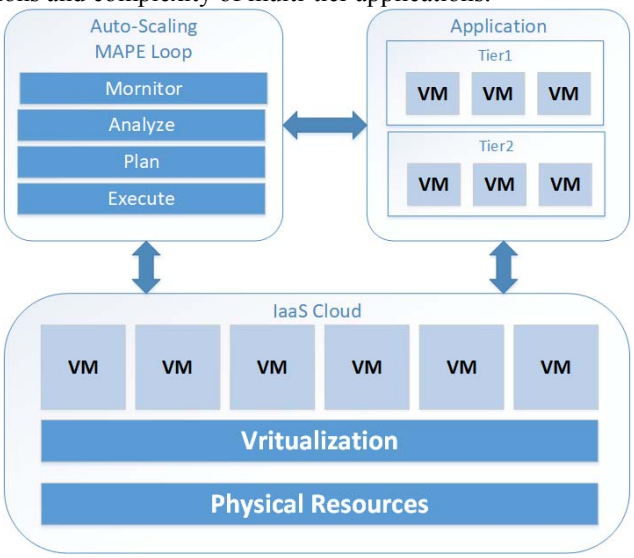

Figure 1. System Architecture

In this study, the LQN model provides performance measurements such as steady state flux and response times. Input to the LQN model is the hardware resources, the intensity of customer work, and customer service needs for the component model at each stage.

B. Autoscaling game model

Supposing in the cloud infrastructure, there are M physical machines (PM). Thanks to virtualization technology, PMs can deploy VMs on their own. The cloud computing system provides VM resources for multi-tier applications $A = \{A_1, A_2, \dots, A_n\}$. A resource allocation vector $\Phi = \{\Phi^1, \Phi^2, \dots, \Phi^n\}$ defines the number of VM copies allocated to each application at all servers so reasonably. VM allocation strategy for each application A_i represented by a matrix Φ^i non-negative of the row k for each tier and m the column for each PM is as follows:

$$\Phi^i = \begin{pmatrix} v_{11}^i & v_{12}^i & \cdots & v_{1m}^i \\ v_{21}^i & v_{22}^i & \cdots & v_{2m}^i \\ \vdots & \vdots & \ddots & \vdots \\ v_{k1}^i & v_{k2}^i & \cdots & v_{km}^i \end{pmatrix}, \quad (1)$$

where $v_{km}^i > 0$ is the number of VMs allocated to the application i at the tire k on the PM m.

A multi-tier application A^i serve $T_i = \{t_{i1}, t_{i2}, \dots, t_{iy}\}$, y is the number tasks. To ensure quality of service for users of the application A^i, we consider the degree of satisfaction of user requirements based on task response time, as follows:

$$R_i = \sum_{j=1}^{y} 1 - \frac{1}{1+e^{r_{ij}^T - r_{ij}}}, \quad (2)$$

where, r_{ij} response time of the application i for task j, r_{ij}^T is the expected response time of the application i for task j. By applying LQN, r_{ij} calculated as follows:

$$r_{ij} = \sum_{\alpha=1}^{k} \frac{s_{ij}^{\alpha}}{1 - \sum_{\beta=1}^{y} \frac{w_{i\beta}^{\alpha}}{\sum_{\delta=1}^{m} v_{i\delta}^{\alpha}} s_{i\beta}^{\alpha}}, \quad (3)$$

where,

- s_{ij}^{α} is the average service time of the application i for task j at tier α,
- $s_{i\beta}^{\alpha}$ is the average service time of the application i for task β at tier α,
- $w_{i\beta}^{\alpha}$ workload of task β of application i at tier α,
- $v_{i\delta}^{\alpha}$ is number of VM of application i at tier α allocated on PM δ.

In this study we calculated the cost of service based on the number of CPU allocated for the application. Supposing, x is the number of resources allocated to the PM i. If more resources are used, the customer has to pay more. Costs are proportional to the use of resources and time to complete the work. So, costs x is shown:

$$p_i(x) = ax + b \ (0 < x < c_i), \quad (4)$$

where, a, b is constant, c_i is the total number of CPUs of the PM i.

Decisions to autoscaling of resources are based on retrieving resource information from the cloud environment monitoring center. We consider the autoscaling of resources as a game and the customer as the player. Each player tries to maximize the resource extraction by adjusting the strategy ϕ^i. The players know each other's strategic information and decision points, so we can set up a cooperative game and have perfect information. From there we approach the notion of Nash's effective Pareto equilibrium of the game as a point where no player gets more harvest by changing his strategy [15]. To represent the tradeoff between quality of service and cost, the playoff function gives the i th player when served the VM as shown below:

$$F^i = \tau R_i + (1-\tau) \sum_{x=1}^{m} \sum_{y=1}^{k} p_x(v_{yx}^i) \ \forall \tau \in [0,1] \quad (5)$$

Assume that each player can change their allocation strategy by one of the following three actions:

$$a_i = \begin{cases} v_{yx}^i + 1 \\ v_{yx}^i - 1 \\ v_{yx}^i + 1, v_{zx}^i - 1 \end{cases}, \quad (6)$$

where, $v_{yx}^i + 1$ is the action of adding a VM to the application i at the tier y on the PM x, on the other hand $v_{yx}^i - 1$ is the act of modifying a VM and $v_{yx}^i + 1, v_{zx}^i - 1$ is the VM migration action from the PM y to the PM z.

In this game, the benefits function of the game has an important influence on a player's strategic decision and the outcome of the game. Each player will choose a strategy to maximize their harvest, so the objective function is as following:

$$Min\ \sum_{i=1}^{n} F^i(\Phi) \quad (7)$$

$$Subject\ to \sum_{i=1}^{n}\sum_{x=1}^{m}\sum_{y=1}^{k}\left(v_{yx}^{i}\right) \le c_x.$$

The Nash equilibrium of the game is a strategy in which no player can increase profits when other players have fixed strategies. Then, if the strategy of the player i is the optimal strategy to be denoted p_i^* , Optimal strategies of other players are denoted by p_{-i}^* Nash's balance of strategy p_i^* will be subject to conditions [16], as follows:

$$F_i(p_{-i}^*, p_i^*) \ge F_i(p_{-i}^*, p_i) \quad (8)$$

In a multi agent system, the equilibrium point may be unstable [17]. In addition, it is difficult to find the Pareto-efficiency of the Nash equilibrium. To solve this problem, most algorithms are based on metaheuristic algorithms. The options for assigning VMs to possible PMs are based on the optimum algorithm. From the set of feasible options that rely on Nash equilibrium conditions will pick the best option. Stop condition of algorithm according to [15].

$$\sum_{i=1}^{n}(F_j^{itr} - F_j^{itr-1})^2 < \varepsilon \quad (9)$$

IV. AN AUTOSCALING VM BASED PSO

Particle Swarm Optimization (PSO) is a randomized optimization technique based on a population developed by Eberhart and Kennedy that simulates the behavior of birds or fish [18]. In PSO, each single solution is a particle in the above scenario. Each element is characterized by two parameters that are the current position of the element $present$ and velocity v. These are two vectors on the numeric field R_n with n is the dimension of the element determined from the particular problem. Each element has an adaptive value $fitness\ value$, evaluated by the adaptive measure$fitness\ function$. At the time of departure, the herd, or precisely the position of each element, is randomly generated (or in some way based on prior knowledge of the problem). In the process of moving, each element is influenced by two information p_{Best} is the best position that the element has achieved in the past, g_{Best} is the best position that the herd has achieved in the past. In the original by Eberhart and Kennedy propose, Elements in the PSO will browse the problem space by following the elements with the best current conditions - the greatest adaptation. Specifically, after each discrete time interval, the velocity and position of each element are updated according to the following formulas:

$$v = v + c1.rand() * (p_{Best} - present) + c2.rand() * (g_{Best} - present) \quad (10)$$

$$present = present + v \quad (11)$$

where, $rand()$ is a random number in the interval $(0,1)$; $c1, c2$ is the learning coefficient.

A. Endcoding Scheme

In order to apply the PSO to solve the autosacling problem, the VM needs to redesign the endcoding scheme and benefit functions. For example, the system has 5 PMs serving assuming 3 applications $A = \{A_1, A_2, A_3\}$, each of which has three layers deployed as shown in Figure 2, include:

- App1 application is deployed on 6 VMs including: Tier1 has 2 VMs at PM1 and PM2; Tier 2 has 2 VMs in PM2 and PM4, Tier 3 has 2 VMs in PM3 and PM5.
- App2 application is deployed on 5 VMs including: Tier1 has 2 VMs at PM3 and PM5; Tier2 has 2 VMs at PM3 and PM5, and Tier3 has 1 VM at PM2.
- App3 application is deployed on 4 VMs including Tier1 has 1 VM at PM4; Tier2 has 1 VM at PM1, Tier3 has 2 VMs at PM1 and PM4.

With the VM deployment model for multi-tier applications as shown in Figure 2, you can demonstrate the VM deployment strategy described in (1) for each of these applications:

$$\Phi^1 = \begin{matrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{matrix} \qquad \Phi^2 = \begin{matrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{matrix}$$

$$\Phi^3 = \begin{matrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{matrix}$$

B. Algorithm

PSOVM algorithm

Input: $Problem_{size}, Population_{size}$
Output: $P_{g\_best} Population \leftarrow \emptyset$
// Initialize the VM deployment strategy $\Phi = \{\Phi^1, \Phi^2, \dots, \Phi^n\}$
$P_{g\_best} \leftarrow \emptyset$
For($i = 0$ **To** $Population_{size}$)
  $P_{velocity} \leftarrow$ RandomVelocity()
  $P_{position} \leftarrow$ RandomPosition($Population_{size}$)
  $P_{p\_best} \leftarrow P_{position}$
  // Cost () function is calculated by the formula (5)
  If (Cost(P_{p_best}) < Cost(P_{g_best}))
    $P_{g\_best} \leftarrow P_{p\_best}$
  End
End
While (¬StopCondition())
  For ($P \in$ Population)
    $P_{velocity} \leftarrow$UpdateVelocity($P_{velocity}, P_{g\_best}, P_{p\_best}$)
    $P_{position} \leftarrow$UpdatePosition($P_{position}, P_{velocity}$)
    If (Cost($P_{position}$) ≤ Cost(P_{p_best}))
      $P_{p\_best} \leftarrow P_{position}$
      If (Cost(P_{p_best}) ≤ Cost(P_{g_best}))
        $P_{g\_best} \leftarrow P_{p\_best}$
      End
    End
  End
End
Return (P_{g_best})

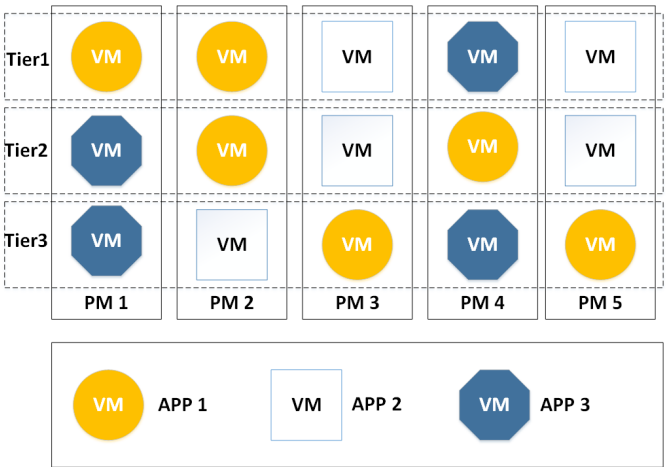

Figure 2. System deployment scheme

Through, each of PSOVM's interation algorithm can find strategies that can be allocated VMs to applications as following:

Strategy 1

$$\Phi^1 = \begin{matrix} 0 & 2 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 2 \\ 1 & 0 & 0 & 3 & 0 \end{matrix}$$

$$\Phi^2 = \begin{matrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 1 & 1 \end{matrix}$$

$$\Phi^3 = \begin{matrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 0 & 1 & 1 \end{matrix}$$

Strategy 2

$$\Phi^1 = \begin{matrix} 1 & 2 & 4 & 1 & 2 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{matrix}$$

$$\Phi^2 = \begin{matrix} 0 & 2 & 4 & 0 & 2 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{matrix}$$

$$\Phi^3 = \begin{matrix} 1 & 2 & 4 & 1 & 2 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{matrix}$$

V. EXPERIMENTS

A. Introduction

In this experiment we are concerned with the quality of the application and the cost of the service. With the PSOVM dark method, the approximation results depend on the parameter ε, the number of individuals in the $swarm_{sizes} = 30$, the tradeoff between service quality and VM rent τ . Therefore, in the following experiments, we find the appropriate parameters for the system as well as evaluate the scaling of VM resources for the application through the quality of service in the formula (4) and the cost of renting the resource according to the formula (5). We build algorithms and experiments based on the CloudSim 3.0 toolkit. We build a datacenter that has 10 PMs.

B. Resutls

Select ε in the case of user-directed processing, direction of data storage, and both. Experiment on 10 PMs and 70 applications. For ε from 0.001 to 0.01, measure the number of cycles set by the algorithm.

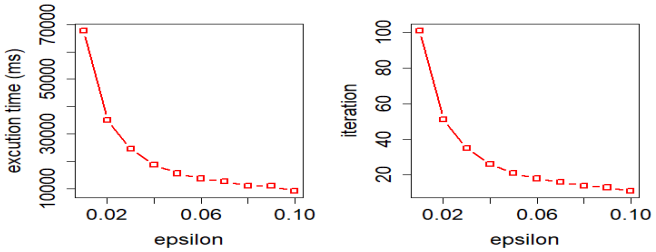

Figure 3. Effectiveness of ε

Select epsilon = 0.03, increase the number of display from 5 to 50 individuals

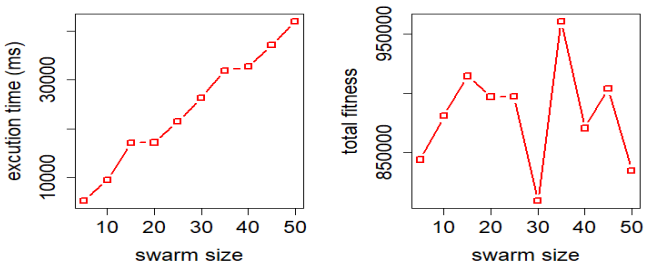

Figure 4. The Effect of the number of individuals in the swarm

In Figure 4 shows that increasing the number of individuals in the swarm then the time increase almost linearly. However, the number of individuals in the swarm initially ranged from 10 to 20 individuals, but the target value increased, but in the individual population 30 the target function was reduced to min. Continuing to increase the number of individuals to 35, the target function value jumps up. The number of individuals increased to 50, the target price dropped. From there, it can be seen that the number of individuals in the swarm if too little or too high leads to the disadvantage of finding the optimal value and execution time of the virtual machine. In Figure 4 above, the min value of the target function is reached when the number of individuals is 30 and the max value is when the number of individuals is 35.

In the following experiment, we selected the number of swarms as 30 and 35 to compare the effectiveness in Figure 5 - 8. We analyzed the effect of the application on the number of swarm size.

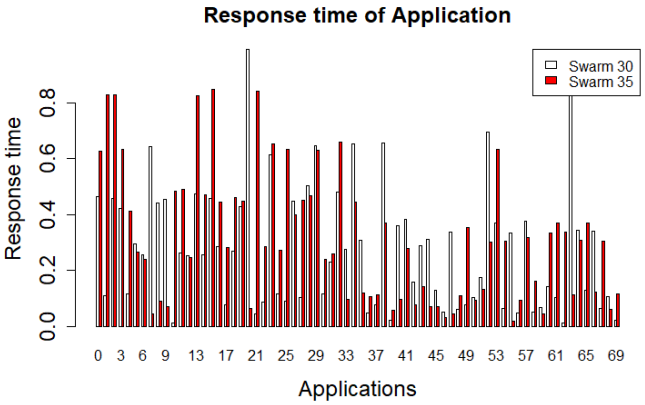

Figure 5. Response time of application

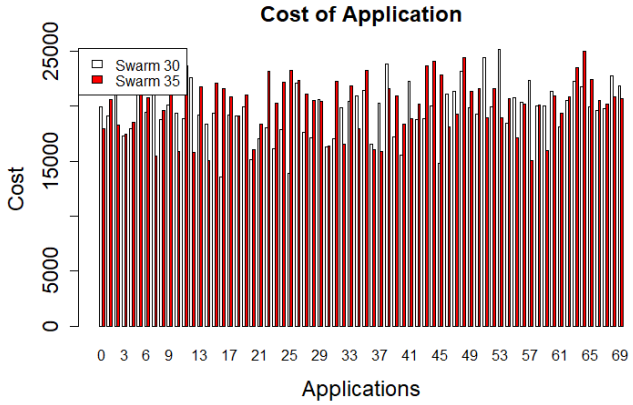

Figure 6. Cost of application

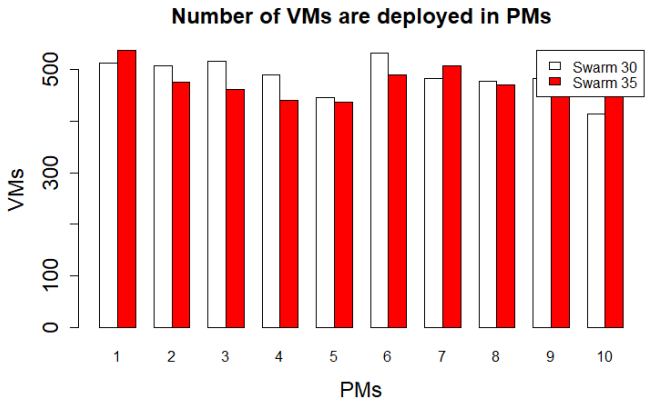

Figure 7. VMs are deployed in PMs

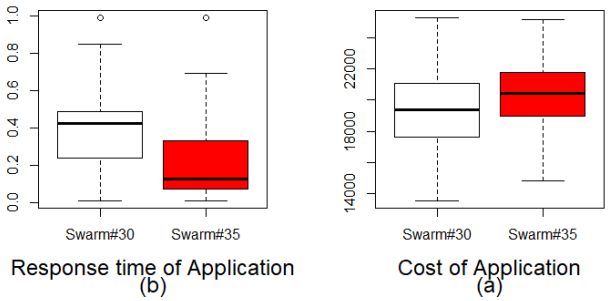

Figure 8. Algorithm result comparison

Figure 8 shows the overall comparison between Swarm#30 and Swarm#35. Swarm#35 provide less latency than Swarm#30 but it take greater cost. The difference ratio of cost is less than the difference ratio of response time, so we can choose proper swarm parameter to ultilize response time and cost.

VI. Conclusions

In this paper, we present a model for autoscaling VM resources for multi-tiered applications based on game theory cloud computing. The resource scaling strategy found in the PSOVM algorithm is based on the PSO algorithm. In the experimental part we present the parameters affecting the efficiency of the algorithm, by adjusting the parameters such as $\varepsilon, swarmsize, \ldots$, ... In the next research, we will improve the efficiency of the algorithm by combining with the ability to learn configuration parameters.

References

[1]. Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J.A.: 'A review of auto-scaling techniques for elastic applications in cloud environments', Journal of grid computing, 2014, 12, (4), pp. 559-592

[2]. Kumar, Y.R., Madhu Priya, M., and Shahu Chatrapati, K.: 'Effective distributed dynamic load balancing for the clouds', International Journal of Engineering Research & Technology (IJERT), 2013, 2, (2), pp. 1-6

[3]. Bai, W.-H., Xi, J.-Q., Zhu, J.-X., and Huang, S.-W.: 'Performance analysis of heterogeneous data centers in cloud computing using a complex queuing model', Mathematical Problems in Engineering, 2015, 2015

[4]. Guo, Y., Stolyar, A., and Walid, A.: 'Online VM Auto-Scaling Algorithms for Application Hosting in a Cloud', IEEE Transactions on Cloud Computing, 2018

[5]. Huang, G., Wang, S., Zhang, M., Li, Y., Qian, Z., Chen, Y., and Zhang, S.: 'Auto scaling virtual machines for web applications with queueing theory', in Editor: 'Book Auto scaling virtual machines for web applications with queueing theory' (IEEE, 2016, edn.), pp. 433-438

[6]. Morton, T., and Pentico, D.W.: 'Heuristic scheduling systems: with applications to production systems and project management' (John Wiley & Sons, 1993. 1993)

[7]. Van Laarhoven, P.J., Aarts, E.H., and Lenstra, J.K.: 'Job shop scheduling by simulated annealing', Operations research, 1992, 40, (1), pp. 113-125

[8]. Colorni, A., Dorigo, M., Maniezzo, V., and Trubian, M.: 'Ant system for job-shop scheduling', Belgian Journal of Operations Research, Statistics and Computer Science, 1994, 34, (1), pp. 39-53

[9]. Ghumman, N.S., and Kaur, R.: 'Dynamic combination of improved max-min and ant colony algorithm for load balancing in cloud system', in Editor (Ed.)^(Eds.): 'Book Dynamic combination of improved max-min and ant colony algorithm for load balancing in cloud system' (IEEE, 2015, edn.), pp. 1-5

[10]. Tsai, C.-W., and Rodrigues, J.J.: 'Metaheuristic scheduling for cloud: A survey', IEEE Systems Journal, 2014, 8, (1), pp. 279-291

[11]. Chi, R., Qian, Z., and Lu, S.: 'A game theoretical method for auto-scaling of multi-tiers web applications in cloud', in Editor: 'Book A game theoretical method for auto-scaling of multi-tiers web applications in cloud' (ACM, 2012, edn.), pp. 3

[12]. Ficco, M., Esposito, C., Palmieri, F., and Castiglione, A.: 'A coral-reefs and game theory-based approach for optimizing elastic cloud resource allocation', Future Generation Computer Systems, 2018, 78, pp. 343-352

[13]. Franks, G., Maly, P., Woodside, M., Petriu, D., and Hubbard, A.: 'Layered Queueing Network Solver and Simulator User Manual (2005)', Carleton Univ

[14]. Jung, G., Joshi, K.R., Hiltunen, M.A., Schlichting, R.D., and Pu, C.: 'Generating adaptation policies for multi-tier applications in consolidated server environments', in Editor: 'Book Generating adaptation policies for multi-tier applications in consolidated server environments' (IEEE, 2008, edn.), pp. 23-32

[15]. Xu, X., and Yu, H.: 'A game theory approach to fair and efficient resource allocation in cloud computing', Mathematical Problems in Engineering, 2014, 2014

[16]. Osborne, M.J., and Rubinstein, A.: 'A course in game theory' (MIT press, 1994. 1994)

[17]. Pendharkar, P.C.: 'Game theoretical applications for multi-agent systems', Expert Systems with Applications, 2012, 39, (1), pp. 273-279

[18]. Kennedy, J.: 'Particle swarm optimization': 'Encyclopedia of machine learning' (Springer, 2011), pp. 760-766