Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 7519507, 13 pages
http://dx.doi.org/10.1155/2016/7519507

*Research Article*

# On Elasticity Measurement in Cloud Computing

**Wei Ai,[1] Kenli Li,[1] Shenglin Lan,[1] Fan Zhang,[2] Jing Mei,[1] Keqin Li,[1,3] and Rajkumar Buyya[4]**

[1]*College of Information Science and Engineering, Hunan University, Changsha, Hunan 410082, China*
[2]*IBM Massachusetts Lab, 550 King Street, Littleton, MA 01460, USA*
[3]*Department of Computer Science, State University of New York, New Paltz, NY 12561, USA*
[4]*Department of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia*

Correspondence should be addressed to Kenli Li; lkl@hnu.edu.cn

Received 21 January 2016; Accepted 8 May 2016

Academic Editor: Florin Pop



Elasticity is the foundation of cloud performance and can be considered as a great advantage and a key benefit of cloud computing. However, there is no clear, concise, and formal definition of elasticity measurement, and thus no effective approach to elasticity quantification has been developed so far. Existing work on elasticity lack of solid and technical way of defining elasticity measurement and definitions of elasticity metrics have not been accurate enough to capture the essence of elasticity measurement. In this paper, we present a new definition of elasticity measurement and propose a quantifying and measuring method using a continuous-time Markov chain (CTMC) model, which is easy to use for precise calculation of elasticity value of a cloud computing platform. Our numerical results demonstrate the basic parameters affecting elasticity as measured by the proposed measurement approach. Furthermore, our simulation and experimental results validate that the proposed measurement approach is not only correct but also robust and is effective in computing and comparing the elasticity of cloud platforms. Our research in this paper makes significant contribution to quantitative measurement of elasticity in cloud computing.

## 1. Introduction

*(1) Motivation*. As a subscription-oriented utility, cloud computing has gained growing attention in recent years in both research and industry and is widely considered as a promising way of managing and improving the utilization of data center resources and providing a wide range of computing services [1]. Virtualization is a key enabling technology of cloud computing [2]. System virtualization is able to provide abilities to access software and hardware resources from a virtual space and enables an execution platform to provide several concurrently usable and independent instances of virtual execution entities, often called virtual machines (VMs). A cloud computing platform relies on the virtualization technique to acquire more VMs to deal with workload surges or release VMs to avoid resource overprovisioning. Such a dynamic resource provision and management feature is called elasticity. For instance, when VMs do not use all the provided resources, they can be logically resized and be migrated from a group of active servers to other servers, while the idle servers can be switched to the low-power modes (sleep or hibernate) [3].

Elasticity is the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible [4]. By dynamically optimizing the total amount of acquired resources, elasticity is used for various purposes. From the perspective of service providers, elasticity ensures better use of computing resources and more energy savings [5] and allows multiple users to be served simultaneously. From a user's perspective, elasticity has been used to avoid inadequate provision of resources and degradation of system performance [6] and also achieve cost reduction [7]. Furthermore, elasticity can be used for other purposes, such as increasing the capacity of local resources [8, 9]. Hence, elasticity is the foundation of cloud performance and can be considered as a great advantage and a key benefit of cloud computing.

Elastic mechanisms have been explored recently by researchers from academia and commercial fields, and

tremendous efforts have been invested to enable cloud systems to behave in an elastic manner. However, there is no common and precise formula to calculate the elasticity value. Existing definitions of elasticity in the current research literature are all vague concepts and fail to capture the essence of elastic resource provisioning. These formulas of elasticity are not suitable for quantifying and measuring elasticity. Moreover, there is no systematic approach that has been proposed to quantify elastic behavior. Only quantitative elasticity value can produce better comparison between different cloud platforms. Therefore, the measurement of cloud elasticity should be further investigated. As far as we know, the current reported works are ineffective to cover all aspects of cloud elasticity evaluation and measurement. Therefore, we are motivated to develop a comprehensive model and an analytical method to measure cloud elasticity.

*(2) Our Contributions*. In this paper, we propose a clear and concise definition to compute elasticity value. In order to do that, an elasticity computing model is established by using a continuous-time Markov chain (CTMC). The proposed computing model can quantify, measure, and compare the elasticity of cloud platforms.

The major contributions of this paper are summarized as follows.

(i) First, we propose a new definition of elasticity in the context of virtual machine provisioning and a precise computational formula of elasticity value.

(ii) Second, we develop a technique of quantifying and measuring elasticity by using a continuous-time Markov chain (CTMC) model. We investigate the elastic calculation model intensively and completely. The model is not only an analytical method, but also an easy way to calculate the elasticity value of a cloud platform quantitatively.

(iii) Third, we examine and evaluate our proposed method through numerical data, simulations, and experiments. The numerical data demonstrate the basic parameters which affect elasticity in our analytical model. The simulation results validate the correctness of the proposed method. The experimental results on a real cloud computing platform further show the robustness of our model and method in predicting and computing cloud elasticity.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 describes the definition of cloud elasticity. Section 4 develops the computing model of cloud elasticity. Sections 5, 6, and 7 present simulation and numerical and experimental results, respectively. Section 8 concludes this paper.

## 2. Related Work

*2.1. Elasticity Definition and Measurement.* There has been some work on elasticity measurement of cloud computing. In [4], elasticity is described as the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible. In [10], elasticity is defined as the ability of customers to quickly request, receive, and later release as many resources as needed. In [11], elasticity is measured as the ability of a cloud to map a single user's request to different resources. In [12], elasticity is defined as dynamic variation in the use of computer resources to meet a varying workload. In [13], an elastic cloud application or process has three elasticity dimensions, that is, cost, quality, and resources, enabling it to increase and decrease its cost, quality, or available resources, as to accommodate specific requirements. Recently, in [14], elasticity is defined by using the expression $1/(\theta \times \mu)$, where $\theta$ denotes the average time to switch from an underprovisioning state to an elevated state and $\mu$ denotes the offset between the actual scaling and the autoscaling. Existing definitions of elasticity fail to capture the essence when elastic resource provisioning is performed with virtual machines, and the formulas of elasticity are not suitable for quantifying elasticity. For example, $\mu$ in the above expression is difficult to obtain when resource of a cloud is increasing or decreasing. In contrast, the definition proposed in our work reflects the essence of elasticity, and the calculation formula focuses on how to measure the elasticity value effectively.

There are many approaches to predicting elasticity, anticipating the system load behavior, and deciding when and how to scale in/out resources by using heuristics and mathematical/analytical techniques. In [4], the authors established an elasticity metric aiming to capture the key elasticity characteristics. In [15], the authors proposed *execution platforms* and *reconfiguration points* to reflect the proposed elasticity definition. In [5, 7, 16–18], the authors adopted predictive techniques to scale resources automatically. Although these techniques perform well in elasticity prediction, further measurement of elasticity is not covered. In [4], the authors just outlined an elasticity benchmarking approach focusing on special requirements on workload design and implementation. In [15], the authors used thread pools as a kind of elastic resource of the Java virtual machine and presented preliminary results of running a novel elasticity benchmark which reveals the elastic behavior of the thread pool resource. These studies mainly present initial research. In most elasticity work, different elasticity benchmark programs are expected to execute on different systems over varying data sizes and reflect their potential elasticity, but they can only get a macroscopic view of elasticity analysis rather than the calculation of the elasticity value. In contrast, our work performs in-depth research focusing on the measurement of elasticity value.

*2.2. Analytical Modeling.* Continuous-time Markov chain (CTMC) models have been used for modeling various random phenomena occurring in queuing theory, genetics, demography, epidemiology, and competing populations [19]. CTMC has been applied in a lot of studies to adjust resource allocation in cloud computing. Khazaei et al. proposed an analytical performance model that addresses the complexity of cloud data centers by distinct stochastic submodels using

CTMC [20]. Ghosh et al. proposed a performance model that quantifies power performance trade-offs by interacting stochastic submodels approach using CTMC [21]. Pacheco-Sanchez et al. proposed an analytical performance model that predicts the performance of servers deployed in the cloud by using CTMC [22]. Ghosh et al. proposed a stochastic reward net that quantifies the resiliency of IaaS cloud by using CTMC [23, 24]. However, to the best of our knowledge, CTMC has never been applied in the research of cloud elasticity. Our work in this paper adopts a CTMC model for effective elasticity measurement.

## 3. Definition of Cloud Elasticity

In this section, we first present a detailed discussion of different states which characterize the elastic behavior of a system. Then, we formally define elasticity that is applied in cloud platforms.

*3.1. Notations and Preliminaries.* For clarity and convenience, Notations describes the correlated variables which are used in the following sections. To elaborate the essence of cloud elasticity, we give the various states that are used in our discussion. Let $i$ denote the number of VMs in service and let $j$ be the number of requests in the system.

(1) *Just-in-Need State*. A cloud platform is in a just-in-need state if $i < j \leqslant 3i$. $T_j$ is defined as the accumulated time in all just-in-need states.

(2) *Overprovisioning State*. A cloud platform is in an overprovisioning state if $0 \leqslant j \leqslant i$. $T_o$ is defined as the accumulated time in all overprovisioning states.

(3) *Underprovisioning State*. A cloud platform is in an underprovisioning state if $j > 3i$. $T_u$ is defined as the accumulated time in all underprovisioning states.

Notice that constants 1 and 3 in this paper are only for illustration purpose and can be any other values, depending on how an elastic cloud platform is managed. Different cloud users and/or applications may prefer different bounds of the hypothetical just-in-need states. The length of the interval between the upper (e.g., $3i$) and lower (e.g., $i$) bounds controls the reprovisioning frequency. Narrowing down the interval leads to higher reprovision frequency for a fluctuating workload.

The just-in-need computing resource denotes a balanced state, in which the workload can be properly handled and quality of service (QoS) can be satisfactorily guaranteed. Computing resource overprovisioning, though QoS can be achieved, leads to extra but unnecessary cost to rent the cloud resources. Computing resource underprovisioning, on the other hand, delays the processing of workload and may be at the risk of breaking QoS commitment.

*3.2. Elasticity Definition in Cloud Computing.* In this section, we present our elasticity definition for a realistic cloud platform and present mathematical foundation for elasticity evaluation. The definition of elasticity is given from a computational point of view and we develop a calculation formula for measuring elasticity value in virtualized clouds. Let $T_m$ be the measuring time, which includes all the periods in the just-in-need, overprovisioning, and underprovisioning states; that is, $T_m = T_j + T_o + T_u$.

*Definition 1.* The elasticity $E$ of a cloud perform is the percentage of time when the platform is in just-in-need states; that is, $E = T_j/T_m = 1 - T_o/T_m - T_u/T_m$.

Broadly defining, elasticity is the capability of delivering preconfigured and just-in-need virtual machines adaptively in a cloud platform upon the fluctuation of the computing resources required. Practically it is determined by the time needed from an underprovisioning or overprovisioning state to a balanced resource provisioning state. Definition 1 provides a mathematical definition which is easily and accurately measurable. Cloud platforms with high elasticity exhibit high adaptivity, implying that they switch from an overprovisioning or an underprovisioning state to a balanced state almost in real time. Other cloud platforms take longer time to adjust and reconfigure computing resources. Although it is recognized that high elasticity can also be achieved via physical host standby, we argue that, with virtualization-enabled computing resource provisioning, elasticity can be delivered in a much easier way due to the flexibility of service migration and image template generation.

Elasticity $E$ reflects the degree to which a cloud platform changes upon the fluctuation of workloads and can be measured by the time of resource scaling by the quantity and types of virtual machine instances. We use the following equation to calculate its value:

$$E = 1 - \frac{(T_o + T_u)}{T_m} = 1 - \frac{T_o}{T_m} - \frac{T_u}{T_m}, \quad (1)$$

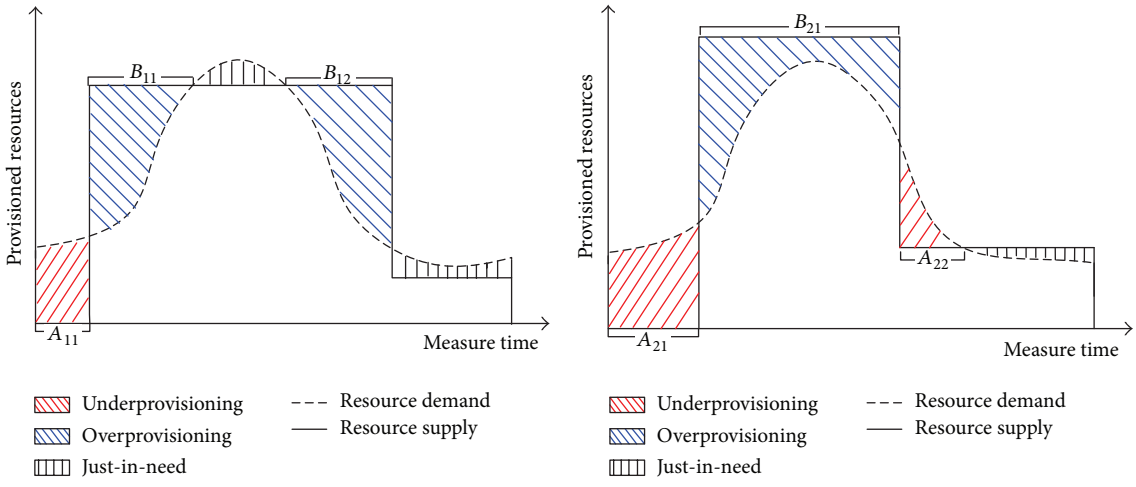
where $T_m$ denotes the total measuring time, in which $T_o$ is the overprovisioning time which accumulates each single period of time that the cloud platform needs to switch from an overprovisioning state to a balanced state and $T_u$ is the underprovisioning time which accumulates each single period of time that the cloud platform needs to switch from an underprovisioning state to a corresponding balanced state.

Let $P_j$, $P_o$, and $P_u$ be the accumulated probabilities of just-in-need states, overprovisioning states, and underprovisioning states, respectively. If $T_m$ is sufficiently long, we have $P_j = T_j/T_m$, $P_o = T_o/T_m$, and $P_u = T_u/T_m$. Therefore, we get

$$E = P_j = 1 - P_o - P_u. \quad (2)$$

Equation (1) can be used when elasticity is measured by monitoring a real system. Equation (2) can be used when elasticity is calculated by using our CTMC model. If elasticity metrics are well defined, elasticity of cloud platforms could easily be captured, evaluated, and compared.

We would like to mention that the primary factors of elasticity, that is, the amount, frequency, and time of resource reprovisioning, are all summarized in $T_o$ and $T_u$ (i.e., $P_o$ and $P_u$). Elasticity can be increased by changing these factors. For example, one can maintain a list of standby or underutilized compute nodes. These nodes are prepared for the upcoming




(a) Elastic cloud resource provisioning in cloud platform $A$

(b) Elastic cloud resource provisioning in cloud platform $B$

Figure 1: An example of elasticity metrics.

surge of workload, if there is any, to minimize the time needed to start these nodes. Such a hot standby strategy increases cloud elasticity by reducing $T_u$.

*3.3. An Example.* In Figure 1, $A_{11} = 3$ hours, $A_{21} = 5$ hours, and $A_{22} = 4$ hours are the time spans in underprovisioning states, and $B_{11} = 4$ hours, $B_{12} = 5$ hours, and $B_{21} = 10$ hours are the time spans in overprovisioning states. The measuring time of cloud platform $A$ is $T_m^A = 24$ hours and cloud platform $B$ is $T_m^B = 26$ hours. So $T_u^A = A_{11} = 3$ hours (i.e., underprovisioning time of cloud platform $A$), $T_o^A = B_{11} + B_{12} = 9$ hours (i.e., overprovisioning time of cloud platform $A$), $T_u^B = A_{21} + A_{22} = 9$ hours (i.e., underprovisioning time of cloud platform $B$), and $T_o^B = B_{21} = 10$ hours (i.e., overprovisioning time of cloud platform $B$). According to (1), the elasticity value of cloud platform $A$ is $E^A = 1 - T_o^A/T_m^A - T_u^A/T_m^A = 0.5$, and the elasticity value of cloud platform $B$ is $E^B = 1 - T_o^B/T_m^B - T_u^B/T_m^B = 0.27$. As can be seen, a greater elasticity value would exhibit better elasticity.

*3.4. Relevant Properties of Clouds.* In this section, we compare cloud elasticity with a few other relevant concepts, such as cloud resiliency, scalability, and efficiency.

*Resiliency*. Laprie [25] defined resiliency as the persistence of service delivery that can be trusted justifiably, when facing changes. Therefore, cloud resiliency implies (1) the extent to which a cloud system withstands the external workload variation and under which no computing resource reprovisioning is needed and (2) the ability to reprovision a cloud system in a timely manner. We think the latter implication defines the cloud elasticity while the former implication only exists in cloud resiliency. In our elasticity study, we will focus on the latter one.

*Scalability*. Elasticity is often confused with scalability in more ways than one. Scalability reflects the performance speedup when cloud resources are reprovisioned. In other words, scalability characterizes how *well* in terms of performance a new compute cluster, either larger or smaller, handles a given workload. On the other hand, elasticity explains how *fast* in terms of the reprovisioning time the compute cluster can be ready to process the workload. Cloud scalability is impacted by quite a few factors such as the compute node type and count and workload type and count. For example, Hadoop MapReduce applications typically scale much better than other single-thread applications. It can be defined in terms of scaling number of threads, processes, nodes, and even data centers. Cloud elasticity, on the other hand, is only constrained by the capability that a cloud service provider offers. Other factors that are relevant to cloud elasticity include the type and count of standby machines, computing resources that need to be reprovisioned. Different from cloud scalability, cloud elasticity does not concern workload/application type and count at all.

*Efficiency*. Efficiency characterizes how cloud resource can be efficiently utilized as it scales up or down. This concept is derived from speedup, a term that defines a relative performance after computing resource has been reconfigured. Elasticity is closely related to efficiency of the clouds. Efficiency is defined as the percentage of maximum performance (speedup or utilization) achievable. High cloud elasticity results in higher efficiency. However, this implication is not always true, as efficiency can be influenced by other factors independent of the system elasticity mechanisms (e.g., different implementations of the same operation). Scalability is affected by cloud efficiency. Thus, efficiency may enhance elasticity, but not sufficiency. This is due to the fact that elasticity depends on the resource types, but efficiency is not limited by resource types. For instance, with a multitenant architecture, users may exceed their resources quota. They may compete for resources or interfere each other's job executions.

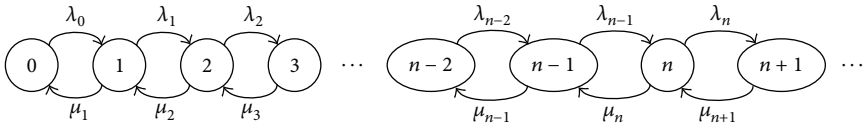



Figure 2: State-transition-rate diagram for a birth-death process.

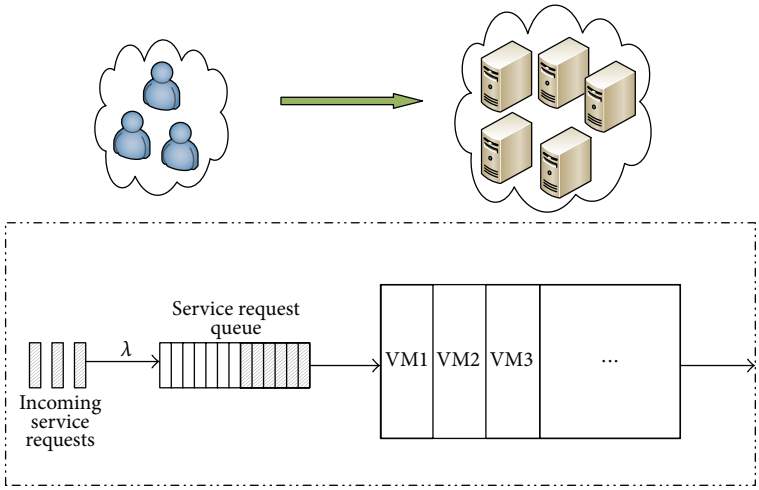



Figure 3: Modeling an elastic cloud computing platform as an extended $M/M/m$ queuing system.

## 4. Elasticity Analysis Using CTMC

In this paper, we implement the cloud elasticity computing model using CTMC.

*4.1. A Queuing Model.* This section mainly explains why the continuous-time Markov chain (CTMC) can be applied to compute cloud elasticity and the connection between them.

A continuous-time Markov chain is a continuous time, discrete-state Markov process. Many CTMC have transitions that only go to neighboring states, that is, either up one or down one; they are called birth-and-death processes. Motivated by population models, a transition up one is called a birth, while a transition down one is called a death. The birth rate in state $i$ is denoted by $\lambda_i$, while the death rate in state $i$ is denoted by $\mu_i$. The state-transition-rate diagram for a birth-and-death process (with state space $\{0, 1, \ldots, n\}$) takes the simple linear form shown in Figure 2.

In many applications, it is natural to use birth-and-death processes. One of the queuing models is $M/M/m$ queue, which has $m$ servers and unlimited waiting room. The main properties of a queuing system are as follows.

(1) Requests arrive in a Poisson process with parameter $\lambda$.

(2) The service times are exponential random variables with parameter $\mu$.

So a queuing system is a birth-and-death process with Markov property.

A cloud computing service provider serves customers' service requests by using a multiserver system. An elastic cloud computing platform treated as a multiserver system and modeled as an extended $M/M/m$ queuing system is shown in Figure 3. Assume that service requests arrive by following a Poisson process and task service times are independent and identically distributed random variables that follow an exponential distribution. When a running request finishes, the capacity used by the corresponding VM is released and becomes available for serving the next request. The request at the head of the queue is processed (i.e., first-come-first-served) on a running VM if there is capacity to run a scheduled request. Elastic resource provisioning cannot be done with physical machines, and only virtual machines can be reconfigured in real time. A cloud platform is able to adapt to variation in workload by starting up or shutting off VMs in an autonomic manner, avoiding overprovisioning or underprovisioning. If no enough running VMs are available (e.g., underprovisioning state), a new VM is started up and used for service. If there are excessive VMs (e.g., overprovisioning state), redundant VMs are shut off.

According to (1) and (2), the calculation of the elasticity value needs to count the accumulated time in all the overprovisioning and underprovisioning states. In real cloud platforms, it is possible to record the overprovisioning time and underprovisioning times. Furthermore and fortunately, the accumulated probability of both overprovisioning and underprovisioning states can be computed using our proposed CTMC model as discussed in the next section.

*4.2. Elastic Cloud Platform Modeling.* To model elastic cloud platforms, we make the following assumptions.

(i) All VMs are homogeneous with the same service capability and are added/removed one at a time.

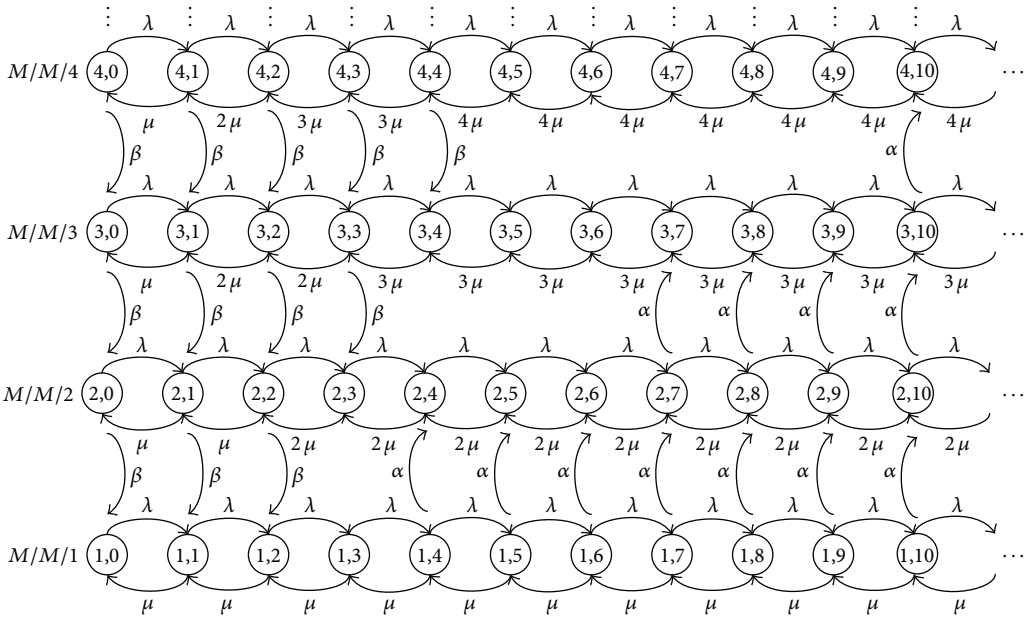



Figure 4: State-transition-rate diagram of our extended $M/M/m$ queuing system.

(ii) The user request arrivals are modeled as a Poisson process with rate $\lambda$.

(iii) The service time, the start-up time, and the shut-off time of each VM are governed by exponential distributions with rates $\mu$, $\alpha$, and $\beta$, respectively [26].

(iv) Let $i$ denote the number of virtual machines that are currently in service, and let $j$ denote the number of requests that are receiving service or in waiting.

(v) Let statev$(i, j)$ denote the various states of a cloud platform when the virtual machine number is $i$ and the request number is $j$. Let the hypothetical just-in-need state, overprovisioning state, and underprovisioning state be JIN, OP, and UP, respectively. We can set the equations of the relation between the virtual machine number and the request number as follows:

$$\text{statev}(i, j) = \begin{cases} \text{OP}, & \text{if } 0 \le j \le i; \\ \text{JIN}, & \text{if } i < j \le 3i; \\ \text{UP}, & \text{if } j > 3i. \end{cases} \tag{3}$$

The hypothetical just-in-need state, overprovisioning state, and underprovisioning state are listed in Table 1.

Based on these assumptions, we build a two-dimensional continuous-time Markov chain (CTMC) for our extended $M/M/m$ queuing system shown in Figure 4, which is actually a mixture of $M/M/m$ systems for all $m = 1, 2, 3, \ldots$. The CTMC model records the number of VMs and the number of user requests received for service, which can eventually be employed to calculate the elastic value $E$.

Each state in the model, shown in Figure 4, is labeled as $(i, j)$, where $i$ ($i \in \{1, \ldots, m\}$) denotes the number of virtual machines that are currently processing requests and $j$ ($j \in \{0, 1, \ldots, m\}$) denotes the number of requests that are receiving service. For the purpose of numerical calculation, we set the maximum number of VMs that can be deployed as $m$, which is sufficiently large to guarantee enough accuracy. Similarly, the maximum $j$ is $m$. Let $\mu$ be the service rate of each VM. So the total service rate for each state is the product of number of running VMs and $\mu$.

Table 1: The relation between the virtual machine number and the request number.

| VM number | Overprovisioning state | Just-in-need state | Underprovisioning state |
|---|---|---|---|
| 1 | $0 \leqslant j \leqslant 1$ | $1 < j \leqslant 3$ | $j > 3$ |
| 2 | $0 \leqslant j \leqslant 2$ | $2 < j \leqslant 6$ | $j > 6$ |
| 3 | $0 \leqslant j \leqslant 3$ | $3 < j \leqslant 9$ | $j > 9$ |
| 4 | $0 \leqslant j \leqslant 4$ | $4 < j \leqslant 12$ | $j > 12$ |
| ⋮ | ⋮ | ⋮ | ⋮ |
| $i$ | $0 \leqslant j \leqslant i$ | $i < j \leqslant 3i$ | $j > 3i$ |
| ⋮ | ⋮ | ⋮ | ⋮ |

The state transition in an elastic cloud computing model can occur due to user request arrival, service completion, virtual machine start-up, or virtual machine shut-off. In state $(i, j)$, according to Table 1, the state can be determined as "just-in-need," "underprovisioning," or "overprovisioning." Depending on the upcoming event, four possible transitions can occur.

*Case 1.* When a new request arrives, the system transits to state $(i, j + 1)$ with rate $\lambda$.

*Case 2.* When a requested service is completed, if the system examines the state as not "overprovisioning," the system

moves back to state $(i, j-1)$ with total service rate $i\mu$. If the system examines the state as "overprovisioning" and $i = j$, the system moves back to state $(i, j-1)$ with total service rate $(j-1)\mu$, because a server is shutting off and cannot perform any task at the moment. If the system examines the state as "overprovisioning" and $i \neq j$, the system moves back to state $(i, j-1)$ with total service rate $j\mu$.

*Case 3.* The system examines the state as "underprovisioning" and transits to state $(i+1, j)$ with rate $\alpha$.

*Case 4.* The system examines the state as "overprovisioning" and transits to state $(i-1, j)$ with rate $\beta$.

We use $P_{i,j}$ to denote the steady-state probability that the system stays in state $(i, j)$, where $i \in \{1, \ldots, m\}$ and $j \in \{0, 1, \ldots, m\}$. We can now set the balance equations as follows:

$$
\begin{aligned}
\lambda P_{i,j} &= K_2 \mu P_{i,j+1} + \beta P_{i+1,j}, && \text{if } i = 1,\ j = 0;\\
(\lambda + K_1\mu) P_{i,j} &= \lambda P_{i,j-1} + K_2 \mu P_{i,j+1} + \beta P_{i+1,j}, && \text{if } i = 1,\ 0 < j \le i+1;\\
(\lambda + K_1\mu) P_{i,j} &= \lambda P_{i,j-1} + K_2 \mu P_{i,j+1}, && \text{if } i = 1,\ i+1 < j \le 3i;\\
(\lambda + K_1\mu + \alpha) P_{i,j} &= \lambda P_{i,j-1} + K_2 \mu P_{i,j+1}, && \text{if } i = 1,\ 3i < j < m;\\
(K_1\mu + \alpha) P_{i,j} &= \lambda P_{i,j-1}, && \text{if } i = 1,\ j = m;\\
(\lambda + \beta) P_{i,j} &= K_2 \mu P_{i,j+1} + \beta P_{i+1,j}, && \text{if } 1 < i < m,\ j = 0;\\
(\lambda + K_1\mu + \beta) P_{i,j} &= \lambda P_{i,j-1} + K_2 \mu P_{i,j+1} + \beta P_{i+1,j}, && \text{if } 1 < i < m,\ 0 < j \le i;\\
(\lambda + K_1\mu) P_{i,j} &= \lambda P_{i,j-1} + K_2 \mu P_{i,j+1} + \beta P_{i+1,j}, && \text{if } 1 < i < m,\ j = i+1;\\
(\lambda + K_1\mu) P_{i,j} &= \lambda P_{i,j-1} + K_2 \mu P_{i,j+1}, && \text{if } 1 < i \le m,\ i+1 < j \le 3(i-1);\\
(\lambda + K_1\mu) P_{i,j} &= \lambda P_{i,j-1} + K_2 \mu P_{i,j+1} + \alpha P_{i-1,j}, && \text{if } 1 < i < m,\ 3(i-1) < j \le 3i;\\
(\lambda + K_1\mu + \alpha) P_{i,j} &= \lambda P_{i,j-1} + K_2 \mu P_{i,j+1} + \alpha P_{i-1,j}, && \text{if } 1 < i < m,\ 3i < j < m;\\
(K_1\mu + \alpha) P_{i,j} &= \lambda P_{i,j-1} + \alpha P_{i-1,j}, && \text{if } 1 < i < m,\ j = m;\\
(\lambda + \beta) P_{i,j} &= K_2 \mu P_{i,j+1}, && \text{if } i = m,\ j = 0;\\
(\lambda + K_1\mu + \beta) P_{i,j} &= \lambda P_{i,j-1} + K_2 \mu P_{i,j+1}, && \text{if } i = m,\ 0 < j \le i+1;\\
(\lambda + K_1\mu) P_{i,j} &= \lambda P_{i,j-1} + K_2 \mu P_{i,j+1} + \alpha P_{i-1,j}, && \text{if } i = m,\ 3(i-1) < j < m;\\
K_1 \mu P_{i,j} &= \lambda P_{i,j-1} + \alpha P_{i-1,j}, && \text{if } i = m,\ j = m,
\end{aligned}
\tag{4}
$$

where

$$
\begin{aligned}
&K_1 = j, \quad \text{if } i > j;\\
&K_1 = j-1, \quad \text{if } i = j,\ j \neq 1;\\
&K_1 = 1, \quad \text{if } i = j,\ j = 1;\\
&K_1 = i, \quad \text{if } i < j;\\
&K_2 = j+1, \quad \text{if } i > j+1;\\
&K_2 = j, \quad \text{if } i = j+1,\ j+1 \neq 1;\\
&K_2 = 1, \quad \text{if } i = j+1,\ j+1 = 1;\\
&K_2 = i, \quad \text{if } i < j+1,
\end{aligned}
\tag{5}
$$

$$
\sum_{i=1}^{m} \sum_{j=0}^{m+1} P_{i,j} = 1.
$$

In the above equations, $\lambda$, $\mu$, $\alpha$, and $\beta$ are the request arrival rate (i.e., the interarrival times of service requests are independent and identically distributed exponential random variables with mean $1/\lambda$), the service rate (i.e., the average number of tasks that can be finished by a VM in one unit of time), the virtual machine start-up rate (i.e., a VM needs time $T = 1/\alpha$ to turn on), and the virtual machine shut-off rate (i.e., a VM needs time $T = 1/\beta$ to shut down), respectively. The balance equations link the probabilities of entering and leaving a state in equilibrium. The total number of equations is $m \times (m+1) + 1$, but there are only $m \times (m+1)$ variables: $P_{1,0}, P_{1,1}, \ldots, P_{m,m}$. Therefore, in order to derive $P_{i,j}$, we need to remove one of the equations to obtain the unique equilibrium solution. Unfortunately, the steady-state balance equations cannot be solved in a closed form; hence, we must resort to a numerical solution.

The input and output parameters of our CTMC model are summarized in the following.

*Input.* The request arrival rate is $\lambda$, the service rate is $\mu$, the virtual machine start-up rate is $\alpha$, and the virtual machine shut-off rate is $\beta$. (In addition, the definitions of "just-in-need," "underprovisioning," and "overprovisioning" states should also be included.)

*Output*

(i) The accumulated underprovisioning state probability $P_u$ of a cloud platform is as follows:

$$P_u = \sum_{i=1}^{m} \sum_{j=3i+1}^{m+1} P_{i,j}, \quad (6)$$

where $P_{i,j}$ is the steady-state probability.

(ii) The accumulated overprovisioning state probability $P_o$ of a cloud platform is as follows:

$$P_o = \sum_{i=2}^{m} \sum_{j=0}^{i} P_{i,j}, \quad (7)$$

where $P_{i,j}$ is the steady-state probability.

(iii) The elasticity value $E$ of a cloud platform is obtained by (2), (6), and (7).

## 5. Model Analysis

In this section, we present some numerical results obtained based on the proposed elastic cloud platform modeling, illustrating and quantifying the elasticity value under different load conditions and different system parameters. All the numerical data in this section are obtained by setting $m$ = 1,000, that is, the maximum number of VMs that can be deployed, to guarantee sufficient numerical accuracy.

*5.1. Varying the Arrival Rate.* For the first scenario, we have considered a system with different service rates ($\mu$ = 100, 120, 140, 160, and 180 jobs/hour), while the arrival rate is a variable from $\lambda$ = 100 to 400 jobs/hour in sixteen steps. In all cases, the virtual machine start-up rate and virtual machine shut-off rate are assigned values of $\alpha$ = 120 VMs/hour and $\beta$ = 540 VMs/hour.

Figure 5 illustrates that the elasticity value is an increasing function of the arrival rate. As can been seen, it increases rather quickly when the arrival rate is up to 300 and smoothly when the arrival rate is higher. This behavior is due to the fact that increasing $\lambda$ results in noticeable reduction of the probability of overprovisioning but slight change of the probability of underprovisioning. Furthermore, it is observed that the elasticity value decreases as the service rate increases, as described in the next section.

*5.2. Varying the Service Rate.* For the second scenario, we have considered a system with different arrival rates ($\lambda$ = 200, 220, 240, 260, and 280 jobs/hour), while the service rate is a variable from $\mu$ = 10 to 290 jobs/hour in fifteen steps. In all cases, the virtual machine start-up rate and virtual machine shut-off rate are assigned values of $\alpha$ = 120 VMs/hour and $\beta$ = 540 VMs/hour.

Figure 6 illustrates that the elasticity value is a decreasing function of the service rate. It shows that, for a fixed arrival rate, increasing service rate decreases the elasticity value sharply and almost linearly. This phenomenon is due to the fact that increasing $\mu$ results in noticeable increment of the probability of overprovisioning, and change of the probability of underprovisioning does not affect the decreasing trend of the just-in-need probability. Figure 6 also confirms that the elasticity value is an increasing function of the arrival rate.

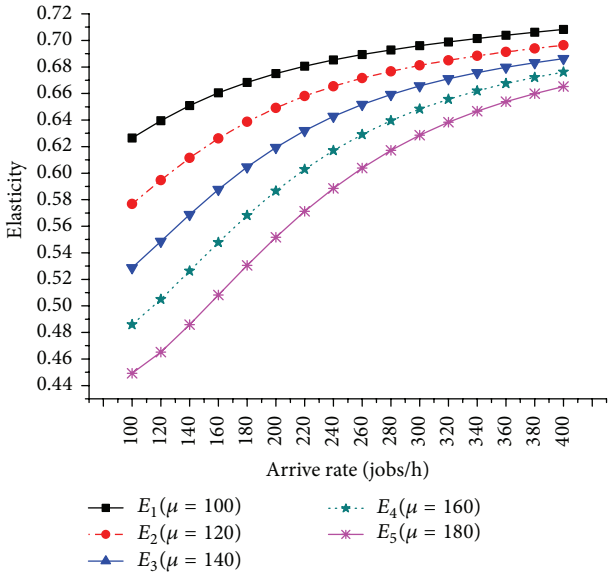



Figure 5: Elasticity versus arrival rate.

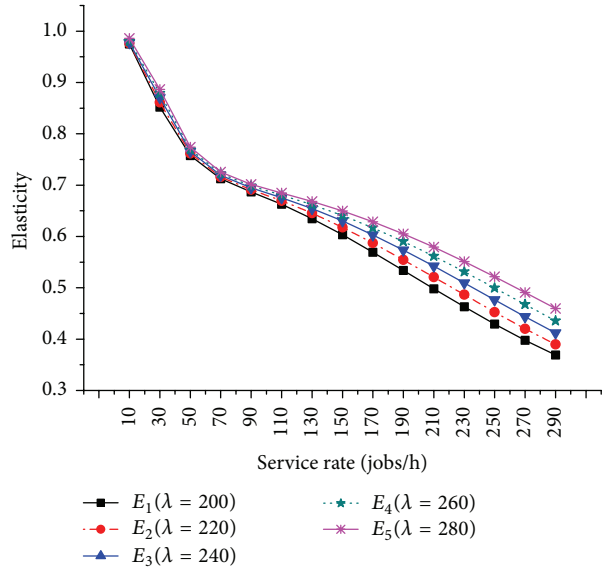
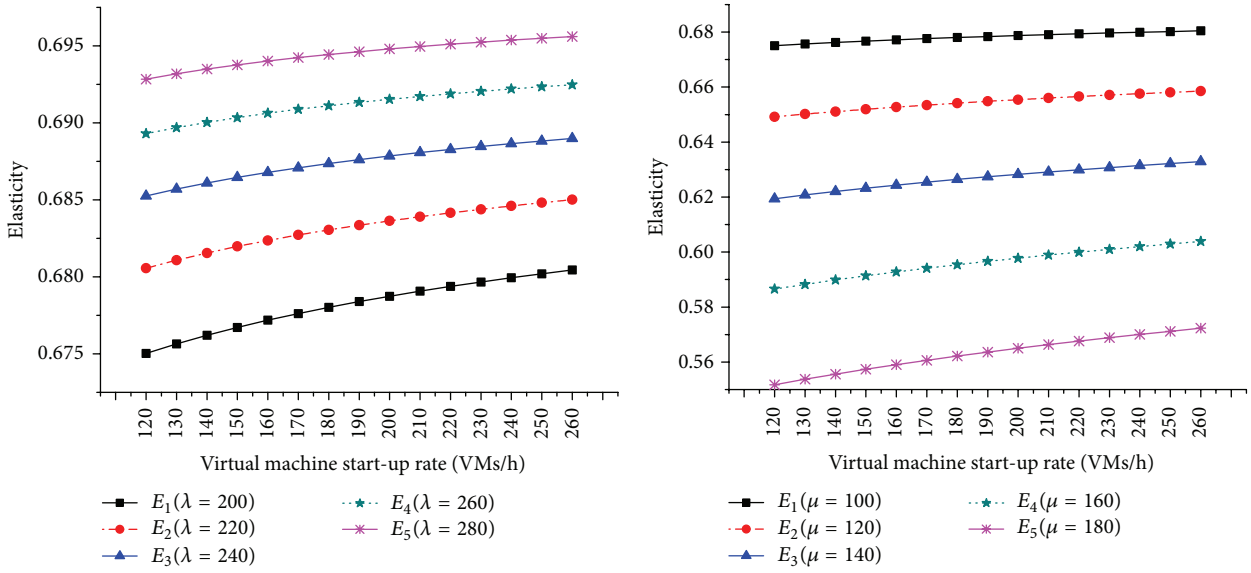



Figure 6: Elasticity versus service rate.

*5.3. Varying the Virtual Machine Start-Up Rate.* For the third scenario, Figure 7 shows numerical results for a fixed arrival rate, service rate, and virtual machine shut-off rate but different virtual machine start-up rates.

First, we characterize the elasticity value by presenting the effect of different arrival rates ($\lambda$ = 200, 220, 240, 260, and 280 jobs/hour) and the virtual machine start-up rate is a variable from $\alpha$ = 120 to 260 VMs/hour in fifteen steps. In all cases, other system parameters are set as follows. The service rate is




(a) Variable arrival rate (b) Variable service rate

FIGURE 7: Elasticity versus virtual machine start-up rate.

$\mu = 100$ jobs/hour, and the virtual machine shut-off rate is $\beta = 540$ VMs/hour. As can be seen in Figure 7(a), it increases slightly when the virtual machine start-up rate increases. This is due to the fact that, for high virtual machine start-up rate, each virtual machine start-up time is shorter, meaning that less time is needed to switch from an underprovisioning state to a balanced state, so the probability of underprovisioning $P_u$ is smaller, while the probability of overprovisioning $P_o$ does not change too much, and the probability of just-in-need $P_j$ is increasing.

Second, we also analyze the effects of different service rates ($\mu = 100$, 120, 140, 160, and 180 jobs/hour), while the virtual machine start-up rate is a variable from $\alpha = 120$ to 260 VMs/hour in fifteen steps. In all cases, other system parameters are set as follows. The arrival rate is $\lambda = 200$ jobs/hour, and the virtual machine shut-off rate is $\beta = 540$ VMs/hour. It can be seen in Figure 7(b) that the elasticity value increases slightly with increasing virtual machine start-up rate.

The results allow us to conclude the increasing elasticity value at increasing virtual machine start-up rate for a fixed arrive rate, service rate, and virtual machine shut-off rate. In other words, increasing the virtual machine start-up rate will decrease the probability of underprovisioning and increase the just-in-need probability. These behaviors (see Figure 7) confirm that the elasticity value of a cloud platform has a relationship to its virtual machine start-up speed.

*5.4. Varying the Virtual Machine Shut-Off Rate.* For the fourth scenario, Figure 8 shows numerical results for a fixed arrival rate, service rate, and virtual machine start-up rate but different virtual machine shut-off rates.

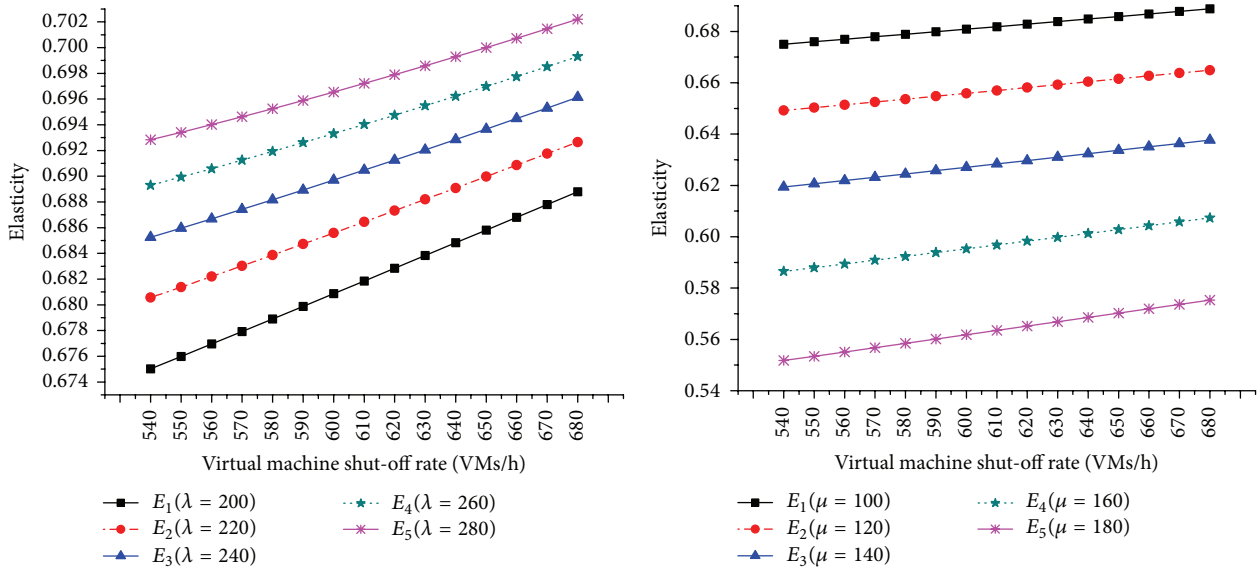
We examine the effect of virtual machine shut-off rate on elasticity. For different arrival rates ($\lambda = 200$, 220, 240, 260, and 280 jobs/hour), the virtual machine shut-off rate is a variable from $\beta = 540$ to 680 VMs/hour in fifteen steps. In all cases, other system parameters are set as follows. The service rate is $\mu = 100$ jobs/hour, and the virtual machine start-up rate is $\alpha = 120$ VMs/hour. It can be seen from Figure 8(a) that the elasticity value increases slightly where the virtual machine shut-off rate is increased from 540 to 680 VMs/hour. This happens because the virtual machine shut-off time is shorter, and a platform becomes more responsive, resulting in diminishing overprovisioning time which is the accumulate time for the system to switch from an overprovisioning state to a balanced state. Furthermore, the probability of overprovisioning $P_o$ is smaller, the probability of underprovisioning $P_u$ shows slight change, and the probability of just-in-need $P_j$ is increasing.

We also calculate the elasticity value under the different service rates ($\mu = 100, 120, 140, 160$, and 180 jobs/hour), while the virtual machine shut-off rate is a variable from $\beta = 540$ to 680 VMs/hour in fifteen steps. The arrival rate is $\lambda = 200$ jobs/hour, and the virtual machine start-up rate is $\alpha = 120$ VMs/hour. In Figure 8(b), the elasticity value increases slightly by increasing the virtual machine shut-off rate. This is also because of the corresponding reduction in virtual machine shut-off time, guaranteeing shorter overprovisioning probability $P_o$.

Based on these results, we can conclude the increasing elasticity value at increasing virtual machine shut-off rate for a fixed arrive rate, service rate, and virtual machine start-up rate. In other words, increasing the virtual machine shut-off rate will decrease the probability of overprovisioning and increase the just-in-need probability. These behaviors of Figure 8 confirm that the elasticity value of a cloud platform has a relationship to its virtual machine shut-off speed.

## 6. Simulation Results

In this section, we present our elastic cloud simulation system called *Cloud Elasticity Value*. Its aim is to demonstrate




(a) Variable arrival rate (b) Variable service rate

FIGURE 8: Elasticity versus virtual machine shut-off rate.

that our elasticity measurement is correct and effective in computing and comparing the elasticity value and to show cloud elasticity under different parameter settings.

*6.1. Design of the Simulator.* Our simulation uses the same code base for the elasticity measurement as the real implementation. The simulator is implemented in about 40,000 lines of C++ code. It runs in a Linux box over a rack-mount server with Intel®Core™2 Duo CPU and 4.00 GB of memory.

The simulator consists of four modules, that is, the task generator module, the virtual machine monitor module, the request monitor module, and the queue module. The task generator module produces simulation of Poisson distribution requests. The virtual machine monitor module is used for deciding whether to start up and shut off the virtual machines and recording the start-up and shut-off times. The request monitor module is used to count how many requests are being serviced in the system and to record the service times. Arrived service requests are first placed in a queue module and recorded their arrival times before they are processed by any virtual machine.

The main process of the simulator is listed as follows.

*Step 1.* The task generator module produces simulation of Poisson distribution requests. When the service requests arrive, they are placed in a queue module and recorded their arrival times.

*Step 2.* The virtual machine monitor module determines whether to start up or shut off the VMs and records the start-up and shut-off times.

*Step 3.* The request monitor module determines whether there is a request in the queue. If there is a request, it takes a request and assigns it to a running virtual machine and records the service time.

*Step 4.* After the simulation time is over, the simulator counts up the accumulated time in overprovisioning and underprovisioning states and returns the elasticity value using (1).

*6.2. Simulation Results and Analysis.* We have evaluated cloud elasticity values using two methods, that is, (1) the elasticity values in terms of the steady-state probabilities obtained for the given parameter and (2) the elasticity values in terms of our simulation system obtained for the same parameters. We compare our CTMC model solutions with the results produced by the simulation method.

We have considered the arrival rate characterized by $\lambda$ = 60, 200, and 600 jobs/hour. The service rate values chosen are $\mu$ = 60, 200, and 600 jobs/hour. In all cases, the virtual machine start-up rate is assigned the value of $\alpha$ = 300 VMs/hour, while the virtual machine shut-off rate is $\beta$ = 540 VMs/hour.

Table 2 shows the difference between the elasticity values obtained by the CTMC model and the simulator. From Table 2, we can see that the elasticity values between the two cases are very close, with the maximum relative difference only 0.8 percent. The agreement between the simulation and CTMC model results is excellent, which confirms the validity of our CTMC model. So we conclude that the proposed elasticity quantifying and measuring method using the continuous-time Markov chain (CTMC) model is correct and effective.

## 7. Experiments on Real Systems

*7.1. Experiment Environment.* We have conducted our experiments on LuCloud, a cloud computing environment located in Hunan University. On top of hardware and Ubuntu Linux 12.04 operating system, we install KVM virtualization

Table 2: Comparison of CTMC model results and simulation results.

| Arrival rate | Service rate | Start-up rate | Shut-off rate | Method | | Difference |
|---|---|---|---|---|---|---|
| | | | | CTMC model | Simulation | |
| 60 | 60 | 300 | 540 | 0.705212 | 0.702837 | 0.3% |
| 60 | 200 | 300 | 540 | 0.445756 | 0.475257 | 0.4% |
| 60 | 600 | 300 | 540 | 0.635151 | 0.637240 | 0.3% |
| 200 | 60 | 300 | 540 | 0.735167 | 0.739055 | 0.5% |
| 200 | 200 | 300 | 540 | 0.543948 | 0.546414 | 0.5% |
| 200 | 600 | 300 | 540 | 0.235727 | 0.234727 | 0.4% |
| 600 | 60 | 300 | 540 | 0.827098 | 0.828784 | 0.2% |
| 600 | 200 | 300 | 540 | 0.688974 | 0.684784 | 0.6% |
| 600 | 600 | 300 | 540 | 0.435171 | 0.438784 | 0.8% |

Table 3: Comparison of CTMC model results and experimental results with exponential service times.

| Arrival rate | Service rate | Start-up rate | Shut-off rate | Method | | Difference |
|---|---|---|---|---|---|---|
| | | | | CTMC model | Experiment | |
| 60 | 60 | 120 | 540 | 0.701205 | 0.706082 | 0.6% |
| 60 | 200 | 120 | 540 | 0.475480 | 0.479752 | 0.9% |
| 60 | 600 | 120 | 540 | 0.631525 | 0.649275 | 3.0% |
| 200 | 60 | 120 | 540 | 0.731117 | 0.739533 | 1.2% |
| 200 | 200 | 120 | 540 | 0.516099 | 0.521308 | 1.0% |
| 200 | 600 | 120 | 540 | 0.221065 | 0.269039 | 2.2% |
| 600 | 60 | 120 | 540 | 0.817088 | 0.826455 | 1.1% |
| 600 | 200 | 120 | 540 | 0.687533 | 0.681169 | 0.9% |
| 600 | 600 | 120 | 540 | 0.409537 | 0.491034 | 0.9% |

software which virtualizes the infrastructure and provides unified computing and storage resources. To create a cloud environment, we install CloudStack open-source cloud environment, which is composed of a cluster and responsible for global management, resource scheduling, task distribution, and interaction with users. The cluster is managed by a cloud manager (8 AMD Opteron Processor 4122 CPU, 8 GB memory, and 1 TB hard disk). We use our elasticity testing platform to achieve the allocation of resources, that is, virtual machine start-up and shut-off on LuCloud.

*7.2. Experiment Process and Results.* First, in order to validate the proposed model, Table 3 summarizes the comparison between the two approaches, that is, the CTMC model and the experiments on LuCloud. We have considered the arrival rate characterized by $\lambda = 60$, 200, and 600 jobs/hour. The service rate values chosen are $\mu = 60$, 200, and 600 jobs/hour. The virtual machine start-up rate is assigned the value of $\alpha = 120$ VMs/hour, and the virtual machine shut-off rate is assigned the value of $\beta = 540$ VMs/hour.

In Table 3, we can observe that the elasticity values of both approaches are very close, with the maximum relative difference only 3.0 percent. We conclude that the proposed CTMC model can be used to compute the elasticity of cloud platforms and can offer accurate results within reasonable difference.

Our second set of experiments focus on the robustness of our model and method, that is, its applicability when the assumptions of our model are not satisfied. We have considered Gamma distributions for the service times. The Gamma$(k, \theta)$ distribution is defined in terms of a shape parameter $k$ and a scale parameter $\theta$ [27]. We use the same parameter settings in Table 3, except that the exponential distributions of service times are replaced by Gamma distributions, that is, Gamma$(0.0083, 2)$, Gamma$(0.0025, 2)$, and Gamma$(0.00083, 2)$, such that the service rates are still $\mu =$ 60, 200, and 600 jobs/hour.

From Table 4, we can see that the elasticity values of the CTMC model and the experiments are very close, with the maximum relative difference only 3.3 percent. We observe that the experimental results with Gamma service times match very closely with those of the proposed CTMC model. So we conclude that the proposed model and method for quantifying and measuring elasticity using continuous-time Markov chain (CTMC) are not only correct and effective, but also robust and applicable to real cloud computing platforms.

## 8. Conclusion

In this paper, we have introduced a new definition of cloud elasticity. We have presented an analytical method suitable for evaluating the elasticity of cloud platforms, by using a continuous-time Markov chain (CTMC) model. Validation of the analytical results through extensive simulations has shown that our analytical model is sufficiently detailed to capture all realistic aspects of resource allocation process,

Table 4: Comparison of CTMC model results and experimental results with Gamma service times.

| Arrival rate | Start-up rate | Shut-off rate | Service distribution | Method | | Difference |
|---|---|---|---|---|---|---|
| | | | | CTMC model | Experiment | |
| 60 | 120 | 540 | Gamma(0.0083, 2) | 0.701205 | 0.716401 | 2.2% |
| 60 | 120 | 540 | Gamma(0.0025, 2) | 0.475480 | 0.476752 | 0.3% |
| 60 | 120 | 540 | Gamma(0.00083, 2) | 0.631525 | 0.612930 | 3.0% |
| 200 | 120 | 540 | Gamma(0.0083, 2) | 0.731117 | 0.711420 | 2.7% |
| 200 | 120 | 540 | Gamma(0.0025, 2) | 0.516099 | 0.522762 | 1.3% |
| 200 | 120 | 540 | Gamma(0.00083, 2) | 0.221065 | 0.227146 | 2.8% |
| 600 | 120 | 540 | Gamma(0.0083, 2) | 0.817088 | 0.835865 | 2.3% |
| 600 | 120 | 540 | Gamma(0.0025, 2) | 0.687533 | 0.667146 | 3.0% |
| 600 | 120 | 540 | Gamma(0.00083, 2) | 0.409537 | 0.395865 | 3.3% |

that is, virtual machine start-up and virtual machine shut-off, while maintaining excellent accuracy between CTMC model results and simulation results. We have examined the effects of various parameters including request arrival rate, service time, virtual machine start-up rate, and virtual machine shut-off rate. Our experimental results further evidence that the proposed measurement approach can be used to compute cloud elasticity in real cloud platforms. Consequently, cloud providers and users can obtain quantitative, informative, and reliable estimation of elasticity, based on a few essential characterizations of a cloud computing platform.

## Notations

$E$: The elasticity value
$i$: The number of VMs in service
$j$: The number of requests in the queue
$T_j$: The accumulated just-in-need time
$T_o$: The accumulated overprovisioning time
$T_u$: The accumulated underprovisioning time
$T_m$: The measuring time
$P_j$: The accumulated probability of just-in-need states
$P_o$: The accumulated probability of overprovisioning states
$P_u$: The accumulated probability of underprovisioning states
$\lambda$: The request arrival rate
$\mu$: The request service rate
$\alpha$: The virtual machine start-up rate
$\beta$: The virtual machine shut-off rate
$(i, j)$: A state in our CTMC model
$P_{i,j}$: The steady-state probability of state $(i, j)$
$N$: The average number of requests in the queue
$T$: The average response time
$M$: The average number of VMs in service
CPR: The cost-performance ratio.

## Competing Interests

The authors declare that they have no competing interests.

## Acknowledgments

The research was partially funded by the Key Program of National Natural Science Foundation of China (Grants nos. 61133005 and 61432005) and the National Natural Science Foundation of China (Grants nos. 61370095 and 61472124).

## References

[1] J. Cao, K. Li, and I. Stojmenovic, "Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 45–58, 2014.

[2] M. Bourguiba, K. Haddadou, I. E. Korbi, and G. Pujolle, "Improving network I/O virtualization for cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 673–681, 2014.

[3] Cost-efficient consolidating service for Aliyun's cloud-scale computing, http://kylinx.com/papers/c4.pdf.

[4] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: what it is, and what it is not," in *Proceedings of the 10th International Conference on Autonomic Computing (ICAC '13)*, pp. 23–27, San Jose, Calif, USA, June 2013.

[5] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "CloudScale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC '11)*, p. 5, ACM, Cascais, Portugal, October 2011.

[6] G. Galante and L. C. E. de Bona, "A survey on cloud computing elasticity," in *Proceedings of the IEEE/ACM 5th International Conference on Utility and Cloud Computing (UCC '12)*, pp. 263–270, Chicago, Ill, USA, November 2012.

[7] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS '11)*, pp. 559–570, IEEE, Minneapolis, Minn, USA, July 2011.

[8] R. N. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya, "The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid clouds," *Future Generation Computer Systems*, vol. 28, no. 6, pp. 861–870, 2012.

[9] J. O. Fitó, Í. Goiri, and J. Guitart, "SLA-driven elastic cloud hosting provider," in *Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP '10)*, pp. 111–118, IEEE, Pisa, Italy, February 2010.

[10] L. Badger, T. Grance, R. Patt-Corner, and J. Voas, *Draft Cloud Computing Synopsis and Recommendations*, vol. 800, NIST Special Publication, 2011.

[11] R. Cohen, *Defining Elastic Computing*, 2009, http://www.elasticvapor.com/2009/09/defining-elastic-computing.html.

[12] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud Computing: Principles and Paradigms*, vol. 87, John Wiley & Sons, New York, NY, USA, 2010.

[13] S. Dustdar, Y. Guo, B. Satzger, and H.-L. Truong, "Principles of elastic processes," *IEEE Internet Computing*, vol. 15, no. 5, pp. 66–71, 2011.

[14] K. Hwang, X. Bai, Y. Shi, M. Li, W. Chen, and Y. Wu, "Cloud performance modeling with benchmark evaluation of elastic scaling strategies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 130–143, 2016.

[15] M. Kuperberg, N. Herbst, J. von Kistowski, and R. Reussner, *Defining and Quantifying Elasticity of Resources in Cloud Computing and Scalable Platforms*, KIT, Fakultät für Informatik, 2011.

[16] W. Dawoud, I. Takouna, and C. Meinel, "Elastic VM for cloud resources provisioning optimization," in *Advances in Computing and Communications*, A. Abraham, J. L. Mauri, J. F. Buford, J. Suzuki, and S. M. Thampi, Eds., vol. 190 of *Communications in Computer and Information Science*, pp. 431–445, Springer, Berlin, Germany, 2011.

[17] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings of the IEEE 4th International Conference on Cloud Computing (CLOUD '11)*, pp. 500–507, IEEE, Washington, DC, USA, July 2011.

[18] Z. Gong, X. Gu, and J. Wilkes, "Press: predictive elastic resource scaling for cloud systems," in *Proceedings of the International Conference on Network and Service Management (CNSM '10)*, pp. 9–16, IEEE, Ontario, Canada, October 2010.

[19] W. J. Anderson, *Continuous-Time Markov Chains*, Springer Series in Statistics: Probability and Its Applications, Springer, New York, NY, USA, 1991.

[20] H. Khazaei, J. Mišić, V. B. Mišić, and S. Rashwand, "Analysis of a pool management scheme for cloud computing centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 5, pp. 849–861, 2013.

[21] R. Ghosh, V. K. Naik, and K. S. Trivedi, "Power-performance trade-offs in IaaS cloud: a scalable analytic approach," in *Proceedings of the IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W '11)*, pp. 152–157, IEEE, Hong Kong, June 2011.

[22] S. Pacheco-Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, "Markovian workload characterization for QoS prediction in the cloud," in *Proceedings of the IEEE 4th International Conference on Cloud Computing (CLOUD '11)*, pp. 147–154, IEEE, Washington, Wash, USA, July 2011.

[23] R. Ghosh, F. Longo, V. K. Naikz, and K. S. Trivedi, "Quantifying resiliency of IaaS cloud," in *Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems*, pp. 343–347, IEEE, New Delhi, India, November 2010.

[24] R. Ghosh, D. Kim, and K. S. Trivedi, "System resiliency quantification using non-state-space and state-space analytic models," *Reliability Engineering & System Safety*, vol. 116, pp. 109–125, 2013.

[25] J.-C. Laprie, "From dependability to resilience," in *Proceedings of the 38th IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. G8–G9, Anchorage, Alaska, USA, June 2008.

[26] M. Mao and M. Humphrey, "A performance study on the VM startup time in the cloud," in *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD '12)*, pp. 423–430, IEEE, Honolulu, Hawaii, USA, June 2012.

[27] S. Ali, H. J. Siegel, M. Maheswaran, and D. Hensgen, "Task execution time modeling for heterogeneous computing systems," in *Proceedings of the IEEE 9th Heterogeneous Computing Workshop (HCW '00)*, pp. 185–199, Cancun, Mexico, 2000.