Hindawi
Scientific Programming
Volume 2019, Article ID 7457938, 14 pages
https://doi.org/10.1155/2019/7457938

Research Article

Design and Analysis of Sustainable and Seasonal Profit Scaling Model in Cloud Environment

Monika Kumari and G. Sahoo

Department of Computer Science and Engineering, BIT Mesra, Ranchi, Jharkhand, India

Correspondence should be addressed to Monika Kumari; monika.kit@gmail.com

Received 17 May 2019; Revised 17 August 2019; Accepted 23 September 2019; Published 24 October 2019

Guest Editor: Justin Shi

Cloud is a widely used platform for intensive computing, bulk storage, and networking. In the world of cloud computing, scaling is a preferred tool for resource management and performance determination. Scaling is generally of two types: horizontal and vertical. The horizontal scale connects users' agreement with the hardware and software entities and is implemented physically as per the requirement and demand of the datacenter for its further expansion. Vertical scaling can essentially resize server without any change in code and can increase the capacity of existing hardware or software by adding resources. The present study aims at describing two approaches for scaling, one is a predator-prey method and second is genetic algorithm (GA) along with differential evolution (DE). The predator-prey method is a mathematical model used to implement vertical scaling of task for optimal resource provisioning and genetic algorithm (GA) along with differential evolution(DE) based metaheuristic approach that is used for resource scaling. In this respect, the predator-prey model introduces two algorithms, namely, sustainable and seasonal scaling algorithm (SSSA) and maximum profit scaling algorithm (MPSA). The SSSA tries to find the approximation of resource scaling and the mechanism for maximizing sustainable as well as seasonal scaling. On the other hand, the MPSA calculates the optimal cost per reservation and maximum sustainable profit. The experimental results reflect that the proposed logistic scaling-based predator-prey method (SSSA-MPSA) provides a comparable result with GA-DE algorithm in terms of execution time, average completion time, and cost of expenses incurred by the datacenter.

1. Introduction

Cloud computing, the booming word in the field of distributed parallel computing, provides cheaper and powerful processing and storage technology [1, 2]. It is a geographically distributed resource sharing system. It deals with three types of resources, compute intensive, storage intensive, and network intensive, and also provides mainly three types of services: Infrastructure-as-a-Service, Platform-as-a-Service, and Software-as-a-Service [1]. The various cloud deployment models provide the way to access these services classified as public, private, hybrid, and federated cloud [3].

Virtualization is considered as the core component of the cloud environment, enabling access to resources. We can think virtualization as an abstraction of the computer. There are three primary fundamentals of virtual environment: guest, host, and virtualization layer [4]. The guest is the part of a system that communicates with the virtualization layer directly. The host provides the actual environment for the guest to process the job. The regeneration of homogeneous and heterogeneous environment for processing the job of guest is the responsibility of the virtualization layer. Cloud on-demand gives resource accessibility of hardware platform, operating systems, storage devices, and network resources to use virtually. Assigning users' requirement to the sui resources distributed geographically over the internet is a tedious job. To perform this task, resources are divided into several execution environments known as virtual machines (VMs), thus providing a complete system to execute user requests [5]. The virtualization technique enables elastically scalable allocation of resources on-demand to the user during the normal season as well as at peak season time efficiently [1].

In recent years, various organizations, namely, Amazon and Google, provide platforms to avail computing and storage facilities by charging some amount as per the user's need. Resource allocation is a way to assign available resources to the needed cloud request as per their demand. It is possible only by employing an efficient resource allocation technique; otherwise, the proper use of scarce resources within the limit of cloud cannot meet cloud user expectations[6].

Cloud computing provides a multiaccess system in which the resource exhibits unpredictable behavior upon the arrival of new user requests, whereas multiaccess systems result in complexity in the management of resources on-demand [7]. Therefore, optimality in the allocation of desired resources is highly desirable with cloud service providers.

The two major aspects of resource allocation in the cloud are VM allocation and resource provisioning. In the literature, we found there are two classifications for resource provisioning: reservation and on-demand plans [8]. When a customer acquires resource in advance based on its probable estimation of resource requirements, it is known as a reservation plan, while when customer acquires resource dynamically at runtime, it is known as an on-demand plan. The cost of resource provisioning is cheaper in the reservation plan than in the on-demand plan, but in the reservation plan, the consumer has to pay in advance to the provider [9]. Due to uncertainty in demand for resources, both plans cannot put the best provisioning scheme. On the basis of reference [10], various challenges in incorporating resource provisioning into operations are shown in Figure 1. Many issues are also involved in consideration of the selection of resource with conflicting demands.

An attempt has been made here to address the uncertain arrival of demand with maximum utilization of the resource. We formulated a predator-prey-based mathematical model to effectively autoscale the resource as per demand with optimum cost. Autoscaling is the process of provisioning and deprovisioning of the resource by adapting the changes incurred due to oscillation in demand. A set of decisions are needed to be taken in such a way that cost is minimized and QoS is maximized [10]. This is the automatic control problem. We present the control cycle of autoscaling in Figure 2 [11].

Here, the unlimited distributed resources in the cloud that are maintained at a positive level are considered renewable resources. Allocation, the central management of renewable resources, involved some critical considerations like what should be the provisioning rate? Also, how sensitive the resource fluctuation caused by the provisioning or by the system failures is? Therefore, the renewable resource control problem is similar to the predator-prey model. A possible phenomenon of this was modeled by Lotka and Volterra [12, 13, 14]. In this paper, we used a predator-prey model for formulating our resource management problem.

The above model is based on the classic predator-prey model of Lotka–Volterra equations and Holling type II functional response. Here, the functional response is the principal of time budget due to behavioral change of system.

The Lotka–Volterra equation is not fit for every situation. Some assumptions restrict the application of this model. The important assumption is that the survival of the predator (jobs) depends on the presence of the prey (resource) [15]. In the predator-prey model, there are two stages in predation: searching for food and handling of food which is very similar to the resource provisioning in a cloud environment; in the first step, we have to identify or search the suitable VMs for the requested job (predator), and in the next step, we have to process the job on allocated resources (prey).

The logistic prey (resource) growth and Holling type II functional response (effect of behavioral change of system after scaling) are taken into consideration. Holling type II response function allows Hopf bifurcation due to the stochastic nature of the arrival of jobs (predator). The Hopf oscillator is the appropriate tool to simulate the seasonal variation as the Hopf oscillator converges into one limit cycle from any set of initial conditions [16].

We consider here optimal scaling of renewable resources such as processing capacity, storage capacity, and bandwidth capacity in the cloud and try to find the solution for the following questions:

(i) With proposed reasonable scaling model for renewable resource, what would be the maximal sustainable scaling?

(ii) Concerning to giving economical parameters like discount rate, price, and overhead costs, what is the maximal sustainable profit?

(iii) At what time or what rate should scaling resume to maximize the long-term profit?

The proposed algorithm was designed with a combination of the genetic algorithm (GA) and the differential evolution (DE) for resource scaling to identify optimum solutions to minimize the task and provide the best schedule within the cloud environment.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of the problems related to the control management and prediction models of resource provisioning in the cloud environment. Section 3 describes the motivation for the present work. The proposed models and algorithms are presented in Sections 4 and 5, respectively. Sections 6 and 7 are devoted to the experimental setup for simulation and result and discussion. Finally, conclusions and future works are presented in Section 8.

2. Related Work

Resource provisioning is one of the crucial and important tasks in cloud computing. In this process, we have to make a sequence of optimal decisions including the identification and selection of resources, determination of resource capacity for allocation, acquistion and release of resources, and methods to use resources dynamically at runtime. Cloud computing provides on-demand scaling or dynamic resource provisioning which helps cloud

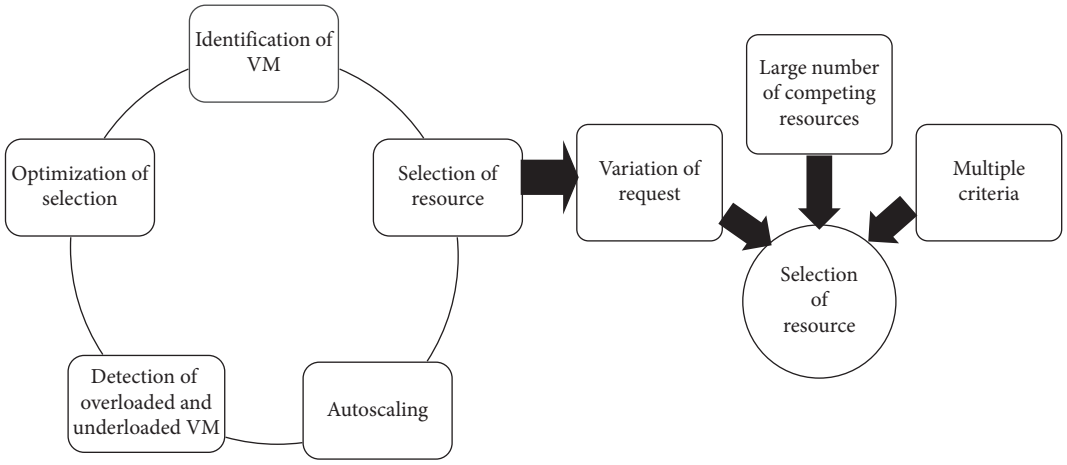

Figure 1: Challenges of resource provisioning.

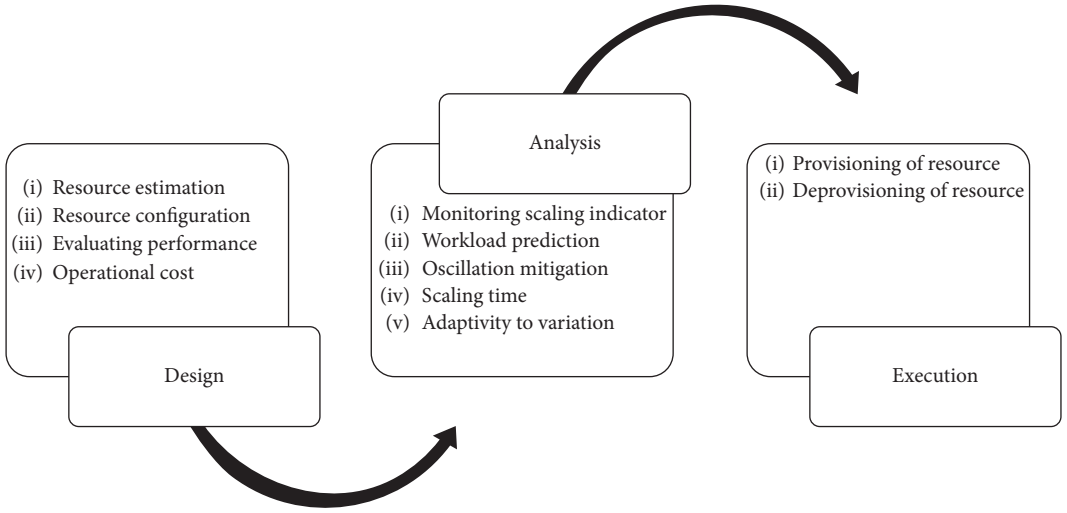

Figure 2: Control cycle and challenges of autoscaling.

providers to deliver service level agreements (SLAs) related to performance-based metrics such as response time and execution time [17]. Many researchers proposed different solutions for autoscaling of resource on the cloud. Mao proposed a budget and deadline-based autoscaling technique and developed a data prefetching algorithm to reduce the intermediate data transfer time [18]. Qu provided autoscaling techniques and solutions for achieving efficient resource utilization with high-cost efficiency, low network latency, and high availability of web application [10]. Also, Qu claimed that the proposed approach maintains acceptable QoS and also increased the number of requests served during overloading periods. Mosleh et al. [19] came up with adaptive cost-based task scheduling algorithm and claimed that it had better performance and CPU utilization. Kohavi and Longbotham [20] outlined the experimental results of two popular web applications showing the direct impact of increasing response time on their revenues. Cloud providers, however, have difficulty in ensuring an SLA response time as described by Islam et al. [21].

Also, for efficient utilization of the resource, the challenge is to model a system's workload behavior so that a predictive function can be used to complete the resource scaling in advance. Previous work used different approaches to model application workload behavior and predict the resources needed to ensure performance. Most of the work used CPU and memory resource usage data as prime inputs to their predictive models because these two resources mainly affect the performance of a task. Ai et al. [22] contributed a Markov chain-based continuous time model for predicting the measurement of elasticity. Wang et al. [23] proposed a theory based on an exogenous model of the autoregressive moving average to predict CPU and memory resource need and scale up virtual machine configurations that include cores, memory, types, and speed. Lu et al. [24] outlined prediction model for reducing undesirable energy consumption of datacenter based on backpropagation neural network. Roy et al. [25] also proposed an ARMA workload prediction model to minimize costs. Huang et al. [26] reported computer resource scaling to meet QoS requirements. Their approach is rule-

based and model-based. It has been found that the model-based approach was the most promising mechanism for managing resources to provide QoS guarantees. Wang et al. [23] used vertical scaling to analyze the relationship between CPU resource allocation and mean response time. Iqbal et al. [27] proposed a reactive model to automatically detect and fix resource bottlenecks to provide a fixed model. Han et al. [28] used a G/G/n queuing model to develop an adaptive scaling technique to minimize cloud infrastructure pricing for users. Bi et al. [29] used a combination of M/M/c and M/M/1 queuing models to dynamically manage resource allocations to cloud-based VMs. Wu proposed a cost-effective dataset management model [30].

A control theory approach for systems provides a rigorous technique for modeling, designing, analyzing and evaluating system performance, as mentioned by Zhu et al. [31]. The theory of control can be used to manage the uncertainty and disruption of a system, described by Maggio et al. [32]. Padala et al. [33] relied on control theory and discussed CPU utilization. Kamra et al. [34] and Liu et al. [35] showed resource management efficiency, but not in cloud computing. Kalyvianaki et al. [36] suggested a self-adaptive controller, which used a Kalman filter to allocate CPU resources dynamically to virtualized requests in a cloud environment. They refer to CPU usage data fluctuations. Diao et al. [37] suggested a method for maintaining system performance by controlling the server's maximum number of connections and monitoring the use of the CPU. Storn and Price's [38] differential evolution (DE) algorithm is a simple but powerful population-based stochastic search technique for solving global optimization problems. It is a heuristic approach to present the possibility of minimizing continuous space functions that are nonlinear and non-differentiable. Through an extensive testbed, it is demonstrated that the DE method converges faster with greater certainty than many other acclaimed global optimization methods. The DE method requires a few variables of control. It is robust, easy to use, and suitable for parallel computation.

Goswami and Saha used first time in 2013 a predator-prey model for allocation of resource in the stable and volatile scenario in a cloud environment [39]. Based on these two scenarios, Goswami et al. used agent-based algorithm for taking an optimum decision through identifying the situation. They extended their work by using the predator-prey model for defining the elasticity of cloud resource allocation [9]. Consequently, they proposed a prediction model based on an ALVEC-model for predicting future load, dynamically auto-tuning the parameters and allocating VM accordingly. Inspired by the aforementioned paper, we used the logistic growth model to determine the resource pool size in VM level scheduling. We found that the demand of resources is scaled up according to the resource carrying capacity of the host, and the performance of the datacenter in terms of reducing the total execution time of the job can be maximized and while also maximizing profits.

3. Motivation

In the cloud, resource scaling is one of the challenging tasks for cloud distributors. This is the backbone of optimum resource utilization. In response to increased workload, cloud providers scale up and down resources (CPU, memory, storage, network, etc.) and have to face functional complexity in maintaining QoS. The user therefore only pays for the number of resources and services they use. Cloud hosting companies, on the other hand, aim to maximize profits by serving customer requirement and also maximizing resource utilization with minimum cost. The motivation of this is to guide a user in their efforts to predict resource scaling and related costs and to help resource providers maximize resource utilization while maintaining a profit. Furthermore, under the reservation plan, we have two major issues, namely, underprovisioning and overprovisioning. Underprovisioning is caused due to high demand and low available resources, while overprovisioning is due to less demand and highly available resource. Consequently, it is a problem of determining resource pool size. As a result, we are motivated to design a mathematical model for profitable optimal resource scaling pool size approximation to maximize the performance of the datacenter by reducing the total execution time of job and trying to maximize their profits.

4. Proposed Work

We proposed to design and analyze two algorithms sustainable and seasonal scaling algorithm (SSSA) and maximum sustainable profit scaling algorithm (MPSA). In the case of SSSA, we first tried to find the approximation of resource scaling ($H(U)$) under both sustainable scaling and seasonal scaling and then to determine the optimal resource scaling pool size (U^*) and time for resuming the scaling (S). For the development of MPSA, we tried to find the optimal cost per reservation and maximum sustainable profit scaling. Here, both algorithms use the differential evolution (DE) algorithm. In the proposed algorithm, DE is used to solve the next population to GA (see Figure 3). In this algorithm, the premature convergence is avoided by comparing the produced children with their parents; if the fitness value is better than the parents, then the parents are replaced by children, or else it is aborted. Once the initial population is generated, the fitness values of the solutions are evaluated. In each iteration of the algorithm, the termination condition is checked. If the termination condition is satisfied, then the algorithm produces optimal solutions. Otherwise, GA combined with DE algorithms are applied to the individuals.

4.1. The Logistic Resource Scaling Model. This model is based on two species: the number of predators (job) and the number of its prey (resource's capacity) with a fixed number of predator (job pool) per prey (resource). This is a scaled predator-prey model. First, we model the case in which the numbers of predators and the prey birth and death rates grow logistically under the environment's carrying capacity

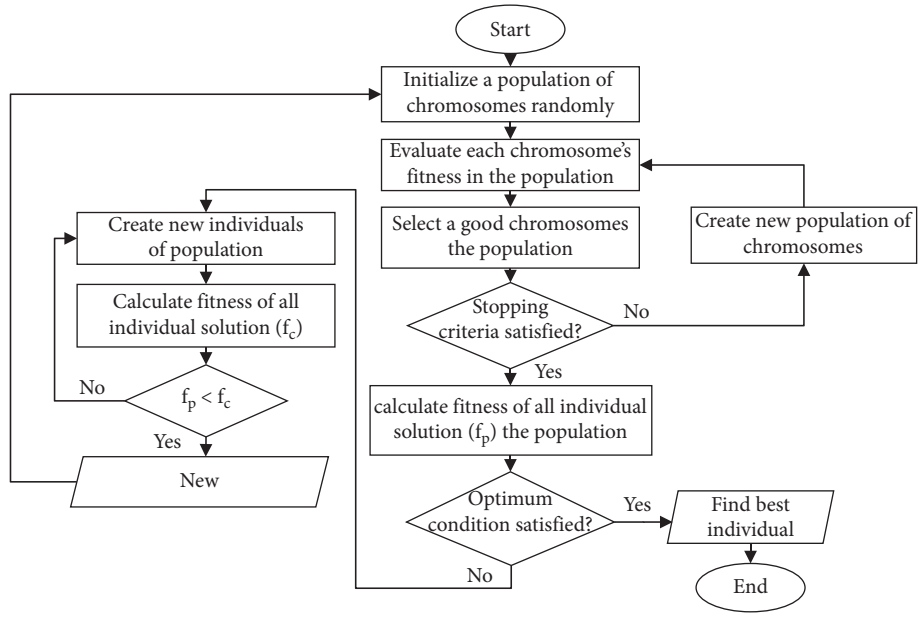

Figure 3: Flowchart of GA-DE algorithm.

limit. The second case is the effect on the population of prey after the predation. This system has a limit cycle. The range of this limit cycle will give information about the resource scaling limit. Inspired from this, we make the simple assumption that resource population under consideration scales according to the proposed model if left alone. Table 1 depicts the variables and parameters used in our model.

In the actual scenario, the four parameters: p, w, n, and C_B, are not constant. But for simplicity, we assume them here as constant. Moreover, we shall mostly be interested in the scaling of a job of constant size on particular physical machine:

$$b(t) = U > 0. \tag{1}$$

The resource scaling at particular time, denoted as $H(t)$, is

$$H(t) = qX(t)b(t), \tag{2}$$

i.e., proportional to the job per resource, $b(t)$, and the population of resource, $X(t)$, with a proportionality constant q known as resource selection rate. Therefore, for a limited job per resource, U, the resource scaling will be

$$H(t) = qUX(t). \tag{3}$$

By this assumption, the resource population, $X(t)$, of the datacenter will satisfy the logistic scaling model:

$$X'(t) = RX(t)\left[1 - \frac{X(t)}{K}\right], \tag{4}$$

where R is a constant, representing an intrinsic rate of increase in the population, and K is the carrying capacity of the datacenter. If we consider the job that is already running on a physical machine (PM), then the scaling of the resource for that job becomes active at a time, $S > 0$, and equation (4) becomes

$$X'(t) = RX(t)\left[1 - \frac{X(t)}{K}\right] - qUX(t), \quad t > S. \tag{5}$$

Equation (5) gives the change in resource capacity due to initiation of resource scaling. After some rearrangement, the overall change in resource capacity of the datacenter will represent the logistic resource scaling model as

$$X' = \begin{cases} RX\left(1 - \dfrac{X}{K}\right), & \text{for } t \le S, \\ RX\left(1 - \dfrac{X}{K} - \dfrac{qU}{R}\right), & \text{for } t > S. \end{cases} \tag{6}$$

The solution to equation (5) will be denoted as $X_1(t)$ or $X_2(t)$ for $t \le S$ and $t > S$, respectively. It is convenient to express the initial resource population as a fraction of the carrying capacity of the datacenter. Hence, we write $X(0) = K/N$, where $N > 1$.

Solving (5) for $X_1(t)$, we have

$$X_1(t) = \frac{K}{1 + (N-1)e^{-Rt}}. \tag{7}$$

Equation (4) is a time separable first-order differential equation and can be easily integrated. In the next section, we classify the resource scaling situation.

4.2. Classification of Scaling. The fulfilment of resource demand from user depends on the availability of resource at the datacenter. The case when the size of resource demand is within the average range of available resource capacity of the datacenter, we name it as stable condition, and if the demand is beyond and below the average range of resource availability, it is the case of unstable condition. The presence of stable conditions is termed sustainable scaling, whereas the presence of demand fluctuation between stable and unstable

Table 1: Variables and parameters.

Variables
$X(t)$ = the population of available resources at time t.
$B(t)$ = amount of job per resource operational at time t.
$H(t)$ = the resource scaling (units of resource scaled per unit time).
Parameters
p = the reservation rate per unit of resource.
w = wage per broker per unit of time.
n = the mean number of broker per resource.
C_B = the overhead cost of allocating one job per unit of time.

conditions is termed seasonal scaling. In the next section, we elaborate sustainable and seasonal scaling in detail.

4.2.1. Sustainable Scaling. In this section, we mostly formulate the static resource population size under sustainable scaling. When scaling starts, the aim is to increase the number of jobs such that resource scaling is sustainable. We must have a stable scaling; when $t > S$, the steady solution $X_2(t)$ of equation (5) is required. To achieve this steady state, $X_2(t)$ is chosen to satisfy

$$1 - \frac{X_2(t)}{K} - \frac{qU}{R} = 0, \tag{8}$$

which yields the following solution:

$$X_2(t) = \left(1 - \frac{qU}{R}\right)K. \tag{9}$$

If scaling starts at time $S > 0$, the resource population $X(t)$ satisfies (4) for $t < S$, and for $t > S$, we insist that $X(t) = X_2(t)$. Of course, $X(t)$ must be continuous at $t = S$; therefore,

$$X_1(s) = \frac{K}{1 + (N-1)e^{-RS}} = X_2(s) = \left(1 - \frac{qU}{R}\right)K. \tag{10}$$

Isolating the exponential gives

$$e^{-RS} = \frac{1}{(N-1)}\left(\frac{R}{R - qU} - 1\right), \tag{11}$$

and taking the logarithm of both sides and solving for S yields

$$S = \frac{1}{R}\ln\left[(N-1)\left(\frac{R}{qU} - 1\right)\right], \tag{12}$$

which gives S as a function of the job per resource size U. This formula for S indicates when scaling should be resumed. In the next section, we will try to formulate a model for maximum sustainable scaling.

(1) Maximizing Sustainable Scaling. In order to determine the maximum limit cycle of resource scaling, here we are using the concepts of Hopf oscillation. We are now trying to formulate the maximum resource scaling approximation for sustainable scaling in this section. In order to find the maximal sustainable scaling from the analysis presented above, a static resource population with a constant number of job per resource is given by expression (9) as

$$X_2(t) = \left(1 - \frac{qU}{R}\right)K. \tag{13}$$

The scaling associated with this population level is

$$H(U) = qX_2(t)U = qUK\left(1 - \frac{qU}{R}\right), \tag{14}$$

and we wish to choose U such that $H(U)$ becomes maximal. To this end, just compute the derivatives

$$H'(U) = qK - \frac{2q^2KU}{R}, \tag{15}$$

and solve the equation $H'(U) = 0$. As a result, $U^* = R/2q$, which determines the optimal amount of job. The equilibrium population X_2 is then $X_2(t) = K/2$, and the maximum sustainable scaling is

$$H(U^*) = \frac{RK}{4}. \tag{16}$$

4.2.2. Seasonal Scaling. In this section, we will assume the case of seasonal scaling and try to formulate the mechanism for optimal resource scaling size.

There is a period (typically when demand is high in a year) where the resources can grow undistributed, and then there is a (usually rather low demand) seasonal scaling. Here, we present a very elementary method to analyze such scenarios. Let X_n be the resource population in the n^{th} time. In the absence of scaling, the discrete model under consideration is of the type

$$X_n = X_{n-1} + RX_{n-1}\left(1 - \frac{X_{n-1}}{K}\right), \tag{17}$$

where X_{n-1} is the size of the resource population from the previous scaling and the latter half of the equation represents the scaling of the population using the logistic resource scaling model given in (4). Note that we are now talking about scaling rather than scaling rates as in the differential equation (4) via the standard Euler approximation where we take

$$R_1 = R\,\Delta t. \tag{18}$$

It can be observed that the discrete dynamical system (17) is, therefore, quite a rough approximation of the logistic resource scaling model. However, it offers an acceptable estimation of the next season's population. It is an estimation which can be used to predict maximal sustainable scaling, the optimal distribution of job per resource, etc.

Our objective is again to maintain a resource population, which may be scaled with an optimal job from time to time; therefore, to maximize the sustainable scaling, it is necessary to consider the scaling portion of the model. The sustainable scaling will be maximized for a population X for which the scaling term is maximized. Hence, we solve

$$\frac{\mathrm{d}}{\mathrm{d}X}\left[R_1X\left(1 - \frac{X}{K}\right)\right]\Big|_{X^*} = 0, \tag{19}$$

which occurs when $X^* = K/2$.This is identical to the population X_2 defined in equation (9). The maximal possible

scaling is then determined by substituting X^* back into the scaling model:

$$R_1 X^* \left(1 - \frac{X^*}{K}\right) = \frac{R_1 K}{2}\left(1 - \frac{1}{2}\right) = \frac{R_1 K}{4}. \tag{20}$$

This gives the identical result to the exact one from the previous calculation, except that R has been replaced by R_1.

Now, it seems natural to expect that the size of the optimal load required to scale the maximal sustainable scaling is $R_1/2q_1$, consistent with the result obtained from the differential equation model. However, this is not quite accurate because of a subtle point that we have so far ignored. The point is that we have not yet defined whether X_n is the resource population before and after the allocation season. The two will not be the same. Assuming that the peak scaling season is relatively short for the closure period, the difference between the two will just equal the scaling.

Let us agree that X_n is the resource population in time n after the peak scaling season. We write the complete model, including running scaling as

$$\left.\begin{aligned} X_n^* &= X_{n-1} + R_1 X_{n-1}\left(1 - \frac{X_{n-1}}{K}\right), \\ X_n &= X_n^* - q_1 U X_n^*. \end{aligned}\right\} \tag{21}$$

Here, the constant q_1 should be thought of as $q_1 = q\Delta t$. X^* is the resource population in time n before scaling, and X_n is the population after scaling. In particular, if $0 < X_{n-1} < K$, then $X_n^* > X_{n-1}$. This implies that the resource population is not steady and workload has to take this into account.

Substituting $X^* = K/2$ for X_n and X_{n-1} and equating $q_1[X^* + R_1 X^*(1 - (X^*/K))]U = R_1 X^*(1 - (X^*/K))$ yields

$$U^* = \frac{R_1}{q_1(2 + R_1)}. \tag{22}$$

Next, we have to compare this with the optimal resource scaling's size predicted by sustainable scaling. Note that the argument in this analysis depends on only three inputs: the available allocated resource population at the same point in time, the carrying capacity of the resource, and the constant intrinsic rate of allocation. All three can be estimated from current or previous scaling statistics. In the following section, we discuss the profit maximization of resource provider by including the complexity of economical parameters like incentive rate per resource type, maintenance cost, and overhead cost, which gives optimal resource scaling size for maximum profit and maximum sustainable profit.

5. Maximizing the Profit

We now include the additional complexity of economic parameters such as maintenance costs, overhead costs, and interest rates. First, we will set up an objective function which will represent the sustainable profit. Profit is revenue minus total cost. The revenue per unit time, denoted as $P_{\text{rev}}(t)$, is the profit per unit scaling:

$$P_{\text{rev}}(t) = pH(t). \tag{23}$$

The total cost per resource, C, is the sum of overhead cost and wage per unit time of a mean number of the broker, given as

$$C = CB + nw, \tag{24}$$

and the cost of job per unit time, $Cb(t)$, is

$$Cb(t) = C_B + nwb(t). \tag{25}$$

This gives the profit per unit time, denoted as $P(t)$, as

$$P(t) = pH(t) - Cb(t). \tag{26}$$

Let us include an interest (discount) rate $\delta > 0$, which is assumed to be constant. Using equation (26), the present value of the expected profit $E(P(t))$ at some time t is given by

$$E(P(t)) = e^{-\delta t}[pH(t) - Cb(t)]. \tag{27}$$

Integrating $E(P(t))$, the true return or total profit rate in present unit cost, denoted as J, is expressed as

$$J = \int e^{-\delta t}[pH(t) - Cb(t)]\mathrm{d}t. \tag{28}$$

This is the sustainable profit function to be maximized. The main weakness of this model is that the interest rate δ is assumed to apply to everything, yet C_B, p, and w are assumed to be constant. In reality, p will probably grow with or even faster than general inflation, and C_B is influenced by factors such as technological advancements, union negotiations, government policies, and tax rates (all come under SLAs). It is possible to include stochastic fluctuations in numerical simulation to arrive at more realistic predictions. However, for a first analysis, we will proceed with the given unrealistic assumptions.

Function (28) becomes

$$J(b) = \int_0^\infty e^{-\delta t} b(t)[pqX(t) - C]\mathrm{d}t, \tag{29}$$

which is a function of resource pool size $b = b(t)$. It is easy to include a modification where the reservation price of the resource and the cost per job, p and C, both increase with time. Suppose, for example, that they both grow at the same rate $\propto$ such that

$$p = P(t) = P_0 e^{\propto t}, \tag{30}$$

$$C = C(t) = C_0 e^{\propto t}. \tag{31}$$

We then obtain an objective function

$$J(b) = \int_0^\infty e^{(\propto - \delta)t} b(t)[P_0 qX(t) - C_0]\mathrm{d}t, \tag{32}$$

which is well defined if $\delta > \propto$ (if $\delta \leq \propto$ the function will in general no longer be finite). Now, when both p and C grow with time, we can replace δ by $\delta - \propto$ so that we can conclude at what time, $t = S > 0$, a workload of constant size $b(t) = U$ should resume scaling such that $J(b)$ is maximized.

5.1. Optimal Resource Scaling Pool Size. In this section, we explain how to find the resource scaling pool size for a maximum profit. Having found a formula for S in equation (12) (the time when scaling should start) as a function of U, the profit function (which in equation (28) is a function of $b(\cdot)$, i.e., of U and S) is a function of U alone.

Substituting $X_1(t) = X_2(t)$ from equation (9) and S as given in equation (12) reduces equation (28) to

$$J(U) = U\int_{S(U)}^{\infty} e^{-\delta t}\left[pqK\left(1-\frac{qU}{R}\right)-C\right]\mathrm{d}t, \tag{33}$$

where for a given resource pool size U, everything except the exponent is constant. Integration gives

$$J(U) = U\left[pqK\left(1-\frac{qU}{R}\right)-C\right]\frac{e^{-\delta S(U)}}{\delta}, \tag{34}$$

and after making the substitution $\beta = C/pqK$, this becomes

$$J(U) = \frac{pqKU}{\delta}\left(1-\frac{qU}{R}-\beta\right)e^{-\delta S(U)}. \tag{35}$$

This profit function will be negative if $1-(qU/R)-\beta <$. This indicates the condition under which it is no longer profitable to scale. The profit function $J(U)$ will be maximized at the point U^* such that

$$U^* = \frac{R}{4q}\left[3-\beta+\frac{\delta}{R}-\sqrt{\left(1+\beta-\frac{\delta}{R}\right)^2+\frac{8\beta\delta}{R}}\right]. \tag{36}$$

Scaling should resume at the time $S^* = S(U^*)$ to maximize the profit.

5.2. Maximal Sustainable Profit. To calculate the maximal sustainable profit, we have to maximize the profit per unit time given by equation (26) in an equilibrium situation. In this case, we take $b(t) = U$ (constant), and substituting the value of $H(t)$ then the rate of profit $P(t)$ will be $U[pqx(t)-c]$, with value of $X(t)$ at equilibrium from the equation (9) becomes the following simple function:

$$J(U) = U\left[pqK\left(1-\frac{qU}{R}\right)-C\right], \tag{37}$$

which is maximized for $U^* = (R(1-\beta))/2q$, with $\beta = C/pqK$, as shown in Section 4 earlier. The associated equilibrium resource population is then

$$X_{\mathrm{eq}} = \frac{K}{2}(1+\beta). \tag{38}$$

The scaling rate H^* and sustainable profit P^* become

$$H^* = qX_{\mathrm{eq}}U^* = \frac{RK}{4}\left(1-\beta^2\right), \tag{39}$$

$$P^* = U^*\left(pqX_{\mathrm{eq}}-C\right) = \frac{pRK}{4}\left(1-\beta^2\right). \tag{40}$$

Next, we develop two algorithms on the above formulated logistic scaling-based predator-prey model. In sustainable and seasonal scaling algorithm (SSSA), we try to find the scaling rate of each allocated resources and check the condition for sustainable and seasonal scaling. We also check the condition to obtain the optimal resource scaling size. Next, in the maximum profit scaling algorithm (MPSA), we evaluate profit per scaling and profit to achieve maximum sustainable scaling and optimum resource scaling size with maximum profit.

5.3. Proposed Algorithm. Now both Algorithms 1 and 2 proposed earlier can be defined as follows.

6. Experimental Setup

To test the proposed model, we first numerically verified this model using the Runge–Kutta fourth order (RK4) approximation method and Adams–Moulton method for parameter estimation. Based on two cases of predator-prey population variation, we analyzed the stability condition. Two cases of parameter variations are as follows:

(i) Job completion rate is higher than job arrival rates $(q > R)$

(ii) The job arrival rate is higher than the job completion rate $(R > q)$

The RK4 method produces sets of the approximate solutions by dividing the solution domain into a set of discrete points. We start the initial data at time $t = 0$ and then estimate the approximation at time $t = i*h$, where $i = 1, 2, 3, \ldots, n$. The step size h is chosen suitably. In each step i, it generates a different solution. By using this different solution, we are evaluating our algorithm for sustainable and seasonable scaling criteria. Under each case, we are evaluating the cost and profitability scaling approximation. We examine the behavior of logistic scaling models with several sets of the parameter's value given in Table 2.

Figure 4 shows the behavior of resource availability, job arrival, and corresponding maximum possible scaling under the sustainable condition for two sets of parameter variation.

We again verified our proposed algorithm for maximum performance of the datacenter by reducing the total execution time (makespan) of the job and maximizing the profits of the datacenter. The proposed scheme is developed using Java programming language-based simulation tools NetBeans 8.1 and CloudSim 4.0. In construction of a cloud datacenter environment, the servers had 240 times MIPS of any VM, 16384 MB RAM, 1 GB storage, quad and dual core processors, and 10000 kbps bandwidth configuration. Also, virtual machines have been configured by assigning their basic entities and parameters such as 100 MIPS, 128 MB the size of RAM, 100 kbps bandwidth, 1 GB storage, and single-core processor. Each datacenter consists of two hosts. The CloudSim initialized and constructed the datacenter on the basis of the assigned configuration for the user's job. All jobs have been submitted dynamically. In almost every simulation, the jobs are dynamically submitted within a time frame, which is 1000 ms. The cloudlets for each simulation have

(1) Input: population of allocated resources = $X(t)$,
Resource scaling's size = U
Resource selection rate = q
Resource allocation rate = R
Carrying capacity of datacenter = K
Output: optimal resource scaling size, maximum sustainable, and seasonal scaling.
(2) Initialize the allocated resource population using equation (7).
(3) Find the scaling of the resource using $H(t) = qUX(t)$
If the population of allocated resources, $X(t) = K(1 - (qU/R))$, then sustainable scaling is possible
Scaling of resource $H(U) = qUK(1 - (qU/R))$
If $H(U) = RK/4$, then sustainable scaling is maximized
If scaling's size $U = R/2q$, then resource scaling's size is optimal, i.e., $(U^* = U)$.
else
Update R and q for the seasonal demand period using the following equation $R_1 = [R(n+h) - R(n)]$ and $q_1 = [q(n+h) - q(n)]$
Update allocated resource population (i.e., $X(t)$) for n^{th} time period using equation (17).
If $(H(U) = (X_n - X_n^*))$, then seasonal scaling is maximized, and if $U = R_1/(q(2 + R_1))$, then resource scaling size is optimal.

Algorithm 1: Sustainable and seasonal scaling algorithm (SSSA).

(1) Input: C_B, p, n, w.
Output: maximum profit, optimal allocation level, sustainable scaling rate, and sustainable profit rate.
(2) Calculate scaling rate $H(t)$.
(3) Calculate the total cost per resource C and the cost per unit workload.
(4) Calculate the profit per unit workload using equation (26).
(5) Calculate the total profit rate (J) using equation (28).
(6) If p and C are increasing and $\delta > \propto$, then update p and C using equations (30) and (31).
(7) Calculate the total profit rate using equation (32).
(8) Calculate the profitable scaling capacity per resource (U^*) using equation (36) and profitable scaling time as $S(U^*)$
(9) If $(U^* = (R(1-\beta))/2q)$ where $\beta = C/pqK$, then maximum sustainable profitable scaling capacity is obtained.
(10) The sustainable scaling rate in equilibrium resource population (38) can be calculated using equation (39) and profit rate by using equation (40).

Algorithm 2: Maximum profit scaling algorithm (MPSA)

Table 2: Sets of parameter value.

	Q	R	x_o	y_o	H
Case 1	6.0	1.0	[1.0.0.3, 0.4]	[6.5, 7.5, 0.4]	[0.01, 0.1, 0.25, 0.5, 1, 4, 5, 10]
Case 2	[0.2, 1.2]	[1.0, 2.0]	[0.1, 0.4]	[0.2, 0.4]	[0.01, 0.1, 0.25, 0.5, 1, 4, 5, 10]

been submitted in three batches. The initial batch of VM and cloudlets are identical for each simulation to have a fair comparison.

7. Result and Discussion

Our end goal of the numerical method was to perform a phase plane analysis on the proposed logistic scaling model. The first step in this direction was to find the nullclines of the system by equating equations (4) and (5) to zero. For phase analysis, we denoted equation (4) as $F(X,U)$ and equation (5) as $G(X,U)$. So, F-nullclines are either $X = 0$ or $X = K$ and G-nullclines are either $X = 0$ or $U = (R/q)(1 - (X/K))$. Next, we found the steady-state condition which was obtained by finding the intersection of F-nullclines with G-nullclines. Thus, the steady states are $(X_1, Y_1) = (0, 0)$, $(X_2, Y_2) = (0, R/q)$, and $(X_3, Y_3) = (K, 0)$. To find stability around each steady state, we found the Jacobian of the system. Hence, a stability condition was obtained around points (X_1, Y_1) and (X_2, Y_2) when R, $q < 0$ at each point, whereas for point (X_3, Y_3), same was obtained when $K > R/q$ and $Rq < 0$. The result shows that the stability of the model does not depend on the population of either resource or job. The scaling amount and direction of the scaling curve at each steady state were determined by the eigenvalues and corresponding eigenvectors, respectively. Therefore, in the same way, we obtained (X_1, Y_1), (X_2, Y_2), and (X_3, Y_3) as saddle points. Therefore, the scaling function H(t) is directly proportional to the resource population $X(t)$, thus implying that the resource should be scaled up as long as the resource carrying capacity shows better resource utilization.

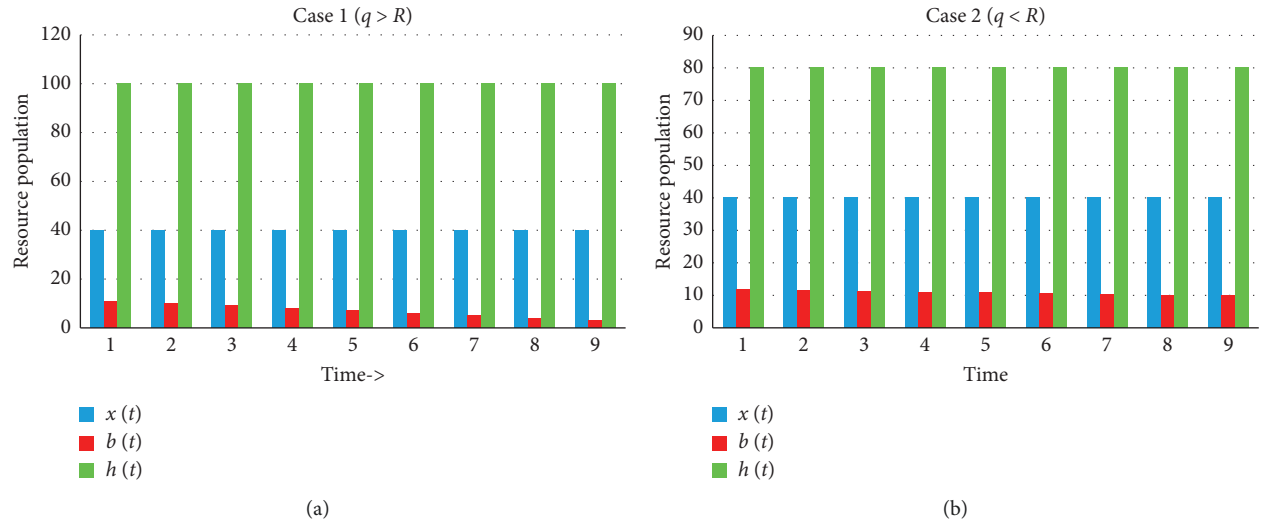

Figure 4: The behavior of logistic scaling model with different sets of parameter.

Figure 5 shows the result of analytical RK4 method for resource fluctuation. The X-axis and Y-axis represent the time variation and the resource population, respectively. This depicts the behavior of resource availability in the cloud at a particular time.

It can be seen in Figure 6 that it is the case of under-provisioning as the size of job $b(t)$ always exceeds the available resource capacity. We may conclude that in this case, seasonal scaling will solve the issue.

Figure 7 shows the results of a numerical method for the approximation of resource scaling. It confirms that the growth of scaling is directly proportional to the available resource capacity monotonically.

Figure 8 shows the results for maximum sustainable scaling of available resource $X(t)$ in terms of $H(t)$, whereas Figure 9 shows the result of instability as we move away from the saddle point (7, 52). Here also we proceed with seasonal scaling approach.

The comparative performance evaluation of the two proposed approaches is depicted in Figure 10. In Figure 10(a), we see that SSSA-MPSA performs better than GA-DE. Here, X-axis represents the number of batch and Y-axis the average completion time, i.e., throughput of tasks. In batch 1, SSSA-MPSA took 200 s while GA-DE 1100 s. Thus, we can say that SSSA-MPSA performs 9 times faster than GA-DE. Similar results can be observed in the case of batch 2 and batch 3 where SSSA-MPSA is 9 times faster than GA-DE.

A graph, in Figure 10(a), is drawn between the number of batches on X-axis and the completion time on the Y-axis. It shows that the efficiency of the resource scaling of the proposed algorithm in a datacenter. In Figure 10(b), a graph is drawn between the number of tasks and the makespan, which are represented by the X-axis and Y-axis, respectively. It depicts the improved execution time of the proposed model compared with GA-DE algorithm. In the figure, the result shows that SSSA-MPSA performs 2 times faster than GA-DE algorithm for 60 tasks and 72 tasks while GA-DE performs 2 times faster than SSSA-MPSA for 150 tasks. Another chart, in Figure 10(c), is plotted between the number of batches on X-axis and the cost of expenses on Y-axis, and it illustrates that the SLA violations are minimized and profit is maximized. Here also we find the SSSA-MPSA performs better than GA-DE.

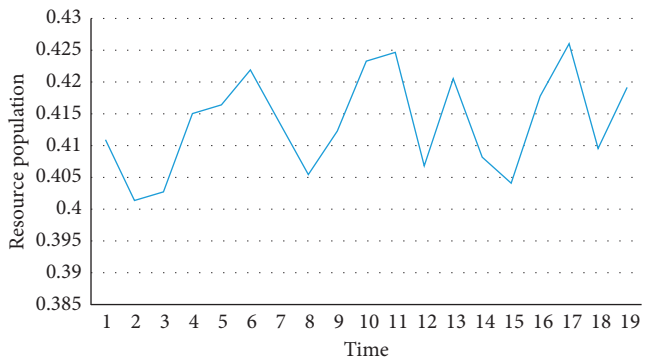

Figure 5: Fluctuation of resource availability.

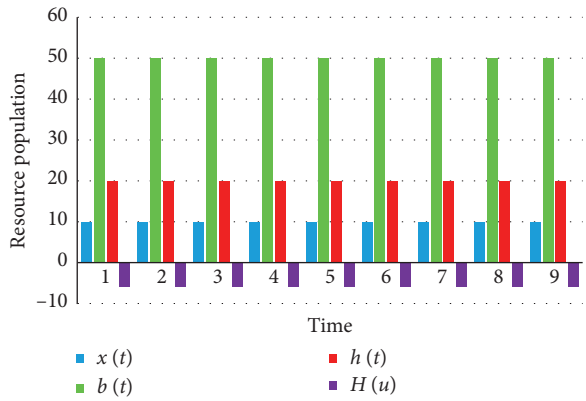

Figure 6: Unstable resource demand.

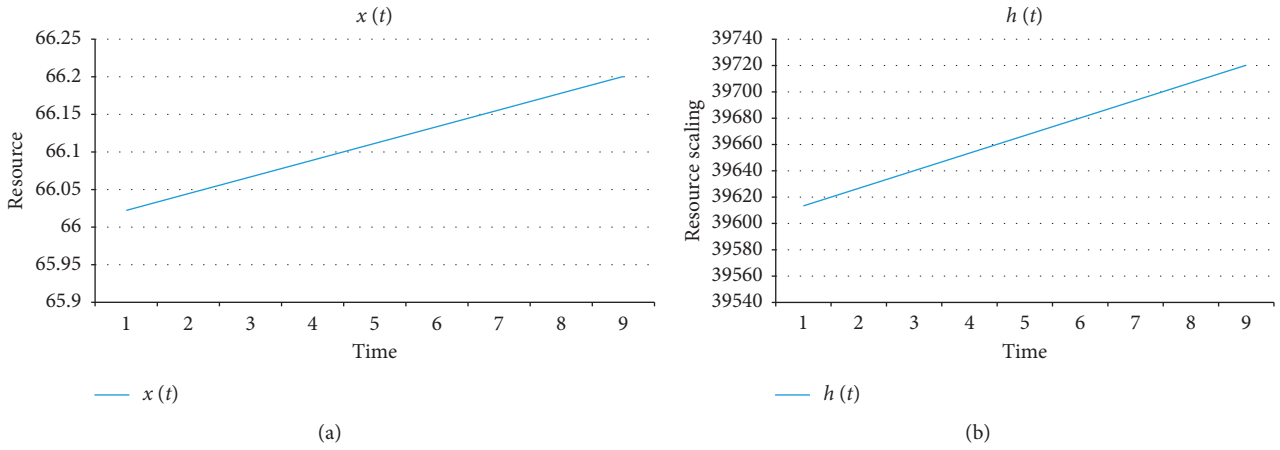

Figure 7: Approximate resource scaling.

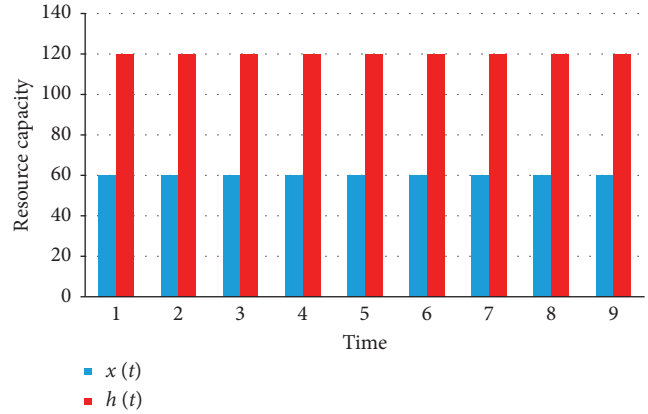

Figure 8: Maximum sustainable scaling of resource.

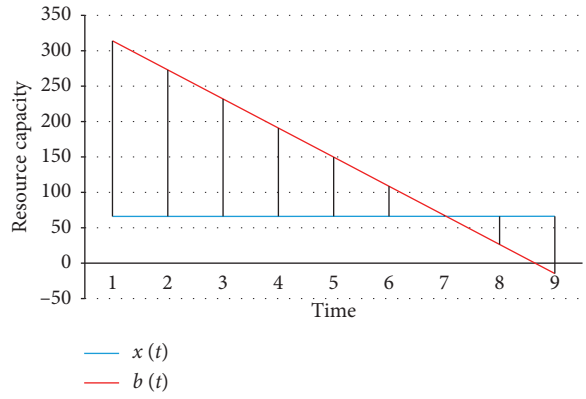

Figure 9: Unstable condition suitable for seasonal scaling.

8. Conclusion and Future Work

In resource provisioning, autoscaling is one of the major issues for the cloud environment at VM level scheduling. Mathematical models based on the predator-prey method was developed and tested to implement the vertical scaling of the task for optimal resource provisioning. Here, we tried not only to find the mechanism for maximizing sustainable and seasonal scaling but also to maximize the profit by considering the optimal resource pool size and incentives per resource for users. Logistic scaling for resource population is one of the realistic approaches rather than exponential as per the environmental carrying capacity limitation. Through phase plane analysis, we discovered three steady states and their stability, which was dependent not on the resource population or the number of jobs but instead on the parameters of the intrinsic rate of increase in resource availability, the resource scaling rate,

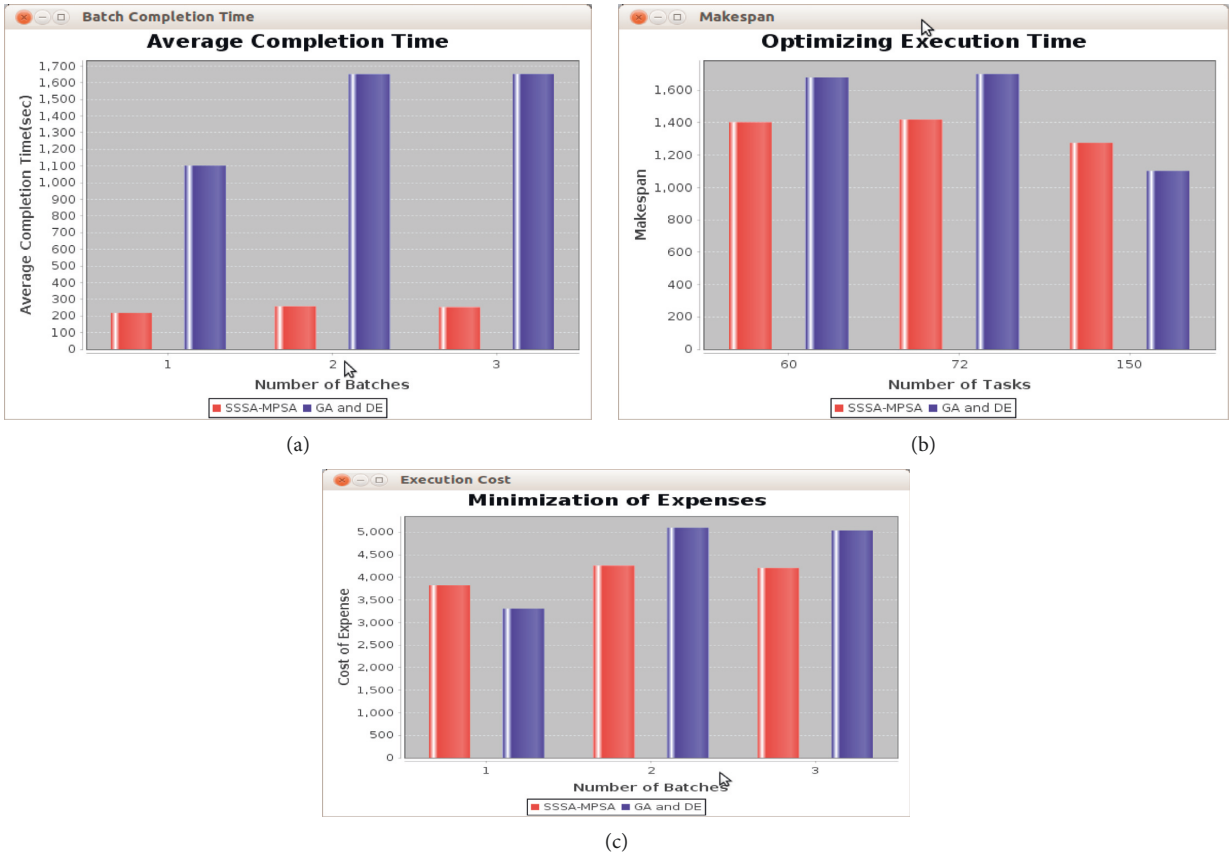

Figure 10: Comparative performance evaluation of SSSA-MPSA with GA-DE.

and the maximum datacenter resource capacity. The behavior of the model is to predict the population scaling of the resource and jobs to reach an equilibrium position. This model introduced sustainable and seasonal scaling algorithm (SSSA) and maximum sustainable profit scaling algorithm (MPSA). It tried to find the approximation of resource scaling under the sustainable and seasonal scaling and also found the optimal cost per reservation and maximum sustainable profit. The logistic scaling-based predator-prey algorithm provided a comparable result with the metaheuristic method based on genetic algorithm (GA) with differential evolution (DE) algorithm for execution time, average completion time, and cost of expenses incurred by the datacenter. For future extension, this work can be used for efficient utilization of energy consumption and minimization of energy expenditure of the datacenter.

Data Availability

The data used to support the findings of this study are included within the supplementary information.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Supplementary Materials

This file contains the sample dataset of our experiment on which we have tested our two algorithms: SSSA-MPSA and GA with DE. (*Supplementary Materials*)

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] M. Armburst, I. Stoica, M. Zaharia et al., "A view of cloud computing," *Communication of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] Y. Zhang, Y. Li, and W. Zheng, "Automatic software deployment using user-level virtualization for cloud-computing," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 323–329, 2013.

[4] M. H. Fedrus, M Murshed, R. N. Calheiros, and R. Buyya, "Network-aware virtual machine placement and migration in cloud data centers," in *Emerging Research in Cloud Distributed Computing Systems*, S. Bagchi, Ed., pp. 42–91, IGI Global, Hershey, PA, USA, 2015.

[5] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *Journal of Network and Computer Applications*, vol. 66, pp. 106–127, 2016.

[6] V. V. Vinothina, R. Sridaran, and P. Ganapathi, "A survey on resource allocation strategies in cloud computing," *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 6, 2012.

[7] K. Leonard, *Queuing System Volume II: Computer Application*, Wiley, Hoboken, NJ, USA, 2014.

[8] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Transaction on Services Computing*, vol. 5, no. 2, pp. 164–177, 2012.

[9] B. Goswami, J Sarkar, S. Saha, S. Kar, and P. Sarkar, "ALVEC: allocation by Lotka Volterra elastic cloud: a QoS aware non linear dynamical allocation model," *Simulation Modelling Practice and Theory*, vol. 93, pp. 262–292, 2019.

[10] C. Qu, "Auto-scaling and deployment of web applications in distributed computing clouds," Ph D. thesis, Department of Computing and Information Systems, The University of Melbourne, Melbourne, Australia, 2016.

[11] J. O. Kephart and D. M. Chess, "The Vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[12] R. Illner, C. S. Bohun, S. McCollum, and T. V. Roode, *Mathematical Modelling: A Case Studies Approach*, American Mathematical Society, Providence, RI, USA, vol. 27 of Student Mathematical Library, 2011.

[13] J. N. Kapur, *Mathematical Modelling*, New Age International Pvt. Ltd., Bengaluru, India, 2000.

[14] F. R. Giordano, W. P. Fox, S. B. Horton, and M. D. Weir, *A First Course in Mathematical Modelling*, Thomson Learning Academic Resource Centre, Belmont, CA, USA, 3rd edition, 2003.

[15] A. Warren, "Math modelling for undergraduates, a major qualifying project," Worcester Polytechnic Institute, Worcester, MA, USA, B.Sc project, 2012.

[16] P. E. F. Sommer, "*Analysis of the rosenzweig-macArtur model with bifurcation structures and stochastic processes*," M.S. thesis, Faculty of Sciences of the University of Lisbon, Lisbon, Portugal, 2016.

[17] A. Mosa and N. W. Paton, "Optimizing virtual machine placement for energy and SLA in clouds using utility functions," *Journal of Cloud Computing*, vol. 5, no. 1, p. 17, 2016.

[18] M. Mao, *Cloud auto-scaling with deadline and budget constraints*, Ph.D. thesis, School of Engineering and Applied Science, University of Virginia, Charlottesville, VA, USA, 2012.

[19] M. A. S. Mosleh, G. Radhamani, M. A. G. Hazber, and S. H. Hasan, "Adaptive cost-based task scheduling in cloud environment," *Scientific Programming*, vol. 2016, Article ID 8239239, 9 pages, 2016.

[20] R. Kohavi and R. Longbotham, "Online experiments: lessons learned," *Computer*, vol. 40, no. 9, pp. 103–105, 2007.

[21] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[22] W. Ai, K. Li, S. Lan et al., "On elasticity measurement in cloud computing," *Scientific Programming*, vol. 2016, Article ID 7519507, 13 pages, 2016.

[23] Z. Wang, X. Zhu, and S. Singhal, "Utilization and SLO-based control for dynamic sizing of resource partitions," in *Ambient Networks*, J. Schonwalder and J. Serrat, Eds., No. 3775 in Lecture Notes in Computer Science, pp. 133–144, Springer, Berlin, Germany, 2005.

[24] Y. Lu, J. Panneerselvam, L. Liu, and Y. Wu, "RVLBPNN: a workload forecasting model for smart cloud computing," *Scientific Programming*, vol. 2016, Article ID 5635673, 9 pages, 2016.

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

[26] D. Huang, B. He, and C. Miao, "A survey of resource management in multi-tier web applications," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1574–1590, 2014.

[27] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 871–879, 2011.

[28] R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond, "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications," *Future Generation Computer Systems*, vol. 32, pp. 82–98, 2014.

[29] J. Bi, Z. Zhu, R. Tian, and Q. Wang, "Dynamic provisioning modelling for virtualized multi-tier applications in cloud data center," in *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, pp. 370–377, Miami, FL, USA, July 2010.

[30] X. Wu, "Data sets replicas placements strategy from cost-effective view in the cloud," *Scientific Programming*, vol. 2016, Article ID 1496714, 13 pages, 2016.

[31] X. Zhu, M. Uysal, Z. Wang et al., "What does control theory bring to systems research?," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 1, pp. 62–69, 2009.

[32] M. Maggio, H. Hoffmann, M. D. Santambrogio, A. Agarwal, and A. Leva, "Decision making in autonomic computing systems: comparison of approaches and techniques," in *Proceedings of the 8th ACM International Conference on Autonomic Computing, ICAC'11*, pp. 201–204, Karlsruhe, Germany, June 2011.

[33] P. Padala, K. G. Shin, X. Zhu et al., "Adaptive control of virtualized resources in utility computing environments," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, pp. 289–302, 2007.

[34] A. Kamra, V. Misra, and E. Nahum, "Yaksha: a self-tuning controller for managing the performance of 3-tiered web sites," in *Proceedings of the Twelfth IEEE International Workshop on Quality of Service-IWQOS 2004*, pp. 47–56, Montreal, Canada, June 2004.

[35] X. Liu, J. Heo, L. Sha, and X. Zhu, "Adaptive control of multi-tiered web applications using queueing predictor," in *Proceedings of the Network Operations and Management Symposium (NOMS 2006)*, pp. 106–114, Vancouver, Canada, April 2006.

[36] E. Kalyvianaki, T. Charalambous, and S. Hand, "Self-adaptive and self-configured CPU resource provisioning for virtualized servers using Kalman filters," in *Proceedings of the 6th international conference on Autonomic computing-ICAC'09*, pp. 117–126, Barcelona, Spain, June 2009.

[37] Y. Diao, J. L. Hellerstein, S. Parekh, R. Griffith, G. Kaiser, and D. Phung, "Self-managing systems: a control theory foundation," in *Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, pp. 441–448, Greenbelt, MD, USA, April. 2005.

[38] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous

spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[39] B. Goswami and S. Saha, "Resource allocation modelling in abstraction using predator-prey dynamics: a qualitative analysis," *International Journal of Computer Application*, vol. 61, no. 6, pp. 6–13, 2013.