Quantitative Modeling and Analytical Calculation of Elasticity in Cloud Computing

Keqin Li, *Fellow, IEEE*

Abstract—Elasticity is a fundamental feature of cloud computing and can be considered as a great advantage and a key benefit of cloud computing. One key challenge in cloud elasticity is lack of consensus on a quantifiable, measurable, observable, and calculable definition of elasticity and systematic approaches to modeling, quantifying, analyzing, and predicting elasticity. Another key challenge in cloud computing is lack of effective ways for prediction and optimization of performance and cost in an elastic cloud platform. The present paper makes the following significant contributions. First, we present a new, quantitative, and formal definition of elasticity in cloud computing, i.e., the probability that the computing resources provided by a cloud platform match the current workload. Our definition is applicable to any cloud platform and can be easily measured and monitored. Furthermore, we develop an analytical model to study elasticity by treating a cloud platform as a queueing system, and use a continuous-time Markov chain (CTMC) model to precisely calculate the elasticity value of a cloud platform by using an analytical and numerical method based on just a few parameters, namely, the task arrival rate, the service rate, the virtual machine start-up and shut-down rates. In addition, we formally define auto-scaling schemes and point out that our model and method can be easily extended to handle arbitrarily sophisticated scaling schemes. Second, we apply our model and method to predict many other important properties of an elastic cloud computing system, such as average task response time, throughput, quality of service, average number of VMs, average number of busy VMs, utilization, cost, cost-performance ratio, productivity, and scalability. In fact, from a cloud consumer's point of view, these performance and cost metrics are even more important than the elasticity metric. Our study in this paper has two significance. On one hand, a cloud service provider can predict its performance and cost guarantee using the results developed in this paper. On the other hand, a cloud service provider can optimize its elastic scaling scheme to deliver the best cost-performance ratio. To the best of our knowledge, this is the first paper that analytically and comprehensively studies elasticity, performance, and cost in cloud computing. Our model and method significantly contribute to the understanding of cloud elasticity and management of elastic cloud computing systems.

Index Terms—Cloud computing, continuous-time Markov chain, cost-performance ratio, elasticity, queueing model

1 Introduction

1.1 Challenges and Motivations

1.1.1 *Elasticity Characterization*

Cloud computing is a paradigm for enabling ubiquitous, convenient, and on-demand network accesses to a shared pool of configurable computing resources (e.g., servers, storage, networks, data, software, applications, and services), that can be rapidly provisioned and released with minimal management effort or service provider interaction [32]. The unique and essential characteristics of cloud computing include on-demand self-service, broad and variety of network access, resource pooling and sharing, rapid elasticity, measured and metered service. Among these features, elasticity is a fundamental and key feature of cloud computing, which can be considered as a great advantage and a key benefit of cloud computing, and perhaps what distinguishes this new computing paradigm from other ones, such as cluster and grid computing [14].

The Merriam-Webster dictionary defines elasticity as the capability of a strained body to recover its size and shape after deformation. Its synonyms include stretchiness, flexibility, pliancy, suppleness, plasticity, resilience, springiness, sponginess, and adaptability. In physics, elasticity (from Greek *ελαστικότητα*, "elastikótita") is the tendency of solid materials to return to their original shape after being deformed. A solid object will deform when forces are applied on it. If the material is elastic, the object will return to its initial status (e.g., shape and size) when these forces are removed. A cloud computing platform is like a solid object. The resource (e.g., virtual machines (VMs)) utilization and quality of service (QoS, e.g., the average task response time) are properties and status of the platform. The dynamic workload (e.g., the number of service requests) changes are external forces. When the workload increases (decreases, respectively), the resource utilization increases (decreases, respectively), and the service quality decreases (increases, respectively), e.g., the average task response time increases (decreases, respectively), i.e., the cloud computing platform is deformed. To return to its original status, the platform should have the capability to adjust itself, e.g., increasing (decreasing, respectively) the number of VMs, so that both resource utilization and quality of service can return to their original status. Notice that the above definition of elasticity is only qualitative, but not quantitative. The most important problem in studying cloud elasticity is the apparent lack of a quantifiable, measurable, and observable

• *The author is with the Department of Computer Science, State University of New York, New Paltz, New York 12561, USA. E-mail: lik@newpaltz.edu.*

Manuscript received 31 Jan. 2016; revised 28 Dec. 2016; accepted 2 Feb. 2017. Date of publication 0 . 0000; date of current version 0 . 0000.
Recommended for acceptance by M. Parashar, O. Rana, and R.C.H. Hsu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2017.2665549

definition of elasticity in cloud computing, and thus no approach to analyzing and predicting elasticity has been well developed so far, although several researchers have attempted to characterize cloud elasticity (see Section 2.1). Such a definition allows for the creation of analytical models and methods that not only calculate elasticity, but also enable deployment, management, improvement, and enhancement of cloud computing platforms.

In economics, elasticity is the measurement of how responsive an economic variable is to a change in another. In particular, elasticity can be quantified as the ratio of the percentage change in one variable to the percentage change in another variable. Using this definition, elasticity in cloud computing can be defined as how the amount of computing resource changes as the current workload changes. It seems that the definition is quantitative and measurable; however, such a definition of responsiveness is not entirely adequate, since it only considers how much, not how fast, the computing resource adapts. If a cloud computing platform takes a long time to provide the correct amount of resources to match the workload (which might not be current any more), it is not considered as elastic. The time required to restore the original status, so that the provided computing resources match the current workload, should be taken into account. Elasticity (i.e., the ability to dynamically acquire or release computing resources in response to variable demand) is meaningful to the cloud users only when the acquired VMs can be provisioned in time and ready to use within the user expectation. The long unexpected VM start-up time could result in resource under-provisioning, which will inevitably hurt system performance [30]. Similarly, the long unexpected VM shut-down time could result in resource over-provisioning, which will inevitably hurt resource utilization.

1.1.2 Performance and Cost Optimization

In addition to the issues mentioned above, existing studies of elasticity mostly focused on characterizing elasticity, but emphasized much less from users' point of view. Customers of cloud services only care high quality of service and low cost of service, and do not care whether such quality and cost are supported by elasticity. Therefore, the ultimate purpose of elasticity is to benefit the users, although such elastic management of a cloud computing platform is transparent to users and applications. All efforts in studying elasticity should be incorporated into performance and cost control, management, prediction, and optimization.

Elasticity research should help in the following two ways.

- Performance and cost predictability—The analytical models and methods developed for measuring elasticity should help to make the performance and cost of a cloud computing platform predictable, manageable, and improvable.
- Auto-scaling scheme optimality—The models and methods should also be able to guide the construction, optimization, and comparison of auto-scaling schemes for the best interest of the users of an elastic cloud computing platform.

Unfortunately, the above challenges have not been well investigated in the existing literature.

1.2 Contributions of the Paper

As mentioned above, one key challenge in cloud elasticity is lack of consensus on a quantifiable, measurable, observable, and calculable definition of elasticity and systematic approaches to modeling, quantifying, analyzing, and predicting elasticity. Another key challenge in cloud computing is lack of effective ways for prediction and optimization of performance and cost in an elastic cloud platform. The main objective of this paper is to address these two pressing issues.

Our contributions in this paper can be summarized as follows.

First, we present a new, quantitative, and formal definition of elasticity in cloud computing, i.e., the probability that the computing resources provided by a cloud platform match the current workload. Our definition is applicable to any cloud platform and can be easily measured and monitored. Furthermore, we develop an analytical model to study elasticity by treating a cloud platform as a queueing system, and use a continuous-time Markov chain (CTMC) model to precisely calculate the elasticity value of a cloud platform by using an analytical and numerical method based on just a few parameters, namely, the task arrival rate, the service rate, the virtual machine start-up and shut-down rates. In addition, we formally define auto-scaling schemes and point out that our model and method can be easily extended to handle arbitrarily sophisticated scaling schemes.

Second, we apply our model and method to predict many other important properties of an elastic cloud computing system, such as average task response time, throughput, quality of service, average number of VMs, average number of busy VMs, utilization, cost, cost-performance ratio, productivity, and scalability. In fact, from a cloud consumer's point of view, these performance and cost metrics are even more important than the elasticity metric. Our study in this paper has two significance. On one hand, a cloud service provider can predict its performance and cost guarantee using the results developed in this paper. On the other hand, a cloud service provider can optimize its elastic scaling scheme to deliver the best cost-performance ratio. We also show that an elastic platform can consume less resources, achieve shorter average task response time, provide the same performance guarantee with higher probability, and have less cost and lower cost-performance ratio than an inelastic platform.

To the best of our knowledge, this is the first paper that analytically and comprehensively studies elasticity, performance, and cost in cloud computing. Our model and method significantly contribute to the understanding of cloud elasticity and management of elastic cloud computing systems.

2 Related Research

In this section, we review four areas related to our study, i.e., cloud elasticity characterization, elastic cloud computing system development, cloud platform modeling and analysis, and elastic system performance assessment.

2.1 Characterizing Cloud Elasticity

Several researchers have attempted to characterize cloud elasticity. These definitions are classified into two categories. The first category includes those definitions which are only qualitative, but not quantitative. In [5], elasticity is defined as the ability for customers to quickly request, receive, and later release as many resources as needed. Elastic computing has the feature of dynamic variation in the use of computer resources to meet a varying workload [7]. In [20], elasticity is defined 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 [27], elasticity is the feature of automated, dynamic, flexible, and frequent resizing of resources that are provided to an application by the execution platform. However, all these characterizations are not quantified.

The second category includes those definitions which are quantitative, but not analytically tractable. Some attempts have been made to propose a quantitative and measurable definition of cloud elasticity. It is mentioned in [27] that a unified (single-valued) metric for elasticity could possibly be achieved by a combination of three characteristics, namely, reconfiguration effect (i.e., the amount of added/removed resources, expressing the granularity of adaptation), reconfiguration frequency (i.e., the density of reconfiguration points over a time period), and reconfiguration time (i.e., the time interval between the instant when a reconfiguration has been triggered/requested and the instant when the adaptation has been completed), in such a way that the elasticity metric is in the range of $[0, 1]$. Although each of the above three properties can be observed and measured, there is no specific equation or formula given in [27] for such a single-valued elasticity metric. In [20], an elasticity metric for scaling up (down, respectively) is defined in such a way that it is inversely proportional to the product of the average time to switch from an under-provisioned (over-provisioned, respectively) state to a normal state, which corresponds to the average speed of scaling up (down, respectively), and the average amount of under-provisioned (over-provisioned, respectively) resources during an under-provisioned (over-provisioned, respectively) period. Since theoretically, the speed of scaling can be arbitrarily fast, the above definition can possibly lead to an "infinitely elastic" cloud computing system. Furthermore, although each of the above two properties can be monitored and measured, there is no given method to predict, e.g., the average amount of under-provisioned or over-provisioned resources, and therefore, there is no way to obtain elasticity analytically. In [22], a definition of elasticity was given, which relates elasticity with over-provisioning and under-provisioning penalties. However, the amounts of over-provisioning and under-provisioning are only observable, but not analytically available and predictable.

Some other efforts have also been made to study elasticity. In [12], elasticity properties have been considered in terms of cost elasticity (i.e., the responsiveness of resource provision to changes in cost) and quality elasticity (i.e., the responsiveness of quality to changes in resource usage). In [14], elastic systems are classified in terms of four characteristics, i.e., scope (infrastructure, application, platform), policy (manual, reactive, predictive), purpose (performance, capacity, cost, energy), and method (replication, resizing, migration). In [37], application elasticity is considered, i.e., making an application automatically adjust to variations in load without the need of intervention of a human administrator and without the need to change its code.

2.2 Developing Elastic Computing Systems

In [8], the authors described a platform for developing scalable applications on the cloud by QoS-driven resource provisioning from different sources and supporting different and elastic applications. In [11], the authors considered elastic VMs for rapid and optimal virtualized resources allocation. In [13], the authors presented an elastic web hosting provider, that makes use of the outsourcing technique in order to take advantage of cloud computing infrastructures for providing scalability and high availability capabilities to the web applications. In [18], the authors presented a novel predictive elastic resource scaling scheme for cloud systems, which unobtrusively extracts fine-grained dynamic patterns in application resource demands and adjusts their resource allocation automatically. In the context of cloud computing, auto-scaling mechanisms hold the promise of assuring QoS properties for applications, while simultaneously making efficient use of resources and keeping operational costs low for the service providers. In [34], the authors developed a model-predictive algorithm for workload forecasting that is used for resource auto-scaling. In [35], the authors developed a cost-aware system that provides efficient support for elasticity in the cloud by (i) leveraging multiple mechanisms to reduce the time to transition to new configurations, and (ii) optimizing the selection of a virtual server configuration that minimizes the cost. Elastic resource scaling allows cloud systems to meet application service-level agreements (SLA) with minimum resource provisioning costs. In [36], the authors presented a system that automates fine-grained elastic resource scaling for multi-tenant cloud computing infrastructures.

In [1], the authors presented a service-oriented dynamic resource management model, which covers the issues of resource prediction, customer type-based resource estimation and reservation, advanced reservation, pricing, refunding and acquired quality of service-based refunding. In [2], the authors provided a holistic brokerage model to manage on-demand and advance service reservation, pricing, and reimbursement, with dynamic management of customer's characteristics and historical record in evaluating the economics related factors.

2.3 Modeling Cloud Platforms

2.4 Assessing Elastic System Performance

(Due to space limitation, Sections 2.3 and 2.4 are moved to the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCC.2017.2665549.)

3 Definition of Elasticity

In this section, we formally define cloud elasticity, and also compare the notion with several related concepts. For reader's convenience, we provide Table 1, which gives a summary of notations and their definitions in the order introduced in the paper.

TABLE 1
Summary of Notations and Definitions

Notation	Definition
E	elasticity
p_{normal}	the probability in a normal state
p_{over}	the probability in an over-provisioning state
p_{under}	the probability in an under-provisioning state
m	the number of active servers (i.e., VMs)
λ	the task arrival rate
μ	the service rate
k	the number of tasks in the system
(m, k)	a state
(a_m, b_m)	a pair of integers defining different states
S	an elastic cloud management and auto-scaling scheme
α	the VM start-up rate
β	the VM shut-down rate
$p(m, k)$	the equilibrium steady-state probability of state (m, k)
N	the average number of tasks
T	the average task response time
R	the throughput
M	the average number of servers
B	the average number of busy servers
U	the VM utilization
ρ	the server utilization
p_k	the probability that a queueing system is in state k
τ	the average response time randomized over k

3.1 A New Definition

It has been clear based on our discussion so far that a definition of elasticity in cloud computing should satisfy the following two conditions.

- Quantitative describability—the definition should be quantifiable, measurable, and observable, which is based on a few parameters and is formally defined based on a rigorous model.
- Analytical tractability—the definition should be analytically available, calculable, and predictable, which is easily obtained by using a simple, standard, and straightforward method.

We say that a cloud computing system is in (1) a normal state if the provided computing resources match the current workload; (2) an over-provisioning state if the provided computing resources exceed the current workload; (3) an under-provisioning state if the provided computing resources cannot handle the current workload. Our definition of *elasticity* of a cloud computing platform with dynamically variable workload is *the percentage of time (or, the probability) that the system is in the normal state*.

Formally, assume that a system is operating for a time period of length T. Let T_{normal} (T_{over}, T_{under}, respectively) be the total time that the system is in the normal (over-provisioning, under-provisioning, respectively) state. It is clear that $T = T_{\text{normal}} + T_{\text{over}} + T_{\text{under}}$. Then, the elasticity is calculated as

$$E = \frac{T_{\text{normal}}}{T} = 1 - \frac{T_{\text{over}} + T_{\text{under}}}{T}. \tag{1}$$

If the system has been operating for a sufficiently long period of time and is in a stable state, then $p_{\text{normal}} = T_{\text{normal}}/T$ is the probability that the system is in the normal state, $p_{\text{over}} = T_{\text{over}}/T$ is the probability that the system is in the over-provisioning state, and $p_{\text{under}} = T_{\text{under}}/T$ is the probability that the system is in the under-provisioning state. Hence, we get

$$E = p_{\text{normal}} = 1 - (p_{\text{over}} + p_{\text{under}}). \tag{2}$$

Notice that our definition of elasticity in Eq. (1) is easily measurable and observable by monitoring a cloud computing platform. Of course, the notions of normal, over-provisioning, and under-provisioning states still need to be quantified. Since our elasticity metric is defined quantitatively as probability, its value is in the range $[0, 1]$. Analytical tractability is impossible unless there is a rigorous mathematical model. We will present a queueing model for cloud platforms, define auto-scaling schemes, employ a CTMC model for elastic cloud platforms and quantitatively characterize our metric, and develop an analytical and numerical method to compute the proposed metric of Eq. (2), thus satisfying the two requirements mentioned earlier. It will also be clear that our elasticity metric depends on only a few (five, in particular) parameters.

It is also noticed that our definition of elasticity captures the three characteristics in [27], i.e., reconfiguration effect, reconfiguration frequency, and reconfiguration time, and the two characteristics in [20], i.e., the average time to switch and the average amount of under-provisioned or over-provisioned resources, where the reconfiguration effect and the average amount of under-provisioned or over-provisioned resources affect the definition of normal/over-provisioning/under-provisioning states, and the reconfiguration frequency, the reconfiguration time, and the average time to switch are all reflected and summarized in E, i.e., T_{over}, T_{under}, p_{over}, and p_{under}.

3.2 Related Notions and Properties

There are several concepts which are related to (and sometimes considered as similar to or even the same as) elasticity. In the following, we clarify the difference between these concepts and elasticity.

Resilience. In material science, resilience is the ability of a material to absorb energy when it is deformed elastically, and release that energy upon unloading. Resiliency is the persistence of service delivery that can justifiably be trusted when facing changes, which should be considered as different from fault-tolerance, reliability, availability, recoverability, and performability [15]. In [16], the authors quantified the resiliency of Infrastructure-as-a-Service (IaaS) clouds subject to changes in demand and available capacity, using a stochastic reward net based model for provisioning and servicing requests, with respect to two key performance measures, i.e., job rejection rate and provisioning response delay.

Scalability. Scalability is the ability of a system, network, or process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth. A scalable system improves its performance proportionally to the added capacity. Scalability has been a significant issue in parallel, distributed, cluster, grid, networked, and cloud computing systems. In [21], elastic scaling strategies are divided into three categories: (1) scale-in and scale-out-strategies which allow adding more homogeneous machine instances or processing nodes of the same type based on the agreed service-level agreement; (2) scale-up and scale-down—strategies which are implemented by using more powerful machine instances or processing nodes with faster processors/cores and more memory and storage; (3) mixed scaling—strategies which allow one to scale up (or scaled own) and scale-out (or scale-in) computing resources in terms of quantity and quality at the same time. In [19], scale-in and scale-out are called horizontal scalability, and scale-up and scale-down are called vertical scalability. In [27], it was mentioned that scalability includes application scalability (i.e., a property which means that an application maintains its performance goals and service-level agreement even when its workload increases) and platform scalability (i.e., the ability of a cloud platform to provide as many resources as needed by an application). In [28], the technique of using workload dependent dynamic power management (i.e., variable power and speed of processor cores according to the current workload, which is essentially vertical scalability) to improve system performance and to reduce energy consumption is investigated by using a queueing model.

4 Analytical Model and Method

In this section, we present our analytical model and method to compute the proposed elasticity value.

4.1 A Queueing Model

A cloud computing platform is a multiserver system which has m identical servers (i.e., VMs). In this paper, a multiserver system is treated as an M/M/m queueing system which is elaborated as follows [26]. There is a Poisson stream of service requests (i.e., tasks) with arrival rate λ (measured by the number of service requests that are submitted in one unit of time), i.e., the inter-arrival times are independent and identically distributed (i.i.d.) exponential random variables with mean $1/\lambda$. A multiserver system maintains a queue with infinite capacity for waiting tasks when all the m servers are busy. The first-come-first-served (FCFS) queueing discipline is adopted. The task execution times are i.i.d. exponential random variables with mean $1/\mu$. The m servers are homogeneous and have identical execution and service rate μ (measured by the number of tasks that can be finished in one unit of time).

Notice that in an elastic cloud computing platform, the number of servers adapts to the current workload (i.e., the number of tasks in the system). Therefore, we have a multiserver queuing system with a variable number of servers, and an elastic cloud computing platform is no longer an M/M/m queueing system. In [4], the authors dealt with a multiserver retrial queueing model in which the number of active servers depends on the number of customers in the system. The servers are switched on and off according to a multithreshold strategy. For a fixed choice of the threshold levels, the stationary distribution and various performance measures of the system are calculated. In [23], a multiserver Poisson queuing system with losses and a variable number of servers was investigated, and all major characteristics of the system were obtained in an explicit form. Unfortunately, these results are not directly applicable to elastic cloud computing systems, because the times to turn on and off the servers are not considered. However, as mentioned before, these factors are critical in measuring elasticity, and must be included into our queueing model.

4.2 Auto-Scaling Scheme

We use (m, k) to denote a *state*, where $m \geq 1$ is the number of active servers, and $k \geq 0$ is the number of tasks in the system. Let (a_m, b_m), $m \geq 1$, be a pair of integers used to determine the status of a state, where $b_m > a_m \geq m-1$, $a_{m+1} \leq b_m$, for all $m \geq 1$, and $a_1 < a_2 < a_3 < \cdots$, $b_1 < b_2 < b_3 < \cdots$. An elastic cloud platform management and auto-scaling scheme can be represented as

$$S = ((a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m), \ldots), \tag{3}$$

which decides how a cloud computing platform responds to the workload change. States are classified into three types.

- A state is an *over-provisioning* state if $0 \leq k \leq a_m$.
- A state is a *normal* state if $a_m < k \leq b_m$.
- A state is an *under-provisioning* state if $k > b_m$.

The number of a servers can be adjusted according to the status of the state. In particular, a new server can be added (i.e., a cloud server system is scaled-out) if the current state is under-provisioning, and an active server can be removed (i.e., a cloud server system is scaled-in) if the current state is over-provisioning.

4.3 A Continuous-Time Markov Chain

To take the virtual machine start-up and shut-down times into consideration, we make the following assumptions. (1) A new server can be added as an active server at any time, and the time to initialize a new server is an exponential random variable with mean $1/\alpha$ (i.e., the VM start-up rate is α, measured by the number of VMs which can be initialized in one unit of time). (2) An active server can be removed at any time, and the time to finalize an active server is an exponential random variable with mean $1/\beta$ (i.e., the VM shut-down rate is β, measured by the number of VMs which can be finalized in one unit of time).

Based on the above assumptions, it is clear that a multiserver system with variable and dynamically adjustable number of servers can be modeled by a continuous-time Markov chain (CTMC).

Our CTMC is actually a mixture of the birth-death processes similar to those for M/M/m queueing systems, with $m \geq 1$. The transitions among the states are described as follows. (Note: We use the notation $(m_1, k_1) \stackrel{r}{\rightarrow} (m_2, k_2)$ to represent a transition from state (m_1, k_1) to state (m_2, k_2) with transition rate r.)

- $(m, k) \stackrel{\lambda}{\rightarrow} (m, k+1)$, $m \geq 1$, $k \geq 0$. This transition happens when a new task arrives.
- $(m, k) \stackrel{m\mu}{\rightarrow} (m, k-1)$, $m \geq 1$, $k > a_m$. This transition happens when a task is completed, and the state (m, k) is normal or under-provisioning.
- $(m, k) \stackrel{\min(m-1,k)\mu}{\rightarrow} (m, k-1)$, $m \geq 1$, $1 \leq k \leq a_m$. This transition happens when a task is completed, and the state (m, k) is over-provisioning. (The value $m-1$ means that a server is being shut down and not serving, but is still in the system. A deactivated server is also a resource until it is removed from the system.)

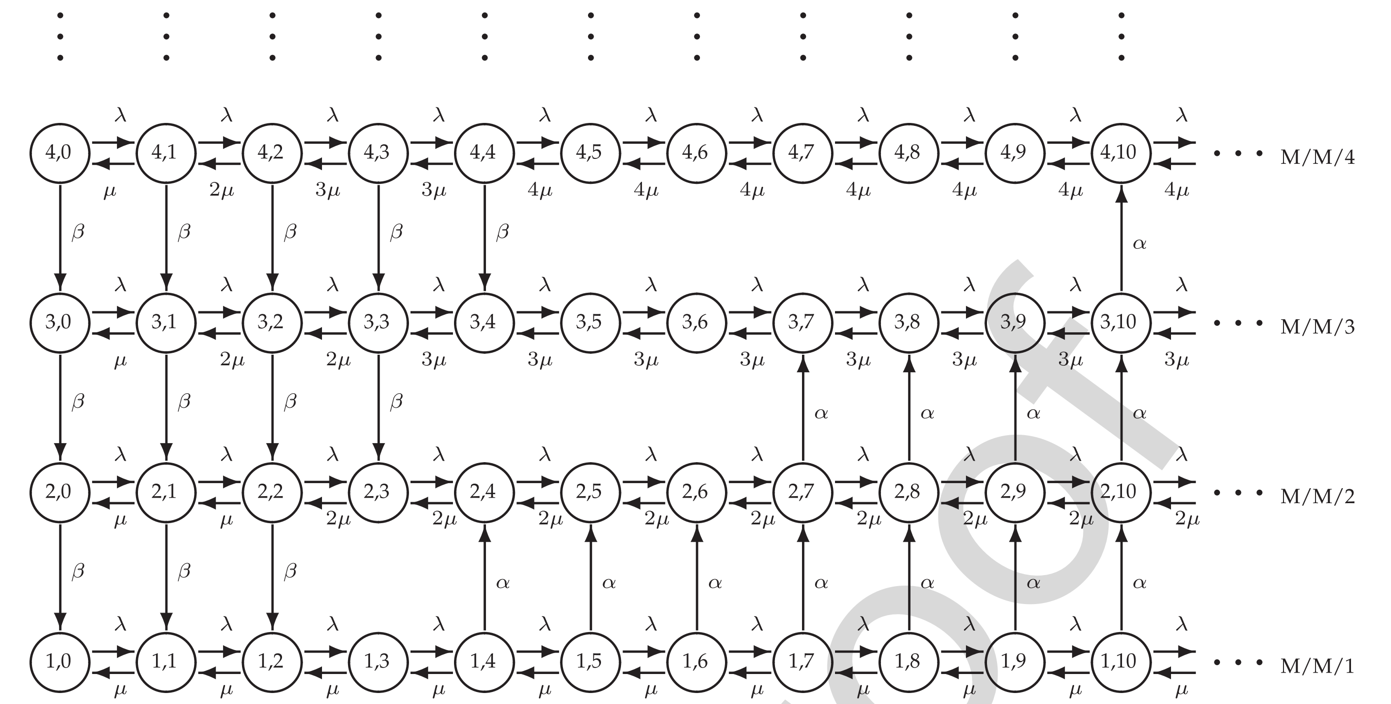

Fig. 1. A state-transition-rate diagram.

- $(m,k) \xrightarrow{\alpha} (m+1,k)$, $m \geq 1$, $k > b_m$. This transition happens when the state (m,k) is under-provisioning, and a new server is activated to join service.
- $(m,k) \xrightarrow{\beta} (m-1,k)$, $m \geq 2$, $1 \leq k \leq a_m$. This transition happens when the state (m,k) is over-provisioning, and an active server is being shut down and removed from further service.

Fig. 1 shows a state-transition-rate diagram, assuming that $a_m = m$ and $b_m = 3m$ for all $m \geq 1$. The states in the diagram are arranged in a two dimensional way, where each row of states is similar to the state-transition-rate diagram of an M/M/m queueing system, with the difference that the number of servers is $m-1$ (not m) when $m-1 \leq k \leq a_m$ due to the VM which is being shut down. Notice that in a state (m,k) where $k \geq b_m + 1$, a new VM is activated and initialized, where the start-up time is an exponential random variable. It is possible that before the initialization is completed, a task arrives or departs, and the state becomes $(m, k \pm 1)$. Since the residual start-up time has the same distribution as the original exponential distribution due to the memoryless property, the transition rate from $(m, k \pm 1)$ to $(m+1, k \pm 1)$ is still α. Similarly, in a state (m,k) where $k \leq a_m$, one VM is deactivated and finalized, where the shut-down time is an exponential random variable. It is possible that before the finalization is completed, a task arrives or departs, and the state becomes $(m, k \pm 1)$. Due to the memoryless property, the transition rate from $(m, k \pm 1)$ to $(m-1, k \pm 1)$ is still β.

To summarize, our CTMC model for an elastic cloud computing system with variable number of virtual machines contains the following parameters: λ, μ, α, β, and of course, S. It is worth to mention that the purpose of our research is to capture the most essential parameters for elasticity quantification and prediction. Our model and method are by no means perfect, but only some initial attempt towards this direction. In a real cloud platform, things can be much more complicated. First, there could be many components in resource management, such as physical machines, storage, and network resources. Second, there could be many factors (other than VM start-up and shut-down times) which affect VM creation and termination. However, it is clear that considering all these factors and facts might result in infeasible modeling and analysis, although they could be included and considered in further investigation. For the purpose of feasible modeling and analysis, our abstract model and analytical method are simplistic and manageable.

4.4 An Analytical and Numerical Method

Let $p(m,k)$ denote the equilibrium steady-state probability that a multiserver system is in state (m,k). Unfortunately, there is no closed-form expression of $p(m,k)$. However, a numerical solution can be easily obtained by solving a linear system of equations resulted from our CTMC model using any standard method from linear algebra.

Once the $p(m,k)$'s are available, we can compute the elasticity metric as follows. The probability that the system is in the over-provisioning state is

$$p_{\text{over}} = \sum_{m=1}^{\infty} \sum_{k=0}^{a_m} p(m,k). \tag{4}$$

The probability that the system is in the under-provisioning state is

$$p_{\text{under}} = \sum_{m=1}^{\infty} \sum_{k=b_m+1}^{\infty} p(m,k). \tag{5}$$

The probability that the system is in the normal state is

$$p_{\text{normal}} = \sum_{m=1}^{\infty} \sum_{k=a_m+1}^{b_m} p(m,k). \tag{6}$$

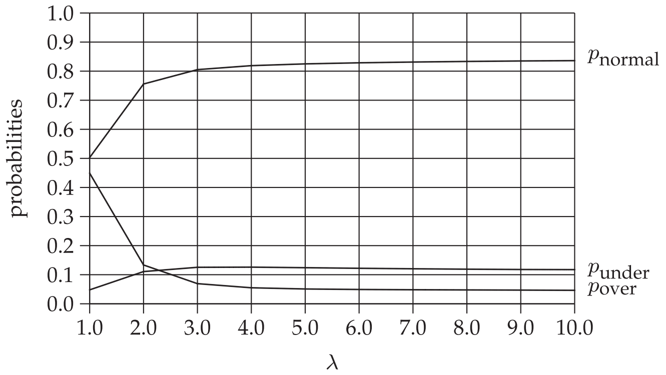

Fig. 2. p_{over}, p_{normal}, and p_{under} versus λ.

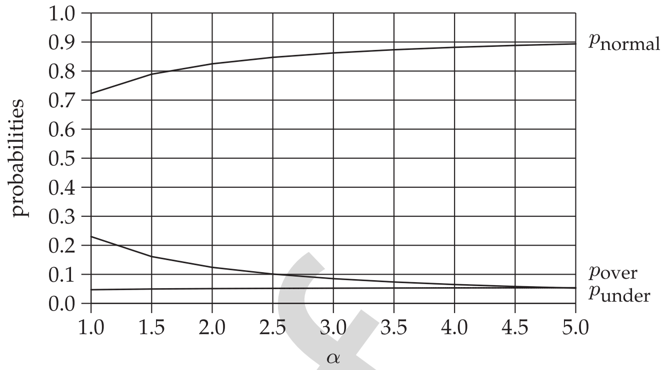

Fig. 4. p_{over}, p_{normal}, and p_{under} versus α.

Based on the above probabilities, our elasticity metric can be obtained by using Eq. (2).

4.5 Impact of the Basic Parameters

It is clear that by using the CTMC model to calculate the elasticity value of a cloud platform, our elasticity metric is determined by only a few parameters, namely, the task arrival rate, the service rate, the virtual machine start-up and shut-down rates, and the scaling scheme. In this section, we present numerical data to demonstrate the impact of these basic parameters on elasticity.

In Figs. 2, 3, 4, and 5, we assume that $a_m = m$ and $b_m = 3m$ for all $m \geq 1$.

Varying the Task Arrival Rate. In Fig. 2, we show p_{over}, p_{normal}, and p_{under} as functions of the task arrival rate λ, where $\mu = 1$, $\alpha = 2$, $\beta = 5$, and $\lambda = 1.0, 2.0, \ldots, 10.0$. It is observed that as λ increases, p_{over} decreases (i.e., more service requests result in less probability of over-provisioning), and p_{under} changes slightly (actually, increases and then decreases, i.e., more service requests result in slight change of the probability of under-provisioning), and p_{normal} increases (i.e., the elasticity increases).

Varying the Service Rate. In Fig. 3, we show p_{over}, p_{normal}, and p_{under} as functions of the task service rate μ, where $\lambda = 5$, $\alpha = 2$, $\beta = 5$, and $\mu = 1.0, 2.0, \ldots, 10.0$. It is observed that as μ increases, p_{over} increases significantly (i.e., faster service rate results in greater probability of over-provisioning), and p_{under} changes noticeably (actually, increases and then decreases, i.e., faster service rate results in noticeable change of the probability of under-provisioning), and p_{normal} decreases significantly (i.e., the elasticity decreases significantly).

Varying the Virtual Machine Start-Up Rate. In Fig. 4, we show p_{over}, p_{normal}, and p_{under} as functions of the virtual machine start-up rate α, where $\lambda = 5$, $\mu = 1$, $\beta = 5$, and $\alpha = 1.0, 1.5, \ldots, 5.0$. It is observed that as α increases, p_{over} increases slightly (i.e., faster virtual machine start-up rate results in greater probability of over-provisioning), and p_{under} decreases noticeably (i.e., faster virtual machine start-up rate results in noticeable reduction of the probability of under-provisioning), and p_{normal} increases noticeably (i.e., the elasticity increases noticeably).

Varying the Virtual Machine Shut-Down Rate. In Fig. 5, we show p_{over}, p_{normal}, and p_{under} as functions of the virtual machine shut-down rate β, where $\lambda = 5$, $\mu = 1$, $\alpha = 2$, and $\beta = 5.0, 5.5, \ldots, 10.0$. It is observed that the impact of β is small. As β increases, p_{over} decreases slightly (i.e., faster virtual machine shut-down rate results in less probability of over-provisioning), and p_{under} increases slightly (i.e., faster virtual machine shut-down rate results in greater probability of under-provisioning), and p_{normal} increases slightly (i.e., the elasticity increases slightly).

Varying the Scaling Scheme. In Fig. 6, we show p_{over}, p_{normal}, and p_{under} as functions of x, where $\lambda = 5$, $\mu = 1$, $\alpha = 2$, $\beta = 5$, $a_m = m$, and $b_m = a_m + x$, for all $m \geq 1$. It is observed that the impact of the scaling scheme is big. As x increases (i.e., the interval $[a_m, b_m]$ gets wider), both p_{over} and p_{under} decrease noticeably (i.e., wider interval $[a_m, b_m]$ results in less probability of over-provisioning and under-provisioning), and p_{normal} increases significantly (i.e., the elasticity increases significantly).

It is worth to mention that the purpose of this section is to demonstrate the impact of some basic parameters on elasticity. These data are obtained based on our model and

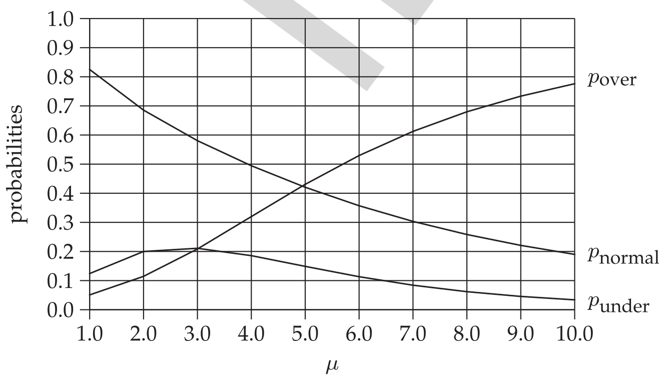

Fig. 3. p_{over}, p_{normal}, and p_{under} versus μ.

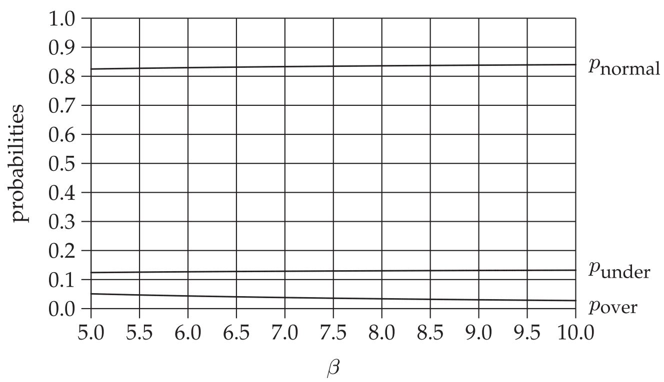

Fig. 5. p_{over}, p_{normal}, and p_{under} versus β.

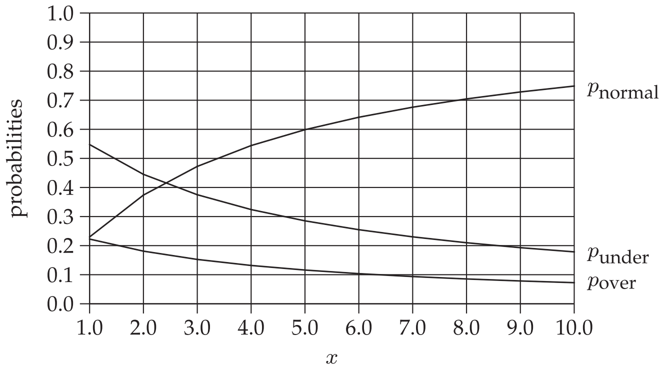

Fig. 6. p_{over}, p_{normal}, and p_{under} versus x ($b_m = a_m + x$).

method, and might not be entirely accurate for any real world use case scenario.

4.6 Simulation Results: Accuracy and Robustness

To validate the accuracy and robustness of our CTMC model, we have performed extensive simulations and experiments. Our simulation environment is an Intel Xeon CPU E5620 2.40 GHz with the Linux OS version RHEL 6.8. The simulation program is written in C++ supported by the g++ 4.4.7 compiler. We simulate an elastic cloud computing platform with $a_m = m$ and $b_m = 3m$ for all $m \geq 1$, and $\lambda = 5$, $\alpha = 2$, $\beta = 5$, and $\mu = 1.0, 2.0, \ldots, 10.0$. We (1) generate a Poisson stream of service requests; (2) run the elastic cloud computing system; (3) record T_{over}, T_{normal}, and T_{under}; (4) and report $p_{\text{over}} = T_{\text{over}}/T$, $p_{\text{normal}} = T_{\text{normal}}/T$, and $p_{\text{under}} = T_{\text{under}}/T$, where $T = T_{\text{normal}} + T_{\text{over}} + T_{\text{under}}$, until 1,000,000 service requests are completed.

In addition to the exponential distribution of task execution times, we also consider several other distributions. The six probability distribution functions (pdf), all with the same expectation $1/\mu$, are described as follows.

- Exponential distribution (EXP): The pdf is $\mu e^{-\mu x}$.
- Hyperexponential distribution (HEX): The pdf is $w_1\mu_1 e^{-\mu_1 x} + w_2\mu_2 e^{-\mu_2 x} + w_3\mu_3 e^{-\mu_3 x}$, where $w_1 = 0.2$, $w_2 = 0.3$, $w_3 = 0.5$, $\mu_1 = y_1\mu'$, $\mu_2 = y_2\mu'$, $\mu_3 = y_3\mu'$, $y_1 = 3$, $y_1 = 2$, $y_1 = 1$, with $\mu' = \mu(w_1/y_1 + w_2/y_2 + w_3/y_3)$.
- Erlang distribution (ERL): The pdf is $\mu' e^{-\mu' x}(\mu' x)^{\gamma-1}/(\gamma-1)!$, where $\mu' = \gamma\mu$ and $\gamma = 5$.
- Hyper-Erlang distribution (HER): The pdf is $w_1\mu_1 e^{-\mu_1 x}(\mu_1 x)^{\gamma_1-1}/(\gamma_1-1)! + w_2\mu_2 e^{-\mu_2 x}(\mu_2 x)^{\gamma_2-1}/(\gamma_2-1)!$, where $w_1 = 0.4$, $w_2 = 0.6$, $\gamma_1 = 3$, and $\gamma_2 = 4$.
- Uniform distribution (UNI): The pdf is $(\mu/2)$ in the range $[0, 2/\mu)$.
- Pareto distribution (PAR): The pdf is $\alpha\beta^\alpha/x^{\alpha+1}$ in the range $[\beta, \infty)$, where $\alpha = 2$ and $\beta = (\alpha-1)/(\alpha\mu)$.

In Table 2, we show p_{over}, p_{normal}, and p_{under} as functions of the task service rate μ, for all the above six probability distribution functions of task execution times, as well as the analytical results of our CTMC model. We have the following important observations. (1) Accuracy—The simulation results for the exponential distribution are very close to the analytical results and validate the accuracy of our CTMC model. (2) Robustness—The simulation results for the hyperexponential distribution, Erlang distribution, hyper-Erlang distribution, uniform distribution, and Pareto distribution, especially the results of p_{normal}, show the robustness of our CTMC model, i.e., the ability of the CTMC model to predict the elasticity E with reasonable accuracy even though some assumptions of our model are not satisfied.

TABLE 2
Simulation Results

μ	ANA	EXP	HEX	ERL	HER	UNI	PAR
				p_{over}			
1.0	0.05087	0.05191	0.05446	0.04341	0.04496	0.04519	0.04737
2.0	0.11458	0.12285	0.12707	0.10075	0.10337	0.10558	0.11159
3.0	0.20854	0.22145	0.22754	0.18933	0.19552	0.19574	0.21464
4.0	0.31983	0.33381	0.34003	0.30715	0.31158	0.30986	0.33752
5.0	0.43061	0.44184	0.44643	0.43138	0.43428	0.42806	0.46350
6.0	0.52955	0.53843	0.54011	0.54501	0.54548	0.53813	0.56954
7.0	0.61252	0.61893	0.61719	0.64157	0.63837	0.63090	0.65286
8.0	0.67979	0.68557	0.67847	0.71666	0.71034	0.70516	0.71685
9.0	0.73348	0.73698	0.72915	0.77327	0.76561	0.76218	0.76773
10.0	0.77617	0.77958	0.77202	0.81703	0.80887	0.80432	0.80550
				p_{normal}			
1.0	0.82503	0.82194	0.81342	0.84834	0.84524	0.84360	0.83649
2.0	0.68546	0.67416	0.66380	0.71782	0.71005	0.70516	0.69784
3.0	0.58055	0.56483	0.55639	0.61378	0.60477	0.60142	0.58623
4.0	0.49450	0.48026	0.47082	0.52304	0.51577	0.51627	0.49342
5.0	0.42053	0.40905	0.40158	0.44179	0.43542	0.43730	0.40807
6.0	0.35708	0.34763	0.34207	0.36755	0.36436	0.36858	0.33649
7.0	0.30332	0.29658	0.29301	0.30213	0.30112	0.30669	0.27723
8.0	0.25830	0.25221	0.25331	0.24687	0.24965	0.25347	0.23043
9.0	0.22091	0.21723	0.21932	0.20324	0.20737	0.21044	0.19283
10.0	0.18997	0.18655	0.18890	0.16780	0.17318	0.17688	0.16356
				p_{normal}			
1.0	0.12410	0.12615	0.13212	0.10825	0.10980	0.11120	0.11614
2.0	0.19996	0.20298	0.20912	0.18143	0.18658	0.18926	0.19056
3.0	0.21091	0.21372	0.21607	0.19689	0.19971	0.20284	0.19913
4.0	0.18567	0.18593	0.18916	0.16981	0.17265	0.17387	0.16907
5.0	0.14886	0.14911	0.15199	0.12683	0.13030	0.13464	0.12843
6.0	0.11337	0.11394	0.11781	0.08743	0.09016	0.09329	0.09397
7.0	0.08416	0.08449	0.08980	0.05630	0.06051	0.06241	0.06991
8.0	0.06192	0.06223	0.06822	0.03647	0.04000	0.04137	0.05273
9.0	0.04562	0.04579	0.05152	0.02349	0.02702	0.02738	0.03944
10.0	0.03386	0.03387	0.03907	0.01517	0.01796	0.01880	0.03095

4.7 Extension of the CTMC Model

The CTMC model can be extended to include more complicated scaling schemes.

Hot, Warm, and Cold VMs. It is known that physical machines (PMs) are categorized into three server pools: hot (i.e., with running VMs), warm (i.e., turned on but without running VM), and cold (i.e., turned off) [24]. Therefore, VMs can also be classified into three categories: hot (currently running), warm (to be started up from a warm PM), and cold (to be started up from a cold PM). It is clear that a warm VM takes much less time to start than a cold VM. Let us assume that a cloud platform keeps certain number m^* of hot and warm VMs and unlimited cold VMs. The warm VM and cold VM start-up rates are α_1 and α_2 respectively, where $\alpha_1 > \alpha_2$. Then, we should have $(m,k) \xrightarrow{\alpha_1} (m+1,k)$, for $1 \leq m < m^*$ and $k > b_m$, and $(m,k) \xrightarrow{\alpha_2} (m+1,k)$, for $m \geq m^*$ and $k > b_m$. That is, the first m^* VMs can be started up faster than the remaining VMs.

Multiple Start-Up and Shut-Down. In our CTMC model in Section 4.3, it is assumed that VM start-up's take place sequentially, i.e., one after another. In state (m, k) where $k > b_m$, there is only one VM being started up, no matter how big k is. Actually, when a platform detects that k is sufficiently large (say, k^*), i.e., a VM takes too long to start up, another VM can be started up simultaneously to handle increasing workload. Therefore, we should have $(m, k) \xrightarrow{\alpha} (m+1, k)$, for $m \geq 1$ and $b_m < k < k^*$, and $(m, k) \xrightarrow{2\alpha} (m+1, k)$, for $m \geq 1$ and $k \geq k^*$. Notice that due to the memoryless property, the residual start-up time of the first VM has the same distribution as the original exponential distribution. Thus, the combined transition rate from (m, k) to $(m+1, k)$ is now 2α. It is clear that this method can be extended to arbitrary simultaneous start-up's. Also, it can be applied to multiple shut-down's when k is sufficiently small.

Minimum Number of Active VMs. In our CTMC model in Section 4.3, it is assumed that the number of active VMs can be as small as one. To ensure certain guaranteed performance, a platform can maintain a minimum number (say, m^*) of active VMs (which is one in Fig. 1). One can simply assume that $a_m = -1$ for this purpose, where $1 \leq m \leq m^*$, i.e., there is no over-provisioning state and thus no VM shutdown when the number of active VMs is no more than m^*.

Heterogeneous VMs. Assume that there are n types of VMs with service rates $\mu_1, \mu_2, \ldots, \mu_n$, start-up rates $\alpha_1, \alpha_2, \ldots, \alpha_n$, and shut-down rates $\beta_1, \beta_2, \ldots, \beta_n$. A state can be described as $(m_1, m_2, \ldots, m_n, k)$, where m_i is the number of VMs of type i, $1 \leq i \leq n$. Hence, we will typically have a transition like $(m_1, m_2, \ldots, m_n, k) \xrightarrow{m_1\mu_1+m_2\mu_2+\cdots+m_n\mu_n} (m_1, m_2, \ldots, m_n, k-1)$. For an under-provisioning state, if a VM of type i is to be activated, we have $(m_1, \ldots, m_i, \ldots, m_n, k) \xrightarrow{\alpha_i} (m_1, \ldots, m_i + 1, \ldots, m_n, k)$. For an over-provisioning state, if a VM of type i is to be deactivated, we have $(m_1, \ldots, m_i, \ldots, m_n, k) \xrightarrow{\beta_i} (m_1, \ldots, m_i - 1, \ldots, m_n, k)$.

5 PERFORMANCE AND COST METRICS

Several important performance and cost metrics can be easily obtained as by-products from our model and method.

5.1 Performance Metrics

The main performance metrics are average task response time, throughput, and quality of service.

Average Number of Requests. The average number N of tasks in a multiserver system, including tasks being served and tasks in the waiting queue, can be calculated by

$$N = \sum_{m=1}^{\infty}\sum_{k=1}^{\infty} kp(m, k) = \sum_{k=1}^{\infty} k\left(\sum_{m=1}^{\infty} p(m, k)\right). \quad (7)$$

Average Task Response Time. The response time of a task includes its waiting time and service time. By Little's result, the average task response time is

$$T = \frac{N}{\lambda}. \quad (8)$$

Throughput. Throughput is the average number of tasks completed per unit of time. It is clear that in any stable service system, the throughput R, i.e., the output, should be the same as the input, i.e., λ, the average number of tasks submitted per unit of time. Thus, we have

$$R = \lambda. \quad (9)$$

Quality of Service (QoS). QoS metrics for cloud computing can be focused on various aspects of cloud services, such as performance, economics, security, and general features [3], [6]. Therefore, QoS can be defined in many different ways. In this paper, we will mainly focus on performance metrics, and in particular, we use the reciprocal of the average task response time $1/T$ as the QoS index

$$\text{QoS} = \frac{1}{T}, \quad (10)$$

which is readily available from our model and method.

It is worth to mention that in a real cloud platform, there could be many factors which affect performance metrics, such as the impact of network resources on the average task response time. Again, considering all these factors is beyond the scope of this paper.

5.2 Cost Metrics

The main cost metric is the average number of VMs, which is directly related to the amount of charge to a customer.

Average Number of VMs. The number m of servers is a random variable in an elastic cloud computing platform. The average number $M = \bar{m}$ (i.e., the expectation of m) of servers, including busy servers, idle servers, and the one being shut down, is given by

$$M = \sum_{m=1}^{\infty} m\left(\sum_{k=0}^{\infty} p(m, k)\right). \quad (11)$$

Average Number of Busy VMs. The average number B of busy servers only includes servers in service, not idle servers and the one being shut down, and is given by

$$B = \sum_{m=1}^{\infty}\left(\sum_{k=0}^{a_m} \min(m-1, k)p(m, k) + \sum_{k=a_m+1}^{\infty} mp(m, k)\right). \quad (12)$$

From another point of view, B is actually the total amount of work finished in one unit of time, i.e., λ/μ. To see this, let $b(t)$ be the number of busy servers at time t. During a time interval $[t_1, t_2]$, the amount of completed work (measured in time) is $\int_{t_1}^{t_2} b(x)dx$. On the other hand, the amount of submitted work is $(t_2 - t_1)\frac{\lambda}{\mu}$. In a stable service system, we must have $\int_{t_1}^{t_2} b(x)dx = (t_2 - t_1)\frac{\lambda}{\mu}$. Furthermore, it is clear that the average number of busy servers is $B = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2} b(x)dx$. Thus, we have

$$B = \frac{\lambda}{\mu}. \quad (13)$$

Utilization. The VM utilization U is the ratio of the average number of busy VMs to the average number of VMs, i.e.,

$$U = \frac{B}{M} = \frac{\lambda}{M\mu}. \tag{14}$$

Cost. There are many different factors which determine the cost of cloud computing. It is clear that the cost of a cloud platform is linearly proportional to the average number M of VMs. For each VM, the cost includes the renting cost and energy consumption cost [9]. Therefore, in this paper, we simply use the following equation to calculate the cost of a cloud computing platform

$$\text{cost} = M(\phi + \psi\mu^d), \tag{15}$$

where ϕ includes the renting cost and static power consumption, and $\psi\mu^d$ is the dynamic power consumption that is linearly proportional to a polynomial of the VM speed. In this paper, we assume that $\phi = 10$, $\psi = 1$, and $d = 3$, unless otherwise stated. Since these constants only give scaling effect, sometimes we just use M as the cost.

5.3 Combined Performance and Cost Metrics

The main combined metric is the cost-performance ratio, which can be applied to define other combined metrics.

Cost-Performance Ratio. The cost-performance (or price-performance) ratio (CPR) refers to a product's ability to deliver performance for its price. Generally speaking, products with a lower CPR are more desirable, excluding other factors. It is clear that the cost of a cloud platform is linearly proportional to the average number M of VMs, and that the performance is inversely proportional to the average task response time T. Hence, we can define CPR as

$$\text{CPR} = \text{cost/performance} = MT(\phi + \psi\mu^d). \tag{16}$$

Productivity. In [21], productivity is defined in such a way that it is proportional to performance and QoS, and inversely proportional to cost. If we use throughput R as the performance index, the reciprocal of the average task response time T as the QoS index, and the average number M of VMs as the cost index, then we will have productivity as

$$\text{Productivity} = \text{performance} \times \text{QoS/cost} = \frac{R}{MT}. \tag{17}$$

Production-Driven Scalability. Recall that a cloud platform management and scaling scheme can be represented as $S = ((a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m), \ldots)$. For given λ, μ, α, β, the scaling scheme S will decide all the cost and performance metrics mentioned above, e.g., the productivity. In production-driven scalability [21], a scaling scheme S is more desirable than another scaling scheme $S' = ((a'_1, b'_1), (a'_2, b'_2), \ldots, (a'_m, b'_m), \ldots)$, if the productivity of S is higher than that of S'. Therefore, the production-driven scalability is

$$\text{Scalability}(S, S') = \frac{\text{Productivity}(S)}{\text{Productivity}(S')}, \tag{18}$$

which can also be represented as

$$\text{Scalability}(S, S') = \frac{\text{CPR}(S')}{\text{CPR}(S)}. \tag{19}$$

6 Performance and Cost Guarantee

All rigorous metrics, quantified measures, accurate models, and analytical methods for elasticity should be applied to provide and predict the required service quality and cost to the users. The purposes of this section are two-fold. First, we show how to provide service quality and service cost guarantee to the users. Second, we show that with certain cost, an elastic platform delivers certain performance guarantee with higher probability than an inelastic platform with the same cost for the same performance guarantee.

6.1 Inelastic Platforms with Fixed Servers

Recall that all task execution times are i.i.d. random variables x. We use $\bar{x}$ to denote the expectation of a random variable x. For an M/M/m queueing system modeling an inelastic cloud computing platform with a fixed number of servers, the server utilization is $\rho = \lambda/m\mu = \lambda\bar{x}/m$, which is the average percentage of time that a server is busy. A state of M/M/m is specified by k, the number of service requests (i.e., tasks, waiting or being processed) in the queueing system. Let p_k denote the probability that the M/M/m queueing system is in state k. Then, we have ([26], p. 102)

$$p_k = \begin{cases} p_0 \dfrac{(m\rho)^k}{k!}, & k \le m; \\ p_0 \dfrac{m^m \rho^k}{m!}, & k \ge m; \end{cases}$$

where

$$p_0 = \left(\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!} \cdot \frac{1}{1-\rho} \right)^{-1}.$$

The probability of queueing (i.e., the probability that a newly submitted service request must wait because all servers are busy) is

$$P_q = \sum_{k=m}^{\infty} p_k = \frac{p_m}{1-\rho} = p_0 \frac{(m\rho)^m}{m!} \cdot \frac{1}{1-\rho}.$$

The average number of service requests (in waiting or in execution) is

$$N = \sum_{k=0}^{\infty} k p_k = m\rho + \frac{\rho}{1-\rho} P_q.$$

Applying Little's result, we get the average task response time as

$$T = \frac{N}{\lambda} = \bar{x}\left(1 + \frac{P_q}{m(1-\rho)}\right) = \bar{x}\left(1 + \frac{p_m}{m(1-\rho)^2}\right).$$

Therefore, we get the following result.

Theorem 1. *An inelastic cloud computing platform with fixed number m of servers can guarantee average task response time*

$$T = \bar{x}\left(1 + \frac{p_m}{m(1-\rho)^2}\right),$$

with cost m, and cost-performance ratio

$$\mathrm{CPR} = m\bar{x}\left(1 + \frac{p_m}{m(1-\rho)^2}\right).$$

Let T_k be the average response time under the condition that a new service request arrives when the system is in state k. In other words, we can consider T as a function τ of k and τ is randomized over the states k. When a task arrives to the system which is in state k, the average response time τ of the task takes the value T_k, and the probability to take this value is p_k. Therefore, T is actually the expectation of τ, i.e.,

$$T = \bar{\tau} = \sum_{k=0}^{\infty} p_k T_k.$$

The following theorem gives a performance guarantee in a stronger way for customers on an inelastic cloud computing platform.

Theorem 2. *For an inelastic cloud computing platform with fixed number m of servers, we have $\tau \le c\bar{x}$, with probability*

$$\sum_{k=0}^{\lfloor cm-1 \rfloor} p_k,$$

where $c > 1$.

Proof. Let W_k be the waiting time of a new service request which arrives when the system is in state k. Then, it is already known from [9] that $\overline{W}_k = 0$ if $0 \le k \le m-1$, and

$$\overline{W}_k = \left(\frac{k-m+1}{m}\right)\bar{x},$$

if $k \ge m$. Since $T_k = \overline{W}_k + \bar{x}$, we get $T_k = \bar{x}$ if $0 \le k \le m-1$, and

$$T_k = \left(\frac{k+1}{m}\right)\bar{x},$$

if $k \ge m$. To have $T_k \le c\bar{x}$, we need $k \le cm - 1$. Since $\tau = T_k$ with probability p_k, the theorem is proven. □

An immediate consequence of Theorem 2 is that $\tau > c\bar{x}$ with probability $\sum_{k=\lfloor cm \rfloor}^{\infty} p_k$. One significance of Theorem 2 is that a cloud service provider can claim to its users that the average task response time is bounded by a constant times the expected task execution time with certain probability. Notice that for a random variable x, a claim such as "x is less than c with high probability" is stronger than "$\bar{x}$ is less than d", even if c is reasonably greater than d.

6.2 Elastic Platforms with Variable Servers

Now we consider an elastic cloud computing platform with variable number of servers. By combining Eqs. (7), (8), and (11), we get the following result.

Theorem 3. *An elastic cloud computing platform with variable number of servers can guarantee average task response time*

$$T = \frac{1}{\lambda}\sum_{m=1}^{\infty}\sum_{k=1}^{\infty} kp(m,k),$$

and expected cost

$$M = \sum_{m=1}^{\infty} m\left(\sum_{k=0}^{\infty} p(m,k)\right),$$

and cost-performance ratio

$$\mathrm{CPR} = \frac{1}{\lambda}\left(\sum_{m=1}^{\infty}\sum_{k=1}^{\infty} kp(m,k)\right)\left(\sum_{m=1}^{\infty} m\left(\sum_{k=0}^{\infty} p(m,k)\right)\right).$$

Again, let $T(m,k)$ be the average response time under the condition that a new service request arrives when the system is in state (m,k). We treat T as a function τ of (m,k) and τ is randomized over the states (m,k).

The following theorem gives a performance guarantee in a stronger way for customers on an elastic cloud computing platform.

Theorem 4. *For an elastic cloud computing platform with variable number of servers, we have $\tau \le c\bar{x}$, where $c > 1$, with probability at least*

$$p_{\mathrm{normal}} + p_{\mathrm{over}} = \sum_{m=1}^{\infty}\sum_{k=0}^{\lfloor cm-1 \rfloor} p(m,k),$$

by setting $a_m = m-1$ and $b_m = \lfloor cm-1 \rfloor$.

Proof. Consider a task submitted to a cloud platform with m servers and k tasks in the system. We notice that the variable number of servers makes the analysis of waiting time much more complicated. First, it is possible that after a task x arrives, future arrival tasks may cause the system entering an under-provisioning state and creating more servers. Fortunately, such change will simply reduce the waiting time of x, which does no affect the upper bound $c\bar{x}$ in the theorem, that is derived based on the assumption that the number of servers does not increase as in M/M/m. Second, it is also possible that after a task x arrives, completed tasks may cause the system entering an over-provisioning state and removing servers. Fortunately, the assumption that $a_m = m-1$ means that a server is removed only when there is no more task in waiting, i.e., x is already in execution and its waiting time is not affected. Therefore, we will simply ignore the possible changes in the number of servers.

We follow an argument similar to that in the proof of Theorem 2. When $m = 1$, the number of active servers is always one. Thus, we get $T(1,k) = (k+1)\bar{x}$ for all $k \ge 0$. When $m > 1$, the number of active servers is $m-1$ if $0 \le k \le a_m$, and m if $a_m < k \le b_m$. Thus, we have $T(m,k) = \bar{x}$ if $0 \le k \le m-1$, and

$$T(m,k) = \left(\frac{k+1}{m}\right)\bar{x} \le \left(\frac{b_m+1}{m}\right)\bar{x},$$

if $m \le k \le b_m$. Hence, the above cases of $T(m,k)$ can be combined into

$$T(m,k) \le \left(\frac{b_m+1}{m}\right)\bar{x},$$

TABLE 3
Comparison of Elastic and Inelastic Platforms

c	p_{over}	p_{normal}	p_{under}	probability	T	M	cost	CPR
elastic platform								
1.25	0.18542	0.26410	0.55048	0.44952	1.37467	10.89474	119.842	164.743
1.50	0.13794	0.46373	0.39833	0.60167	1.44754	10.78981	118.688	171.806
1.75	0.10992	0.58292	0.30716	0.69284	1.52815	10.72906	118.020	180.352
2.00	0.08648	0.67675	0.23678	0.76322	1.64230	10.67876	117.466	192.915
2.25	0.07316	0.72918	0.19766	0.80234	1.73454	10.65005	117.151	203.203
2.50	0.06103	0.77702	0.16195	0.83805	1.85590	10.62435	116.868	216.895
2.75	0.05233	0.81112	0.13655	0.86345	1.97080	10.60592	116.665	229.924
3.00	0.04447	0.84061	0.11492	0.88508	2.11404	10.58911	116.480	246.244
inelastic platform								
1.25	–	–	–	0.24109	2.66581	11.00000	121.000	322.563
1.50	–	–	–	0.33995	2.66581	11.00000	121.000	322.563
1.75	–	–	–	0.42592	2.66581	11.00000	121.000	322.563
2.00	–	–	–	0.50070	2.66581	11.00000	121.000	322.563
2.25	–	–	–	0.54506	2.66581	11.00000	121.000	322.563
2.50	–	–	–	0.60432	2.66581	11.00000	121.000	322.563
2.75	–	–	–	0.65586	2.66581	11.00000	121.000	322.563
3.00	–	–	–	0.70069	2.66581	11.00000	121.000	322.563

for all $m \geq 1$ and $0 \leq k \leq b_m$. To have $T(m,k) \leq c\bar{x}$, we need $b_m = cm - 1$ (actually $b_m = \lfloor cm - 1 \rfloor$ to have an integer). Since $\tau = T(m,k)$ with probability $p(m,k)$, the theorem is proven. □

One significant implication of Theorem 4 is that the average task response time is well bounded as long as a cloud computing platform is not in the under-provisioning state. In particular, the inequality of the theorem holds with probability at least $1 - p_{\text{under}}$, which is greater than E.

By using Theorem 4, a cloud service provider can claim to a customer that the expected task response time is no more than $c\bar{x}$ for some small constant $c > 1$ with probability higher than E, by appropriate design of the elastic scaling scheme. Furthermore, the cloud service provider can tell the customer the estimated cost based on Eq. (15).

6.3 Comparison

In this section, we show that with certain cost, an elastic platform delivers certain performance guarantee with higher probability than an inelastic platform with the same cost for the same performance guarantee. Furthermore, an elastic platform is able to achieve higher QoS by consuming less resources than an inelastic platform, and thus achieving lower CPR, higher productivity, and dual improvement of both performance and cost.

Let us assume that $\lambda = 10.5$, $\mu = 1$, $\alpha = 2$, $\beta = 5$, $a_m = m - 1$, and $b_m = \lfloor cm - 1 \rfloor$, for all $m \geq 1$. For $c = 1.25, 1.50, \ldots, 3.00$, we show p_{over}, p_{normal}, p_{under}, the probability in Theorem 4, T, M, cost, and CPR for an elastic platform in Table 3. It is observed that as c increases, both p_{over} and p_{under} decrease significantly, and p_{normal} (i.e., elasticity E) increases significantly. Furthermore, the probability $p_{\text{normal}} + p_{\text{over}}$ in Theorem 4 increases significantly. However, such increased elasticity is due to the increased b_m, which actually degrades system performance, since the platform is less responsive to the increased workload. As expected, the average task response time increases noticeably, and the average number of VMs and the cost reduce slightly. However, the cost-performance ratio increases significantly.

By letting $m = 11$, $\lambda = 10.5$, $\mu = 1$, and for $c = 1.25, 1.50, \ldots, 3.00$, we also show the probability in Theorem 2, T, M, cost, and CPR for an inelastic platform in Table 3. It is observed that for the same c, the elastic platform with M less than that of m of the inelastic platform, achieves significantly shorter average task response time, provides the same performance guarantee with noticeably higher probability, and has less cost and much lower cost-performance ratio.

7 Cost-Performance Ratio Optimization

As mentioned earlier, the ultimate purpose of studying elasticity is not just to measure elasticity quantitatively and analytically, but for a cloud service provider to construct and manage an elastic cloud computing platform to serve users better in terms of higher performance and lower cost. The purposes of this section are three-fold. First, we discuss one important issue, i.e., comparison of scaling schemes and optimal design of an elastic scaling scheme to minimize the CPR. Second, we show how to optimize a cloud computing platform, such that the CPR is minimized. Third, we mention how to compare different platforms from different service providers.

7.1 Optimization of Scaling Schemes

In this section, we first consider the following problem. For a given application and system environment specified by λ, μ, α, β, how to compare two different elastic scaling schemes S and S'. Our approach is to compare the CPR(S) and CPR(S') of the two schemes. If CPR(S) is less than CPR(S'), then S is better than S', since the production-driven scalability is CPR(S')/CPR(S) > 1 (see Eq. (19)).

An elastic cloud platform management and auto-scaling scheme $S = ((a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m), \ldots)$ can be manipulated. For instance, one can decrease a_m or increase b_m to increase the value of elasticity. However, doing so increases the number of VMs, or increases the task response time and reduces the quality of service. On the other hand, increasing a_m or decreasing b_m not only reduces the value of elasticity, but also increases the task response time, or increases the number of VMs and the cost of service. It is clear that for the minimized T and the best QoS, both a_m and b_m should be minimized, e.g., $a_m = m - 1$ and $b_m = m$. However, the average number M of VMs is maximized.

It is clear that there is trade-off between performance and cost. It is a challenge on how to balance the two conflicting requirements of maximizing quality of service and minimizing cost of service. In this section, we consider the following optimization problem. For a given application and system environment specified by λ, μ, α, β, find an optimal auto-scaling scheme S, such that the cost-performance ratio CPR is minimized.

Let us assume that $\lambda = 7$, $\mu = 1$, $\alpha = 2$, $\beta = 5$, $a_m = m$, and $b_m = a_m + x$, for all $m \geq 1$. For $x = 1, 2, \ldots, 20$, we show p_{over}, p_{normal}, p_{under}, T, M, cost, and CPR for an elastic platform in Table 4. It is observed that as x increases, both p_{over} and p_{under} decrease significantly, and p_{normal} (i.e., elasticity

TABLE 4
Optimal Scaling Scheme

x	p_{over}	p_{normal}	p_{under}	T	M	cost	CPR
1	0.23368	0.17882	0.58750	1.48578	7.36402	81.0042	120.354
2	0.19827	0.30318	0.49854	1.52808	7.30819	80.3901	122.843
3	0.17221	0.39476	0.43303	1.57738	7.26731	79.9404	126.097
4	0.15222	0.46501	0.38277	1.63141	7.23605	79.5966	129.855
5	0.13639	0.52063	0.34298	1.68878	7.21137	79.3251	133.962
6	0.12355	0.56576	0.31069	1.74858	7.19138	79.1052	138.322
7	0.11292	0.60311	0.28397	1.81023	7.17486	78.9234	142.870
8	0.10398	0.63454	0.26148	1.87330	7.16097	78.7706	147.561
9	0.09635	0.66135	0.24230	1.93749	7.14912	78.6403	152.365
10	0.08977	0.68449	0.22574	2.00257	7.13890	78.5279	157.258
11	0.08403	0.70467	0.21130	2.06840	7.13000	78.4300	162.224
12	0.07898	0.72242	0.19860	2.13483	7.12216	78.3438	167.251
13	0.07450	0.73816	0.18734	2.20177	7.11522	78.2674	172.327
14	0.07050	0.75221	0.17729	2.26914	7.10901	78.1992	177.445
15	0.06691	0.76482	0.16826	2.33687	7.10344	78.1379	182.598
16	0.06367	0.77622	0.16011	2.40492	7.09840	78.0824	187.782
17	0.06073	0.78656	0.15271	2.47324	7.09383	78.0321	192.992
18	0.05805	0.79599	0.14596	2.54179	7.08965	77.9861	198.225
19	0.05559	0.80462	0.13979	2.61055	7.08582	77.9440	203.477
20	0.05334	0.81255	0.13411	2.67949	7.08228	77.9051	208.746

E) increases significantly, due to the increased b_m. Consequently, the average task response time increases noticeably, while the average number of VMs and the cost reduce slightly, and the cost-performance ratio increases significantly. Therefore, the best auto-scaling scheme is the one with $x = 1$, a surprising result.

7.2 Optimization of Platforms

In addition to S, the service rate μ is also an important parameter that a service provider can decide. One should notice that changing μ does not mean scale-up or scale-down, since μ is pre-set and once set, does not change with the current workload. Intuitively, increasing μ reduces T and M. However, the cost might increase due to the increased dynamic energy consumption. Thus, it is an interesting problem to find the optimal μ that minimizes CPR.

Let us consider $\alpha = 2$, $\beta = 5$, $a_m = m$, and $b_m = 2m$, for all $m \ge 1$. For $\lambda = 7$ and $\mu = 1.0, 1.5, \ldots, 5.0$, we show p_{over}, p_{normal}, p_{under}, T, M, cost, and CPR for an elastic platform in Table 5. It is observed that as μ increases, both T and M reduce significantly, and both cost and CPR decrease and then increase. Hence, there is an optimal choice of μ which minimizes CPR.

TABLE 5
Optimal Service Rate

μ	p_{over}	p_{normal}	p_{under}	T	M	cost	CPR
1.0	0.09672	0.66179	0.24148	1.75809	7.13653	78.5018	138.0129
1.5	0.12684	0.56279	0.31037	1.27455	4.83683	64.6925	82.4540
2.0	0.15291	0.49222	0.35487	1.03154	3.69250	66.4649	68.5613
2.5	0.17865	0.44112	0.38023	0.87816	3.00810	77.0826	67.6905
3.0	0.20587	0.40288	0.39125	0.76730	2.55354	94.4809	72.4955
3.5	0.23523	0.37308	0.39169	0.68031	2.23095	117.9612	80.2501
4.0	0.26672	0.34887	0.38440	0.60849	1.99165	147.3819	89.6805
4.5	0.30002	0.32842	0.37156	0.54732	1.80863	182.8975	100.1033
5.0	0.33461	0.31054	0.35485	0.49422	1.66560	224.8560	111.1278

TABLE 6
Comparison of Platforms

Platform	p_{over}	p_{normal}	p_{under}	T	M	cost	CPR
P	0.09103	0.66913	0.23984	1.73563	10.14662	111.6128	193.7182
P'	0.04720	0.87296	0.07984	2.15966	10.07341	110.8075	239.3071

7.3 Comparison of Service Providers

In this section, we consider the following problem. For a given application environment specified by λ and μ, how to compare two different cloud service providers specified by $P = (\alpha, \beta, S)$ and $P' = (\alpha', \beta', S')$. Our approach is to compare the CPR(P) and CPR(P') provided by the two cloud computing platforms.

Assume that $\lambda = 10$ and $\mu = 1$. Platform P is specified by $\alpha = 2$, $\beta = 5$, $a_m = m$, and $b_m = 2m$, for all $m \ge 1$. Platform P' is specified by $\alpha' = 3$, $\beta' = 5$, $a'_m = m$, and $b'_m = 3m$, for all $m \ge 1$. It is clear that Platform P' is less responsive, but has faster virtual machine start-up rate. For both platforms, we show p_{over}, p_{normal}, p_{under}, T, M, cost, and CPR in Table 6. It is observed that Platform P' has greater elasticity, longer task response time, less VMs, lower cost, and higher cost-performance ratio. Thus, Platform P is preferred to Platform P'.

8 Concluding Remarks

We have emphasized two significant issues in elastic cloud computing, i.e., the need of a quantifiable, measurable, observable, and calculable metric of elasticity and a systematic approach to modeling, quantifying, analyzing, and predicting elasticity, and the need of an effective way for prediction, comparison, and optimization of performance and cost in an elastic cloud platform. This paper has contributed significantly to address these two pressing issues. We have not only developed analytical model and method to precisely calculate the elasticity value of a cloud platform, but also applied our model and method to predict many important properties of an elastic cloud computing system and to optimize an elastic scaling scheme and a cloud computing platform to deliver the best cost-performance ratio.

The main challenge of our CTMC model is lack of closed-form expressions for its major elasticity, performance, and cost metrics, e.g., p_{over}, p_{normal}, p_{under}, T, M, cost, and CPR. This makes analytical study of an elastic cloud computing platform very difficult. Future research efforts should be directed towards this direction.

Acknowledgments

The author would like to express his gratitude to four anonymous reviewers for their criticism and comments on improving the quality of the manuscript.

References

[1] M. Aazam and E.-N. Huh, "Cloud broker service-oriented resource management model," *Trans. Emerging Telecommun. Technol.*, vol. 28, no. 2, pp. 1–17, 2017.

[2] M. Aazam, E.-N. Huh, M. St-Hilaire , C.-H. Lung, and I. Lambadaris, "Cloud customer's historical record based resource pricing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 7, pp. 1929–1940, Jul. 2016.

[3] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez, and W. Wang, "Quality-of-service in cloud computing: Modeling techniques and their applications," *J. Internet Services Appl.*, vol. 5, no. 11, pp. 1–17, 2014.

[4] J. R. Artalejo, D. S. Orlovsky, and A. N. Dudin, “Multi-server retrial model with variable number of active servers,” *Comput. Ind. Eng.*, vol. 48, no. 2, pp. 273–288, 2005.

[5] L. Badger, T. Grance, R. Patt-Corner, and J. Voas, *Cloud Computing Synopsis and Recommendations*, IEEE Standard 800–146, National Institute of Standards and Technology, U.S. Department of Commerce, 5/29/2012.

[6] A. K. Bardsiri and S. M. Hashemi, “QoS metrics for cloud computing services evaluation,” *I.J. Intell. Syst. Appl.*, vol. 12, pp. 27–33, 2014.

[7] R. Buyya, J. Broberg, and A. Goscinski, Eds., *Cloud Computing Principles and Paradigms*. Hoboken, NJ, USA: Wiley, 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 Comput. Syst.*, vol. 28, pp. 861–870, 2012.

[9] J. Cao, K. Hwang, K. Li, and A. Zomaya, “Optimal multiserver configuration for profit maximization in cloud computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.

[10] 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 Trans. Comput.*, vol. 63, no. 1, pp. 45–58, Jan. 2014.

[11] W. Dawoud, I. Takouna, and C. Meinel, “Elastic VM for cloud resources provisioning optimization,” *Adv. Comput. Commun.*, vol. 190, pp. 431–445, 2011.

[12] S. Dustdar, Y. Guo, B. Satzger, and H.-L. Truong, “Principles of elastic processes,” *IEEE Internet Comput.*, vol. 15, no. 5, pp. 66–71, Sep./Oct. 2011.

[13] J. O. Fitó, I. Goiri, and J. Guitart, “SLA-driven elastic cloud hosting provider,” in *Proc. 16th Euromicro Conf. Parallel Distrib. Netw.-Based Process.*, 2010, pp. 111–118.

[14] G. Galante and L. C. E. de Bona, “A survey on cloud computing elasticity,” in *Proc. IEEE/ACM 5th Int. Conf. Utility Cloud Comput.*, 2012, pp. 263–270.

[15] R. Ghosh, D. Kim, and K. S. Trivedi, “System resiliency quantification using non-state-space and state-space analytic models,” *Rel. Eng. Syst. Safety*, vol. 116, pp. 109–125, 2013.

[16] R. Ghosh, F. Longoy, V. K. Naikz, and K. S. Trivedi, “Quantifying resiliency of IaaS cloud,” in *Proc. 29th IEEE Int. Symp. Reliable Distrib. Syst.*, 2010, pp. 343–347.

[17] R. Ghosh, V. K. Naik, and K. S. Trivedi, “Power-performance trade-offs in IaaS cloud: A scalable analytic approach,” in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw. Workshops*, 2011, pp. 152–157.

[18] Z. Gong, X. Gu, and J. Wilkes, “PRESS: Predictive elastic resource scaling for cloud systems,” in *Proc. Int. Conf. Netw. Service Manage.*, 2010, pp. 9–16.

[19] N. R. Herbst, Quantifying the Impact of Platform Configuration Space for Elasticity Benchmarking, Study Thesis, Department of Informatics, Karlsruhe Institute of Technology, Karlsruhe, Baden-Wrttemberg, 2011.

[20] N. R. Herbst, S. Kounev, and R. Reussner, “Elasticity in cloud computing: What it is, and what it is not,” in *Proc. 10th Int. Conf. Autonomic Comput.*, 2013, pp. 23–27.

[21] K. Hwang, X. Bai, Y. Shi, M. Li, W.-G. Chen, and Y. Wu, “Cloud performance modeling with benchmark evaluation of elastic scaling strategies,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 130–143, Jan. 2016.

[22] S. Islam, K. Lee, A. Fekete, and A. Liu, “How a consumer can measure elasticity for cloud platforms,” School Inf. Technol., Univ. Sydney, Camperdown, NSW, Australia Tech. Rep. 680, 2011.

[23] A. I. Ivaneshkin, “Optimizing a multiserver queuing system with a variable number of servers,” *Cybern. Syst. Anal.*, vol. 43, no. 4, pp. 542–548, 2007.

[24] H. Khazaei, J. Mišić, and V. B. Mišić, “A fine-grained performance model of cloud computing centers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2138–2147, Nov. 2013.

[25] H. Khazaei, J. Mišić, V. B. Mišić, and S. Rashwand, “Analysis of a pool management scheme for cloud computing centers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 5, pp. 849–861, May 2013.

[26] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. Hoboken, NJ, USA: Wiley, 1975.

[27] M. Kuperberg, N. Herbst, J. von Kistowski, and R. Reussner, “Defining and quantifying elasticity of resources in cloud computing and scalable platforms,” Karlsruhe Institute of Technology, Karlsruhe, Karlsruhe Reports in Informatics, Tech. Rep. 16, 2011.

[28] K. Li, “Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management,” *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 122–137, Apr.-Jun. 2016.

[29] C. Liu, K. Li, C.-Z. Xu, and K. Li, “Strategy configurations of multiple users competition for cloud service reservation,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 508–520, Feb. 2016.

[30] M. Mao and M. Humphrey, “A performance study on the VM startup time in the cloud,” in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, 2012, pp. 423–430.

[31] J. Mei, K. Li, A. Ouyang, and K. Li, “A profit maximization scheme with guaranteed quality of service in cloud computing,” *IEEE Trans. Computers*, vol. 64, no. 11, pp. 3064–3078, Nov. 2015.

[32] P. Mell and T. Grance, “The NIST definition of cloud computing,” U.S. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. 800–145, Sep. 2011.

[33] S. Pacheco-Sanchez , G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, “Markovian workload characterization for QoS prediction in the cloud,” in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, 2011, pp. 147–154.

[34] N. Roy, A. Dubey, and A. Gokhale, “Efficient autoscaling in the cloud using predictive models for workload forecasting,” in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, 2011, pp. 500–507.

[35] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, “A cost-aware elasticity provisioning system for the cloud,” in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 559–570.

[36] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, “CloudScale: Elastic resource scaling for multi-tenant cloud systems,” in *Proc. 2nd ACM Symp. Cloud Comput.*, 2011, Article No. 5.

[37] P. Sobeslavsky, “Elasticity in Cloud Computing,” Master Thesis in Informatics, Joseph Fourier Univ., Grenoble, France, 2011.

Keqin Li is a SUNY distinguished professor of computer science. He is also a distinguished professor of Chinese National Recruitment Program of Global Experts (1000 Plan) with the Hunan University, China. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber-physical systems. He has published more than 480 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Cloud Computing*, the *IEEE Transactions on Services Computing*, and the *IEEE Transactions on Sustainable Computing*. He is a fellow of the IEEE.