AN OPEN JACKSON NETWORK MODEL FOR HETEROGENEOUS INFRASTRUCTURE AS A SERVICE ON CLOUD COMPUTING

Chien Nguyen Khac[1, 2], Khiet Bui Thanh[3, 4], [4]Hung Ho Dac, [2]Son Nguyen Hong[3]Vu Pham Tran and [2]Hung Tran Cong

[1]Department of Mathematics – Informatics, The People's Police University, Ho Chi Minh City, Vietnam

[2]Training and Science Technology Department, Posts and Telecoms Institute of Technology Ho Chi Minh City, Vietnam

[3]Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Vietnam

[4]Faculty of Technology Engineering, Thu Dau Mot University, Vietnam

ABSTRACT

Cloud computing is an environment which provides services for user demand such as software, platform, infrastructure. Applications which are deployed on cloud computing have become more varied and complex to adapt to increase end-user quantity and fluctuating workload. One popular characteristic of cloud computing is the heterogeneity of network, hosts and virtual machines (VM). There were many studies on cloud computing modeling based on queuing theory, but most studies have focused on homogeneity characteristic. In this study, we propose a cloud computing model based on open Jackson network for multi-tier application systems which are deployed on heterogeneous VMs of IaaS cloud computing. The important metrics are analyzed in our experiments such as mean waiting time; mean request quantity, the throughput of the system. Besides that, metrics in model is used to modify number VMs allocated for applications. Result of experiments shows that open queue network provides high efficiency.

KEYWORDS

Heterogeneous Infrastructure as a Service, Cloud Computing, Open Jackson Network

1. INTRODUCTION

Cloud computing services are provided flexibly according to user demand and access via the internet. It helpsto increase computing power and system management cost savings.[21, 25]. Cloud computing provides services to users through three basic models: infrastructure as a service (IaaS), including infrastructure and associated middleware - which are provided in the form of virtual machine (VM); Platform as a Service (PaaS) include APIs for developing applications on a specific technology platform; Software as a service (SaaS) - most of which are provided as a web-based and remote-access application. A data center that provides cloud computing services has a heterogeneous environment because it contains multiple generations of servers with different hardware configurations, especially the size and speed of the processor. These servers are added to the data center gradually and are provided to replace existing (or "the old one") machines already available [8, 15]. The heterogeneity of these server platforms will affect the performance of the data center. The fluctuation of the work load according to the needs of customers is frequent, and it is difficult to predict accurately in the environment of cloud computing. In addition, applications deployed on today's cloud are evolving towards service. Accordingly, an

DOI: 10.5121/ijcnc.2019.11104

application is deployed on a set of standalone services. This is different with monolithic applications including strictly integrated modules, applications based on service-oriented architecture that are well-suited for the infrastructure of the cloud [29].

The fluctuation of workload often occurs in the environment of cloud computing. It affects the quality even though the architecture is scalable - the ability of dynamically resource allocating and retrieving based on current workload requirements. As a matter of fact, Quality of Service (QoS) is a standard of service level agreement (SLA) established between the customer and the cloud service provider, which is one of the key issues. It is an important factor for the cloud provider [22]. To assess QoS for cloud computing, there are many studies which use the important system metrics such as average response time, average latency, average workload, refuse to serve[23]. These measurements can be analyzed and modeled based on queue theory. Applications deployed on the cloud are typically built using simple queue models such as a *G/G/c* queue, where *c* can be changed [1]. The model is used to estimate parameters such as the resources required for a given input job load or the average response time for the requirements. Then, this information is transferred to the predictor, controller or to solve the optimization problem. However, when the architecture of the application on cloud computing grows and becomes more complex, using a single queue model becomes more difficult. Therefore, the queue network model is used to create an application layer consisting of K application servers [24] or consider a queue for a server [20], or just a queue for each tier [3, 27]. Most current studies are rarely considered to the heterogeneity of cloud infrastructure services. For example, an application deployed in multiple cloud VMs with different generations with different CPU speeds and capabilities. Rather, studies often assume that each node operates at the same speed.

In this paper, we has been investigated and proposed a queue model of analyzing and evaluating performance measurements in a heterogeneous environment in order to meet customer demand for QoS. The main contributions of this paper can be summarized as follows:

- Proposed an open Jackson queue network model for multi-tier application systems which are deployed on heterogeneous VMs of IaaS cloud computing and formulas proposed to calculate the system performance metrics.
- The Proposed model that is evaluated through empirical simulation can be trusted.
- Based on the proposed model, we solve the auto-scaling problem for creating/removing VMs as in [30, 14]. Specifically, optimizing the performance utilization of a system based on a performance threshold of the VM that automatically adjusting the number of VMs is to ensure the minimum response time of the system.

The remainder presents the relevant studies in section II. Multi-tier application on cloud computing are presented in Section III. Part IV presents Jackson's open network model for the cloud infrastructure service. Experimental evaluation of the model is presented in Section V. Section VI presents the conclusions and research directions.

2. RELATED WORK

The cloud-based performance computing service model that focuses on QoS measurements is response time, throughput and network utilization, which has been extensively studied in [19, 25-27]. Zhang *et al.*[27] used a regression-based estimation to estimate the CPU demand for customer transactions. In [25], the response time distribution of a cloud system is modeled on a classical M/M/m open network assuming an exponential density function for time intervals and service time To determine the optimal service level and the relationship between the maximum the number of jobs and the minimum number of resources, namely VMs. Response time is calculated both the waiting time in the queue and the service time.

In [19], Slothouber *et al.* studied the parameters that affect response time when web servers and network bottlenecks. A single queue is not enough to model a complex system such as a web server system, so the system response time can be estimated using the Jackson network model. In [26], a cloud center was modeled as a $GIX/M/S/\mathrm{N}$ queue model that identifies cloud service performance related to error recovery. Network nodes that form the cloud architecture can be analyzed independently as they form the Jackson open network. Internal connectivity and behavior between queues are law-defined by Burke's theorem [4] and Jackson [10, 11]. Buke states that they can connect multiple nodes together with a queuing network and still avoid interpreting each node in the network when the arrival time and service time are modeled in terms of exponential density. In addition, Jackson [10, 11] focuses on the computation of total speed to average, we have to calculate the total time coming from outside the system plus the time coming from all the nodes inside. As a result, we can connect different processing nodes to cloud architecture designs over time as responses such as QoS performance metrics. In [23], Based on queue theory and the Jackson open network, a combination of M/M/1 and M/M/m queues was presented to model the platform for QoS requirements.

Information necessary for the queue model, such as input workload (number of requests, transactions) or service time can be obtained by online monitoring [20]. In addition, Nah *et al.*[16] state that the time available for users to skip downloading a web page may have different scenarios and contexts. This research suggests that most users are willing to wait only two seconds for simple web-based queries. Thus, if we apply this result to cloud computing, a decision may be needed to design the cloud with a good and predictable response time to evaluate QoS parameters. The authors in [2] show how many VMs can share CPUs and main memory very efficiently in cloud computing, but networks and file systems are often unclear. Therefore, the author designing the web application architecture will consider the separation between the processing server and the data server. In [12], authors used the $M/G/m/m$ + r queue system to evaluate a geodatabase and obtained a full probability distribution of response time, task number in the system and other important performance measurements. Average waiting time varies between heterogeneous services and uniform services under the same conditions in the considered system. In [9], the authors present a M/M/m queuing system and propose an optimized model of optimization, function and strategy to optimize the performance of services in the center of the cloud.

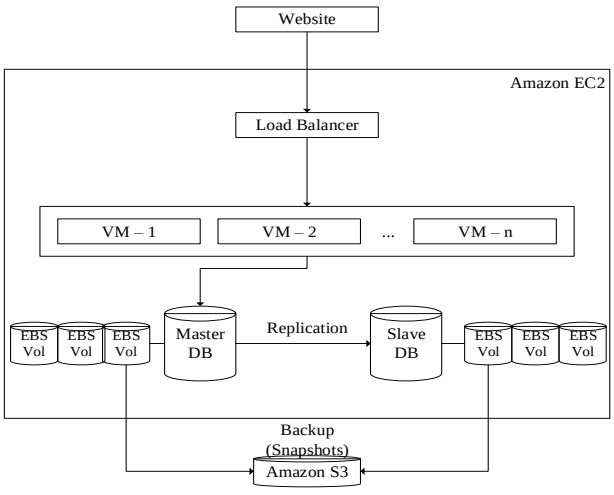

Figure 1. Example of a web application with a three-tier architecture

In addition, queue theory has been used to analyze or optimize factors such as allocation of processing capacity, load distribution, and profit control. In [5], the *M/M/m* queue model,

considering factors such as the service requirements of a service and the configuration of a multiserver system, is used to optimize the configuration. Multiserver image and maximize profitability in the cloud. In [7], the authors present a *M/M/r/k* queuing system to model a server cluster in the cloud provided with finite power to optimize returns. Another study presents a queue model for a heterogeneous multi-core server cluster with different sizes and speeds to optimize the allocation of processing power and load distribution in an environment.[6].

Although there have been many studies analyzing the efficiency of a cloud computing center based on queue models, the factors include flexible cloud architecture, the heterogeneous infrastructure of a cloud computing center is considered. Each processing node has different speeds and different processing times. Different task roles often have different probabilities.

3. SYSTEMMODEL

3.1. Application model

An Application has an architecturein which layers are sequentially connected. For each layer, request or processing results on the previous layeris an input of next layer for further processing and return final results to the user.

A typical multi-level application architecture consists of three layers: front-endlayer, logic layer, and database layer. The database layer is unable to dynamically adjust and often ignore the automatic adjustment feature.

We consider an elastic application deployed on a group of VMs (Figure 1). VMs may have the same or different resources assigned (eg 1GB of memory, 2 CPUs, ...), but each VM has a unique number of formats (possibly its IP address). Requests received by the load balancer may come from the actual end-user or from another application. We will assume that the execution time of a request can change between milliseconds and minutes.

Load balancers will receive all incoming requests and forward them to one of the servers in the business process layer (logic layer). It can be done in different ways. Assume that the load balancer has updated information about the VMs that are being used (the active VMs): it will immediately stop sending requests to the VMs that have been removed, and it will start sending Load jobs to new VMs added. It also assumes that each request will be assigned to a single VM, which will run it until the completion of the task associated with it. Some load balancing policies may be used, for example: random policy, round robin, or least connection. In case of heterogeneous VM clusters, workload coordination must be proportional to the processing power of the VMs.

3.2. Infrastructure model

In this section, the model of cloud infrastructure service is presented. Virtualization resources are provided by the IaaS cloud provider as a VM. VMs differ in terms of CPU, memory, storage, network, and availability at different rates. An application on the cloud can be deployed on a VM cluster, where the VMs may not be synchronized. VMs are classified according to their specific processing roles such as compute processing, memory, storage, GPU, and so on. Customers can choose the appropriate VMs for their applications. For example, VMs that handle CPUs for web servers and memory optimizers for high-performance database operations.

In order to ensure QoS for customers, service providers often target certain QoS objectives, such as response time, throughput, latency, execution time, and transaction time. Quality control will create reputable services that bring customer satisfaction and thereby increase the number of users and revenue.

Figure 2 depicts the IaaS cloud architecture, which includes the Input Controller, responsible for distributing requests to the VM cluster running the application at client request. The Infrastructure Manager is responsible for supplying resources to the VM clusters according to the needs of the customer. The Performance Manager is responsible for decision-making on input control and dynamic resource allocation policies, which makes virtual machine creation/destroy. In order to implement the automatic adjustment of resources in the cloud computing system, we use the MAPE loop to automatically adjust, to monitor, to analysis, to plan and to execute tasks. The monitoring module, a service typically provided by cloud service providers such as Amazon CloudWatch and Google Cloud Monitoring, continuously monitors the performance of hosted applications. Information received from the monitoring module is used during the analysis and planning phase to estimate future resource requirements and to plan for an appropriate resource adjustment action. Input parameters are information on job load, QoS measurements, and thresholds.

Each VM cluster runs an application of Load Balancer that is responsible for distributing requests to different VMs. VM clusters may be heterogeneous, so we use weighted load balancing strategies to distribute requests to different VMs that correspond to their capabilities. Each VM runs a local agent that queries the current performance parameters, such as CPU and memory load.

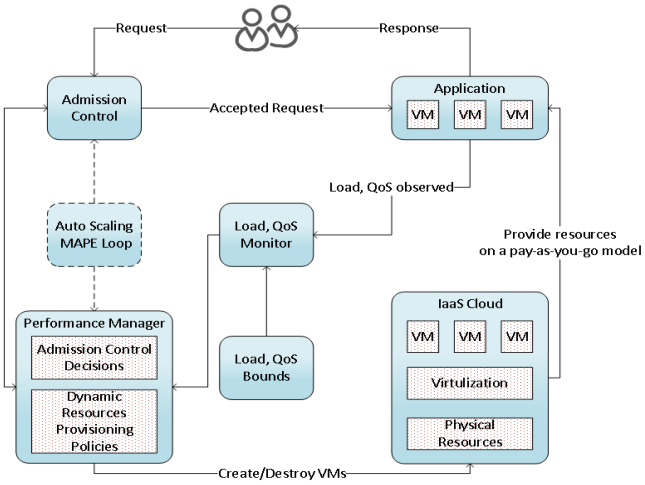

Figure 2. Infrastructure service system architecture

3.3. Queueing network modelfor applications

As the architecture of the applications on cloud computing is growing and becoming more complex, we proposed a model for a cloud computing using open Jackson queue network (Figure 3).In Figure 1, the application on the cloud has multiple layers that assume different roles in the application as well as the need to use different types of VMs at each level. Specifically, in Figure 1, each application server cluster for the end user includes a load balancer, a business processing layer, and a database processor.In this paper, we consider Load Balancer as a VM that serves as the coordinating role required for business process layers. The business process layer consists of many heterogeneous VMs that can be processed in parallel and capable of processing differently depending on the configuration of each VM. Once processed at the business processing layer, the request can be routed through a data processor, then the processing results are passed to the user.

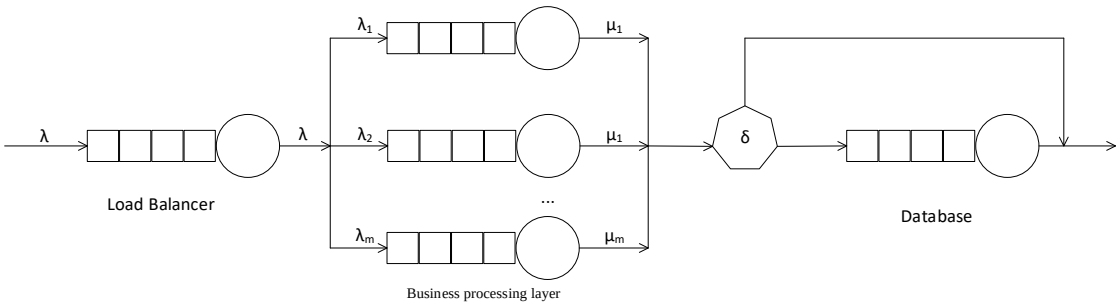

Figure 3: Open queue network model for applications

The load balancer receives requests coming from the end user and then processes and dispatches the requests to the business processing VMs based on their configuration. The policy allocates resources to request in the form of space-shared mean at a time the VM only serves a request. Coordination of requests from load balancers to business processors VMs is calculated according to the Round Robin load balancing policy. Depending on the weight of each VM in the business process layer. The speed of VM service depends on the VM's processing power. In this study, we focus on heterogeneous VMs, whereby each VM has c_i processing cores, and the execution speed of each core is s (GIPS).

Service rate of VM is calculated as follows:

$$\mu_i = \frac{s * c_i}{\bar{Z}}, \tag{1}$$

In that, $\bar{Z}$ is the average number of orders of a request executed at the VM.

Table 1. Round Robin Algorithm

```
Input: Requests {Requests list}
Output: Requests distribution → VMs
map ← empty dictionary
cur_p {The first of VMin VMs list}
size ← {No of VMs}
for ∀vm ∈Requests do
 cur_p = cur_p%size
 if vm can be accepted cur_p then
        map ← p
        cur_p ← VM next in Requests
 else
end for
return map
```

In the business processing layer, the VMs processed in the network may have dependency loads, which means that the request rate for leaving the service is a function of the number of requests currently in the VM that $\mu_i(k)$. In the database storage, the probability δ that a request from the VM handles business access to the VM storage database. When we perform a web-based modeling, not all requests will require access to the database server. However, notice that this probability is often relatively high.

We consider the following assumptions, which are also used in [17, 18, 23]:(1) Requests coming from outside of cloud computing system on a node *i* follow the Poisson process; (2) The service time at each node *i* in the cloud system is independent and complies with the exponential

distribution and is served according to the FCFS (First-Come First-Serve) principle; (3) Probability when a request is completed at a node i can be passed to the node j $(i \neq j)$ $r_{ij} \geq 0$ independent of the system state or will leave the system and not return to the probability $r_{i0} = 1 - \sum_{j=1}^{N} r_{ij}$, where N is the number of network nodes in the queue network.

These properties satisfy the requirements of an open queue network Jackson [4, 10]. Thus, we can see that the cloud computing system is an open Jackson network queue with N network nodes, each corresponding network node is a VM.

4. MODEL OF MULTI-LAYER APPLICATIONS ON CLOUD COMPUTING

4.1. Analyze the proposed model

As mentioned above, in this section, we will model a multi-layered application cloud system that is an open Jackson network queue consisting of N nodes, each corresponding network node is a VM and each VM is modeled as one M/M/1 queue.

The load balancer receives requests from outside users and then processes and dispatches requests to the business process layer VMs based on their configuration. Requires an external load balancer of Poisson distribution with the parameter γ (number of requests per second). The interval between times to average is $1/\gamma$. We set the average resource utilization factor for each ithVM (symbol ρ_i) with the threshold in the range $[\alpha_l, \alpha_h]$, depending on the different VMs with the service rate μ_i. This describes the rate at which the servers at each node handled the $1/\mu_i$ request as the average service time.

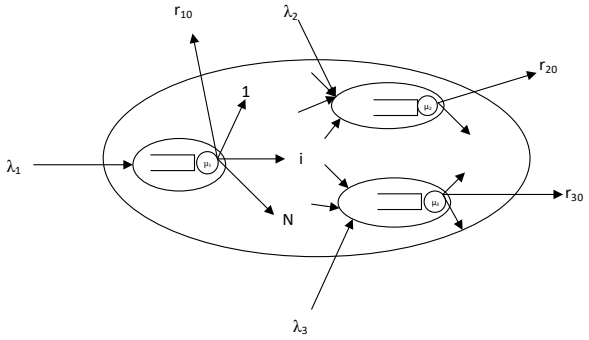

Figure 4. General open network queue

The nodes can be linked together in either serial or parallel mode. Assume the length of each queue is infinite and the queue's serving principle is FCFS. At each node of the system is a M/M/1 queue and the generation of HTTP requests by the user is a randomized process. Let r_{ij} be the probability of transferring the state after the request has been processed at node i which will move to node j. Yes, the total transfer probability in the open queue network is 1 (here: r_{i0} is the probability of moving out of the network from the ith node):

$$r_{i0} = 1 - \sum_{j=1, j\neq i}^{N} r_{ij}, i \in [1, N], \quad (2)$$

Let λ_i be the arrival rate from the outside into the ith queue (node) and Λ_i is the sum of the arrival rates of the ith node queue (including both incoming and outgoing rates from inner nodes network), we have:

$$\Lambda_i = \lambda_i + \sum_{j=1, j\neq i}^{N} \Lambda_j r_{ji}, i \in [1, N], \quad (3)$$

Thus, the average resource utilization of i[th] VM is:

$$\rho_i = \frac{\Lambda_i}{\mu_i}, \quad (4)$$

Let$\Lambda = [\Lambda_1, \Lambda_2, \dots, \Lambda_N]^T$ is the speed vector to the queues of nodes (including the incoming and outgoing rate from the nodes in the network). Let $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_N]^T$be the incoming vector velocity from the outside into the nodes, then the formula (3) can be rewritten as:

$$\Lambda = \lambda + R^T \Lambda, \quad (5)$$

where R is the state transition matrix.

$$R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1N} \\ r_{21} & r_{12} & \cdots & r_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ r_{N1} & r_{N1} & \cdots & r_{NN} \end{pmatrix} \quad (6)$$

Since the number of requests in nodes may be different, let$X_i(t)$ as the random variable that determines the number of requests in the i[th] VM ($i = 1,2, \dots, N$) at time t. The state of the system at time t is denoted by: $X(t) = (X_1(t), X_2(t), \dots, X_N(t))$. Then, in steady state, we wish to determine their probability distribution over the long term as follows: $p_{\mathrm{n}_1,\mathrm{n}_2,\dots,\mathrm{n}_N} \equiv P(\mathrm{n}_1, \mathrm{n}_2, \dots, \mathrm{n}_N) = P\{X_1(t) = n_1, X_2(t) = n_2, \dots, X_N(t) = n_N\}$. Since the VMs are independent of each other and $P(n_i)$ is the boundary probability that $X_i(t) = n_i$. From the joint probability, we can compute the boundary probability of a particular number of requests at one VM.

An open Jackson network is considered as a continuous time Markov chain with state vector

$$\bar{n} = (\mathrm{n}_1, \mathrm{n}_2, \dots, \mathrm{n}_N), \quad (7)$$

where n_i is the number of requests that are available at VM i. We can use equilibrium equation based on the Markov system.

In Table 1, we have $\bar{n}$representing the steady state of the system; When there is a request to VM i, the system from state $\bar{n}$ to state $\bar{n}; i^+$; whereas there is a request to leave VM i, the system will move from state $\bar{n}$ to state $\bar{n}; i^-$; or system state from $\bar{n}$ to state$\bar{n}; i^+ j^-$ when a request from VM j passes to VM i.

Table 2. States description table

States	Notation
$\mathrm{n}_1, \dots, \mathrm{n}_{i-1}, \mathrm{n}_i, \mathrm{n}_{i+1}, \dots, \mathrm{n}_N$	$\bar{n}$
$\mathrm{n}_1, \dots, \mathrm{n}_{i-1}, \mathrm{n}_i + 1, \mathrm{n}_{i+1}, \dots, \mathrm{n}_N$	$\bar{n}; i^+$
$\mathrm{n}_1, \dots, \mathrm{n}_{i-1}, \mathrm{n}_i - 1, \mathrm{n}_{i+1}, \dots, \mathrm{n}_N$	$\bar{n}; i^-$
$\mathrm{n}_1, \dots, \mathrm{n}_{i-1}, \mathrm{n}_i + 1, \mathrm{n}_{i+1}, \dots, \mathrm{n}_{j-1}, \mathrm{n}_j - 1, \mathrm{n}_{j+1}, \dots, \mathrm{n}_N$	$\bar{n}; i^+ j^-$

This Markov chain has the probability of transmitting the notation $p_{\cdot,\cdot}$ as follows:

$$p_{\bar{n};i^+} = \lambda_i$$

$$p_{\bar{n};i^-} = \mu_i r_{i0}$$

$$p_{\bar{n};i^+j^-} = \mu_j r_{ji}$$

Using the principle of balancing state flow into state $\bar{n}$ with flow out of state $\bar{n}$, assuming $n_i \geq 1$ at every processes VM, we have:

$$\sum_{i=1}^{N} \lambda_i \, p_{\bar{n};i^-} + \sum_{j=1}^{N} \sum_{\substack{i=1 \\ i \neq j}}^{N} \mu_i r_{ij} p_{\bar{n};i^+j^-} + \sum_{i=1}^{N} \mu_i r_{i0} p_{\bar{n};i^+} = \sum_{i=1}^{N} \mu_i (1 - r_{ii}) \, p_{\bar{n}} + \sum_{i=1}^{N} \lambda_i \, p_{\bar{n}}. \quad (8)$$

According to Jackson's theorem [10, 11] providing the general distribution for all VMs, the steady state solution for (8) is:

$$p_{\bar{n}} = P\{X_1(t) = n_1, X_2(t) = n_2, \dots, X_N(t) = n_N\} = P(X_1(t) = n_1) P(X_2(t) = n_2) \dots P(X_N(t) = n_N) \quad (9)$$

$$\text{where } P(X_i(t) = n_i) = (1 - \rho_i)\rho_i^{n_i}, with \rho_i < 1.$$

Thus,

$$p_{\bar{n}} = \sum_{i=1}^{N} (1 - \rho_i)\rho_i^{n_i}$$

The average number of requests L_i at the i[th] VM for the *M/M/1* queue with the total arrival rate to Λ_i:

$$L_i = \frac{\rho_i}{1 - \rho_i}, i = 1,2, \dots, N. \quad (10)$$

The total number of average requests for an entire network is calculated as follows:

$$L = L_1 + L_2 + \cdots + L_N = \sum_{i=1}^{N} \frac{\rho_i}{1 - \rho_i}. \quad (11)$$

Average waiting time of requests in the network with Little's Law [10, 11]:

$$W = \frac{L}{\beta} \quad (12)$$

where $\beta = \sum_{i=1}^{N} \lambda_i$ is the total arrival rate from the outside. All requests from outside must go through the load balancer, thus $\beta = \gamma$. The average response time of a request in the network at each i[th] VM, $(i = 1,2, \dots, N)$ is calculated by the following formula [10, 11]:

$$W_i = \frac{L_i}{\Lambda_i} = \frac{1}{\mu_i(1 - \rho_i)}, \quad (13)$$

WhereΛ_i is calculated by formula (3) and$W \neq W_1 + W_2 + \cdots + W_N$.

4.2. An example for proposed model

Figure 5 is an example of an open Jackson queue model for a three-layered application network with four network nodes. The node 1 is VM_1 represents the required load balancer coming from the outside into γ, the business processing layer consists of two nodes: VM_2 and VM_3, the required speed are transitioned from VM_1 to the corresponding transition probabilities $r_{12} = 0.4$ and $r_{13} = 0.6$. Once a request has been completed at the business process layer, it is possible to switch to access the database store VM_3 with probability $r_{34} = r_{24} = \delta = 0.3$ or no ($r_{30} = r_{20} = 0.7$). The results are then feedback to the customer. The service speed at each i th VM in exponential distribution has the parameter μ_i.

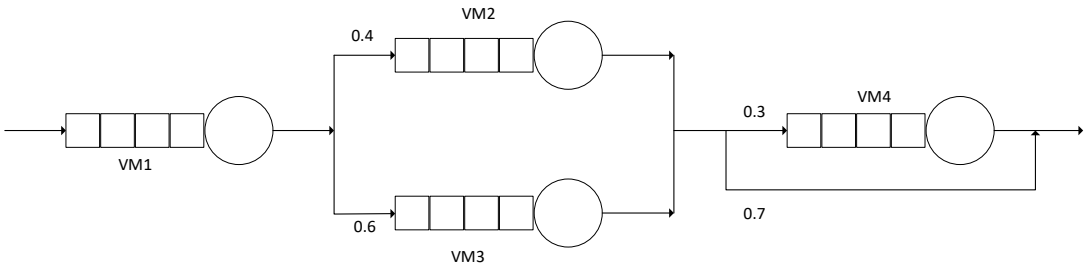

Figure 5. For example an open queue network model for multilevel application on the cloud computing

Then, we have the following transition matrix:

$$R = \begin{pmatrix} 0 & 0.4 & 0.6 & 0 \\ 0 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (14)$$

where the probability of leaving the system at each $i, i \in [1,2,3,4]$ is: $r_{10} = 0, r_{30} = r_{20} = 0.7, r_{40} = 1$.

Arrival rate of load balancer is γ = 20 requests/sec.Services time of each VMs is $\frac{1}{\mu_1} = 0.03\ sec, \frac{1}{\mu_2} = 0.06\ sec, \frac{1}{\mu_3} = 0.05$ sec $and\ \frac{1}{\mu_4} = 0.04\ sec.$

Assuming: Calculated steady state probability of state $(n_1, n_2, n_3, n_4) = (3, 2, 4, 1)$.

Step 1: compute the arrival rates for each node from the traffic equations (4). We have:

$$\Lambda_1 = \gamma = 20;\ \Lambda_2 = \Lambda_1.\mathrm{r}_{12} = 8; \Lambda_3 = \Lambda_1.\mathrm{r}_{13} = 12; \Lambda_2 = \Lambda_2.\mathrm{r}_{24} + \Lambda_3.\mathrm{r}_{34} = 6$$

Step2: Compute the state probabilities for each node

Use utilization: $\rho_i = \frac{\Lambda_i}{\mu_i}$to get the service demands for each node: $\rho_1 = 0.6, \rho_2 = 0.48, \rho_3 = 0.6,$ and $\rho_4 = 0.24$.
Use the equation $P_i(n_\mathrm{i}) = P(X_\mathrm{i}(t) = n_\mathrm{i}) = (1 - \rho_i)\rho_i^{n_i}$ to compute the probability of having$n_\mathrm{i}$requests in each M/M/1 queue. We have: $P_1(3) = 0.09, P_2(2) = 0.12, P_3(4) = 0.05, and P_4(1) = 0.18$.

Step3: Compute the steady state probability P(3, 2, 4,1). According to equation (9), we have $P(3, 2, 4, 1) = P_1(3)P_2(2)P_3(4)P_4(1)$=0.0000972.

Some important performance measures for this open Jackson network. Compute the mean number of requests in each queue with $L_i = \frac{\rho_i}{1-\rho_i}$, we have: $L_1 = 1.5, L_2 = 0.92, L_3 = 1.5, and L_4 = 0.32.$

Compute the mean response time of each queue with $W_i = \frac{L_i}{\Lambda_i}$, we have: $W_1 = 0.075, W_2 = 0.115, W_3 = 0.125, and W_4 = 0.053.$

Compute the mean overall response time following equation (12):

$$W = \frac{L}{\gamma} = \frac{1}{\gamma}\sum_{i=1}^{4} L_i = 0.212 \neq W_1 + W_2 + W_3 + W_4 = 0.368$$

4.3. Optimized model uses VM in clusters

In this section, we present the application of the QoS parameter used for the model to solve the optimization problem as in [5-7]. Assume that an application deployed on the cloud architecture is shown in Figure 6, consisting of three layers: load balancing, business processing layer, and database layer. The majority of cloud deployments have a load balancer and database engine so in this context, we only see the VM model for the business process layer.

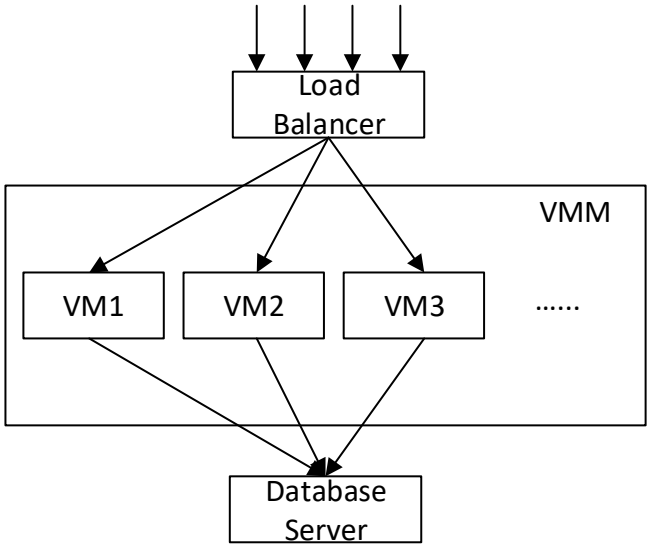

Figure 6. The architecture of the application

Specifically, based on network queue evaluation parameters, it is possible to optimize system performance based on the performance thresholds of VMs and to have at least one VM enabled at the business process layer. To ensure that the system is always ready to serve, we can determine the number of VMs processed at the business layer with minimal response time.

The parameters of the problem are as follows:

- ρ_i is the average resource utilization level of the ith VM,
- r_i is the probability that the i th VM is selected to handle incoming requests,
- W_i is the average waiting time of a request in the network at a i th VM,
- z_i is the binary variable that decides to turn on VM i ($z_i = 1$), in turn decides to turn off VM on i ($z_i = 0$).

The objective function is defined as follows:

$$\min_{\{\rho_i, r_i, W_i\}} f: \sum_{i=1}^{N} z_i r_i W_i, \quad (15)$$

Subject to:

(1) $\rho_i = \begin{cases} 0, z_i = 0 \\ \epsilon[\alpha_l, \alpha_h], z_i = 1 \end{cases}$,

(2) $z_i \in \{0,1\} với\ 1 \le i \le N$,

(3) $\sum_i^N r_i = 1\ với r_i \in [0,1]$,

(4) $\sum_i^N z_i \ge 1$.

In the optimization problem we have the variable z_i that determines creating/removing of the VM so this is a 0/1 Knapsack problem [28]. We use the met heuristic algorithm in particular as the optimal PSO algorithm [13] to solve the optimal tuning (creating/removing) of the target function f and its constraints on response time average response.

Table 3. VM Auto Scaling

```
// Initialize the forum
P = Partice_Initialization();
For i = 0 to it_max
For each p in P do
//Value of objective function as equal (15)
fp = f(p)
if fp ≥f(pBest)
pBest = p;
end
end
// Update global good value
 gBest = best p in P;
 // Update velocity and position of the instrument
For each p in P do
v = v + c1*rand*(pBeset – p)
+ c2*rand*(gBeset – p);
p = p + v;
 end
end
```

5. EXPERIMENT

5.1. System setup

In this section, we conducted an experiment to evaluate performance measured in Java language on a laptop with an Intel® Core ™ i7-7500U CPU configuration @ 2.70GHz, 8GB Ram, 1 TB HDD. and run the Windows operating system. 10. Experimental review of the cloud infrastructure service system consists of two VM clusters that serve requests from web service clients. Each VM cluster has 7 VMs (IDs 1 through 7). In each cluster, there is a VM that performs the work of the load balancer (VM_1), 1 VM takes on the database storage (VM_7) and the rest takes care of the workload. Business process layer (VM_i, with i = 2,3, ..., 6). The processing power of each VM is calculated based on GIPS. Two VM clusters have different configurations as shown in Table 3.2, in both clusters on the load balancer and the database storage server have the same capacity. In VMC#1 cluster, business processors have the ability to be consistent. In VMC#2 cluster the business processors are not homogeneous.

Table 4. Configuration of VM clusters

VM	VM Processing Capability (GIPS)	
	VMC#1	VMC#2
1	4	4
2	2	2
3	2	4
4	2	6
5	2	8
6	2	10
7	4	4

5.2. Experimental results

Experiment 1 evaluates the parameters in the model, for the number of requests to 1000, λ = {158,161,164,167,170,173,176,179,181,185, 188} the load balancer's processing rate is $\mu_{LB} = \frac{\lambda}{\rho_{LB}}$. The probability of passing to the business processing VM is calculated using the formula r_{ij}. After 30 experimental runs, we calculated the average waiting time $\bar{W}$, the average response time $\bar{T}$. We then calculated 95% confidence intervals for empirical data and computed data from the model.

Experiment 1 shows that when we adjust the required speed increases from 158 to 188, the average waiting time is normal but the amplitude is not large. For the VMC#1 cluster the machines have identical CPU configurations, while the VMC#2 cluster has a heterogeneous CPU configuration and processing speed is greater than VMC#1 so that the average response time VMC#2 is much smaller than VMC#1 (Figure 7b).

In Experiment 2 we use the data set of Table 2 with λ = 185, which is executed 30 times and we calculate the average waiting time $\bar{W}$. We then calculated 95% confidence intervals for empirical data and computed data from the model.

The results in Figure 8 show that the average response times of VM_1 and VM7 are very small in both VMC#1 and VMC#2 clusters. VM_1 takes care of load balancing, fast processing time. For VM_7 taking on the task of storing the database, with probability to low δ = 0.2. The VM_i (i = 2,3,..., 6) in the business process layer has an average response time in two different clusters. Accordingly, the average response time of each machine in VMC#1 cluster is not too large. While VMC#2 cluster tends to dwindle from VM_2 to VM_6, as the VM_2's VM2 configuration to VM6 in VMC cluster 2 increases, the processing time on these machines decreases dramatically.

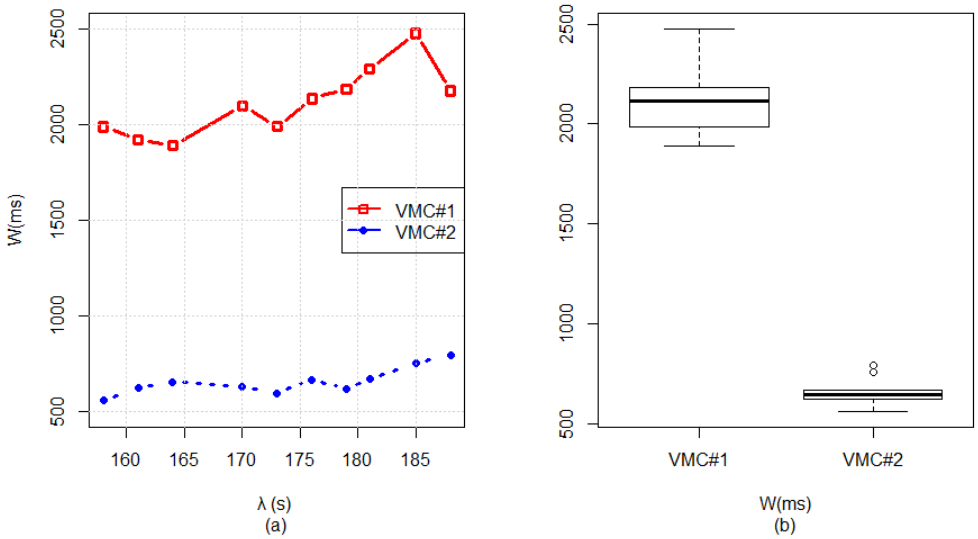

Figure 7. Average waiting time of the VM cluster

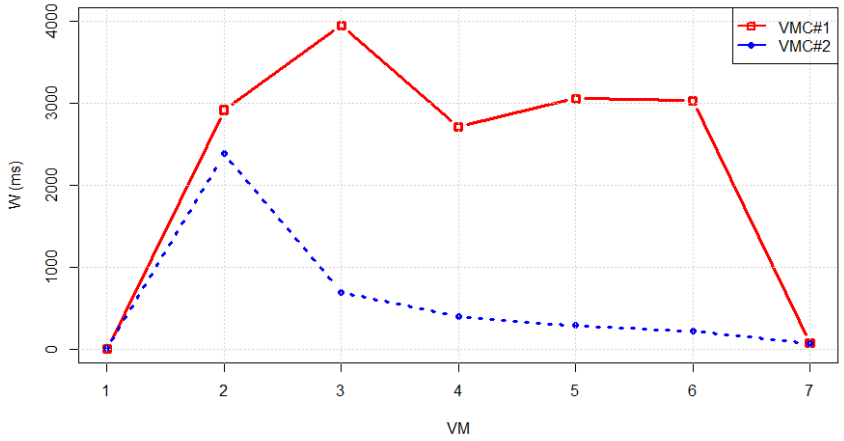

Figure 8. Average waiting time of VMs

Experiment 3 is to analyse the relationship of parameters in the model. We use the two VM clusters as shown in Table 2 at average speed$\lambda = 185$, adjusting the resource utilization $\rho_i$of each VM in Table 2, and then analysing the average system response time for each VM in clusters. We adjusted the resource utilization of the VMs in the application's business layer from 0.71 to 0.79.

Experiment 4 for the evaluation of the performance max model values using VM at the business processing from VM2 to VM6. We adjust the performance of the hosts with the distribution with mean = 0.5 (Figure 9). Follow the formula (14), we use the metaheuristic algorithm for the best fighter algorithm for PSO to resolve to find the best VM (enable/disable) history of f and find force it about the average response time. Figure 9 shows$\rho \in (0,1)$ of the VM in the logical level of an application with sample get template t = 50. Confused the selected VM processing with same. Then, we got at the time of the way of the different VM account.

Table 5. 95% confidence interval of average waiting time of each VM in each cluster

Average waiting time $\overline{W}$					
VM	ρ_i	**VMC**	**Average**	**Lower**	**Upper**
1	0.8	VMC#1	6.39069	6.20901	6.57238
		VMC#2	6.37414	6.16965	6.57863
2	0.71	VMC#1	3987.14	3126.29	4848.00
		VMC#2	4115.76	3241.04	4990.49
3	0.73	VMC#1	5256.16	4211.63	6300.7
		VMC#2	803.147	701.051	905.243
4	0.75	VMC#1	3430.45	2538.11	4322.80
		VMC#2	434.463	401.364	467.562
5	0.77	VMC#1	4127.60	3154.68	5100.51
		VMC#2	277.476	249.991	304.960
6	0.79	VMC#1	3756.96	2980.27	4533.66
		VMC#2	236.522	219.641	253.403
7	0.81	VMC#1	73.9026	68.5611	79.2442
		VMC#2	69.6167	65.9044	73.3289

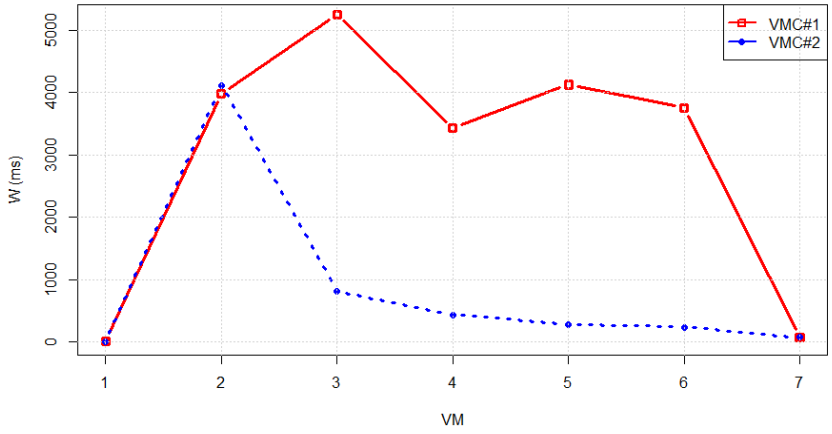

Figure 9. Average expected time of VM

Figure 11 shows the decision to enable/disable the VM based on the resource usage of each VM at each specific time point. For example, at t = 4 use level VM#2,4,5,6 low VM#3 high algorithm decided to turn off the VM#2,4,5,6 and turn on VM#3.

6. CONCLUSIONS

Ensuring QoS in the cloud is an important issue. Accordingly, the performance analysis of heterogeneous datacenter is an important aspect for both cloud service providers and cloud service customers. Based on the complexity and heterogeneity of the deployment application on cloud computing, we propose a model of a closed-loop opportunity-based cloud computing system that evaluates the performance of a cloud-based system through measures such as average latency, average resource utilization factor of VMs, problem of adjusting VMs in a cluster when there is load variation in workload to implement automatic adjustment mechanism in cluster.

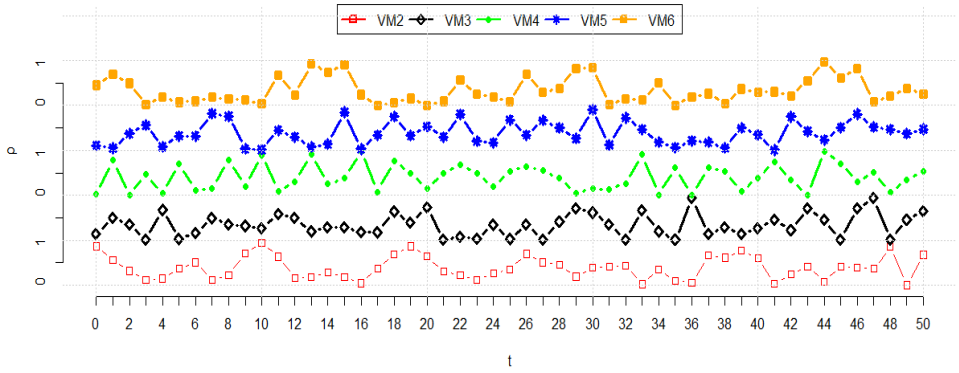

Figure 10. ρ usage level of the VM

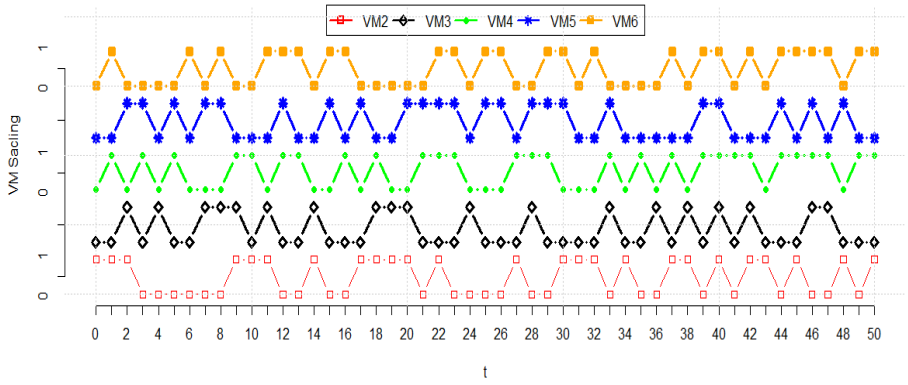

Figure 11. Adjust VM on/off over time

The results of the evaluation of the proposed model have shown its optimization. Through empirical simulation, the average waiting time, the average resource utilization coefficient ρ_i and the problem of adjusting the VMs in each cluster show up the effect of the non-uniformity. Most of the performance is great. In the future, we use this model in proposing an automatic adjustment mechanism in the cloud computing system.

REFERENCES

[1]. Ali-Eldin, Ahmed, Tordsson, Johan, and Elmroth, Erik (2012), An adaptive hybrid elasticity controller for cloud infrastructures, Network Operations and Management Symposium (NOMS), 2012 IEEE, IEEE, pp. 204-212.

[2]. Armbrust, Michael, et al. (2010), "A view of cloud computing", Communications of the ACM. 53(4), pp. 50-58.

[3]. Bai, Wei-Hua, et al. (2015), "Performance analysis of heterogeneous data centers in cloud computing using a complex queuing model", Mathematical Problems in Engineering. 2015.

[4]. Burke, Paul J (1956), "The output of a queuing system", Operations research. 4(6), pp. 699-704.

[5]. Cao, Junwei, et al. (2013), "Optimal multiserver configuration for profit maximization in cloud computing", ieee transactions on parallel and distributed systems. 24(6), pp. 1087-1096.

[6]. Cao, Junwei, Li, Keqin, and Stojmenovic, Ivan (2014), "Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers", IEEE Transactions on Computers. 63(1), pp. 45-58.

[7]. Chiang, Yi-Ju and Ouyang, Yen-Chieh (2014), "Profit optimization in SLA-aware cloud services with a finite capacity queuing model", Mathematical Problems in Engineering. 2014.

[8]. Delimitrou, Christina and Kozyrakis, Christos (2013), Paragon: QoS-aware scheduling for heterogeneous datacenters, ACM SIGPLAN Notices, ACM, pp. 77-88.

[9]. Guo, Lizheng, et al. (2014), "Dynamic performance optimization for cloud computing using m/m/m queueing system", Journal of applied mathematics. 2014.

[10]. Jackson, James R (1957), "Networks of waiting lines", Operations research. 5(4), pp. 518-521.

[11]. Jackson, James R (1963), "Jobshop-Like Queueing Systems. Mgmt. Sci. 10, 131-142", Jackson13110Mgmt. Sci.

[12]. Khazaei, Hamzeh, Misic, Jelena, and Misic, Vojislav B (2012), "Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems", IEEE Transactions on parallel and distributed systems. 23(5), pp. 936-943.

[13]. Liang, Yanbing, et al. (2010), Optimizing particle swarm optimization to solve knapsack problem, International Conference on Information Computing and Applications, Springer, pp. 437-443.

[14]. Mao, Ming, Li, Jie, and Humphrey, Marty (2010), Cloud auto-scaling with deadline and budget constraints, Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on, IEEE, pp. 41-48.

[15]. Mars, Jason, Tang, Lingjia, and Hundt, Robert (2011), "Heterogeneity in "homogeneous" warehouse-scale computers: A performance opportunity", IEEE Computer Architecture Letters. 10(2), pp. 29-32.

[16]. Nah, Fiona Fui-Hoon (2004), "A study on tolerable waiting time: how long are web users willing to wait?", Behaviour & Information Technology. 23(3), pp. 153-163.

[17]. Salah, Khaled (2013), "A queuing model to achieve proper elasticity for cloud cluster jobs", International Journal of Cloud Computing. 1, pp. 53-64.

[18]. Salah, Khaled, Elbadawi, Khalid, and Boutaba, Raouf (2016), "An analytical model for estimating cloud resources of elastic services", Journal of Network and Systems Management. 24(2), pp. 285-308.

[19]. Slothouber, Louis P (1996), A model of web server performance, Proceedings of the 5th International World wide web Conference.

[20]. Urgaonkar, Bhuvan, et al. (2008), "Agile dynamic provisioning of multi-tier internet applications", ACM Transactions on Autonomous and Adaptive Systems (TAAS). 3(1), p. 1.

[21]. Vaquero, Luis M, et al. (2008), "A break in the clouds: towards a cloud definition", ACM SIGCOMM Computer Communication Review. 39(1), pp. 50-55.

[22]. Vecchiola, Christian, Pandey, Suraj, and Buyya, Rajkumar (2009), High-performance cloud computing: A view of scientific applications, Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on, IEEE, pp. 4-16.

[23]. Vilaplana, Jordi, et al. (2014), "A queuing theory model for cloud computing", The Journal of Supercomputing. 69(1), pp. 492-507.

[24]. Villela, Daniel, Pradhan, Prashant, and Rubenstein, Dan (2007), "Provisioning servers in the application tier for e-commerce systems", ACM Transactions on Internet Technology (TOIT). 7(1), p. 7.

[25]. Xiong, Kaiqi and Perros, Harry (2009), Service performance and analysis in cloud computing, Services-I, 2009 World Conference on, IEEE, pp. 693-700.

[26]. Yang, Bo, Tan, Feng, and Dai, Yuan-Shun (2013), "Performance evaluation of cloud service considering fault recovery", The Journal of Supercomputing. 65(1), pp. 426-444.

[27]. Zhang, Qi, Cherkasova, Ludmila, and Smirni, Evgenia (2007), A regression-based analytic model for dynamic resource provisioning of multi-tier applications, Autonomic Computing, 2007. ICAC'07. Fourth International Conference on, IEEE, pp. 27-27.

[28]. Qu, Chenhao (2016), "Auto-scaling and Deployment of Web Applications in Distributed Computing Clouds".

[29]. Sahni, Jyoti and Vidyarthi, Deo Prakash (2016), "Heterogeneity-aware adaptive auto-scaling heuristic for improved QoS and resource usage in cloud environments", Computing, pp. 1-31.

[30]. Sowjanya, T Sai, et al. (2011), "The queueing theory in cloud computing to reduce the waiting time".

AUTHORS

Chien Nguyen Khac, received his master degree in Computer Science from the University of Natural Sciences in HCM City in 2008. He is currently a lecturer at the University of the People's Police, and is doing a PhD candidate in Computer Engineering at the PTIT, Hanoi. His research interests: Auto-Scaling in cloud computing. Email: nkchienster@gmail.com.

Khiet Bui Thanh, is a PhD candidate at Computer Science, Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology. Research Interests: Cloud computing. Email: khietbt@tdmu.edu.vn

Hung HoDac, Born in 1991 in Binh Duong. Graduated from the Graduate School of Information Systems at the Ho Chi Minh City Post and Telecommunications Institute of Technology. Currently working at Faculty of Engineering and Technology, Thu Dau Mot University, Binh Duong.
Research: Game, Cloud Computing. Email: hunghd@tdmu.edu.vn

Son Nguyen Hong, received his B.Sc. in Computer Engineering from the University of Technology in HCM city, his M.Sc. and PhD inCommunicationEngineeringfromthePostandTelecommunicationInstituteofTechnologyHanoi.Hiscurrentresearchinterestsincludecommunicationengineering,networksecurity,computer engineering and cloud computing. Email: ngson@ptithcm.edu.vn

Vu Pham Tran, Currently Deputy Dean of Computer Science and Technique, Ho Chi Minh City University of Technology. Research: Intelligent Transport Systems (ITS), Big Data Analytics, Peer-to-Peer Computing Email: ptvu@hcmut.edu.vn

Hung Tran Cong,He received the master of engineering degree in telecommunications engineering course from postgraduate department Hanoi University of technology in Vietnam, 1998. He received Ph.D at Hanoi University of technology in Vietnam, 2004. His main research areas are B – ISDN performance parameters and measuring methods, QoS in high speed networks, MPLS. He is, currently, Associate Professor PhD. of Faculty of Information Technology II, Posts and Telecoms Institute of Technology in Ho Chi Minh, Vietnam. Email: conghung@ptithcm.edu.vn