Communications and Network, 2020, 12, 74-97
https://www.scirp.org/journal/cn
ISSN Online: 1947-3826
ISSN Print: 1949-2421

Prediction-Based Resource Assignment Scheme to Maximize the Net Profit of Cloud Service Providers

Sarabjeet Singh, Marc St-Hilaire

Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
Email: sarabjeet.singh@carleton.ca, marc_st_hilaire@carleton.ca

How to cite this paper: Singh, S. and St-Hilaire, M. (2020) Prediction-Based Resource Assignment Scheme to Maximize the Net Profit of Cloud Service Providers. *Communications and Network*, **12**, 74-97.
https://doi.org/10.4236/cn.2020.122005

Received: April 24, 2020
Accepted: May 26, 2020
Published: May 29, 2020

Abstract

In a cloud computing environment, users using the pay-as-you-go billing model can relinquish their services at any point in time and pay accordingly. From the perspective of the Cloud Service Providers (CSPs), this is not beneficial as they may lose the opportunity to earn from the relinquished resources. Therefore, this paper tackles the resource assignment problem while considering users relinquishment and its impact on the net profit of CSPs. As a solution, we first compare different ways to predict user behavior (*i.e.* how likely a user will leave the system before its scheduled end time) and deduce a better prediction technique based on linear regression. Then, based on the RACE (Relinquishment-Aware Cloud Economics) model proposed in [1], we develop a relinquishment-aware resource optimization model to estimate the amount of resources to assign on the basis of predicted user behavior. Simulations performed with CloudSim show that cloud service providers can gain more by estimating the amount of resources using better prediction techniques rather than blindly assigning resources to users. They also show that the proposed prediction-based resource assignment scheme typically generates more profit for a lower or similar utilization.

Keywords

Cloud Service Provider, Resource Assignment, Net Profit, User Behavior, Relinquishment, Machine Learning, Linear Regression

1. Introduction

Cloud computing is known as the most widespread paradigm of distributed and parallel computing. The core feature of cloud computing is to provide reliable and consistent services to users. Users request services from the Cloud Service Providers (CSPs) with certain requirements and deadlines. The prices for these services are negotiated and then finalized into a Service Level Agreement (SLA) [2]. Based on the SLA, the provider leases/rents the amount of resources required to run the service at any place and time through the Internet. The resources that are provided to run a service are housed by a CSP in a facility called data center, which includes CPU, GPU, memory, storage.

The basic advantage for a user to use the computing resources in the form of a service from a CSP is the pay-as-you-go model [3]. In this model, users have the liberty to give up the allocated resources whenever they want. In fact, resources are allocated to users based on demand. Most of the cloud providers ask their users to determine and specify the amount of resources required to process and complete their jobs. Experienced cloud customers may be able to gather historical data and properly determine the required amount of resources. However, most users may not have data to estimate the resource needs of their applications. Subsequently, this often leads to a situation where users overestimate the amount of resources and therefore relinquish them before their scheduled completion time (*i.e.* CSPs operate in an environment where users can relinquish their services at any point in time). For example, it has been observed that many users tend to purchase 10 times the amount of resources needed for the operation of their jobs, resulting in low server resource utilization [4]. In this case, since CSPs reserve the resources for the requested duration by the users, they may lose opportunities to earn profit as users will only pay for the duration for which the service was used [1]. During this time, providers may have to deny other potential customers and bear a loss in revenue [1].

In addition, the arrival rate of users varies throughout the day, leading to variable server utilization. For example, servers may be over-provisioned during peak hours or under-provisioned during non-peak hours. Likewise, when a user relinquishes, the server remains idle until the next request arrives. This idle time could be utilized better (by being assigned to other requests, for instance) if the provider can accurately guess when the user may potentially relinquish [5]. Due to these changeable trends, resources are not adequately utilized in either case. Consequently, several models, such as [6], [7] and [8], have been developed to assign resources to improve utilization. However, they do not cover the economic aspects such as the income generated and the electricity expenses incurred by the provider.

Moreover, user behavior also plays an important part and varies along with the varying arrival rate of the users. Here, the term user behavior refers to how likely the user will be to relinquish the requested cloud services. Therefore, for optimal assignment of resources, CSPs must be aware of user behavior by predicting the duration for which they may use the service. To deal with this, various algorithms have been proposed to predict user behavior by simply computing an average based on the user history [6] [7]. Although these algorithms produced good results, they are not reliable enough in terms of accuracy due to their use of average approximation algorithms to predict user behavior. To make better decisions, the accuracy of the predictions needs to be improved. Nowadays, machine learning is considered as one of the best approaches to make predictions in any field of study. Policies based on prediction through any machine learning algorithms can prove to be handy for optimal resource assignment to maximize the net profit of cloud service providers. To that end, it could be beneficial for CSPs to store the user history related to resources utilization. The history could then be used to predict the user behavior and allocate the resources based on this information when the user returns for the service. This way, CSPs have the potential to maximize their server utilization and, at the same time, minimize their loss.

Based on the problems discussed above, the goal of this paper is to work in the subtle stream of interdisciplinary research between computer science and economics. More precisely, we:

- Propose a new technique based on linear regression to predict user behavior. Compared to other available methods (such as [6] and [7]), the method we propose provides better accuracy;
- Propose a new mathematical model to optimally assign resources so that the net profit of cloud providers can be maximized. The novelty of the model is that it considers historical user behavior when making a decision, while maintaining the market value of the provider by entertaining the maximum number of users requesting resources. The model combines the technical and economical aspects of resource management in cloud computing;
- Compare various existing resource assignment schemes and show that the best results are obtained with the proposed model.

The rest of the paper is organized as follows. In Section 2, we review the related work. Then, we present the implementation of linear regression to predict the user behavior in Section 3 followed by the proposed optimization model in Section 4. The experimental results are shown in Section 5 and finally, the paper is concluded in Section 6.

2. Related Work

With the emergence of cloud computing, the resource pool has expanded, but is still limited since it must be shared by multiple users at once. No matter which type of cloud is deployed, efficient resource management strategies are needed to harness the power of the underlying resource pool of the cloud [9]. Through effective resource management strategies, providers can increase the profit earned while improving resource utilization at the same time. There is an abundance of detailed literature available on the resource management of clouds focusing on various areas. In context to this paper, we present the studies that have used various prediction schemes and used batch workloads for allocating resources to maximize the net profit.

A. Resource management in clouds using various prediction techniques

The study of Farahnakian *et al.* [10] predicts the future utilization of the hosts to carry out the VM migration. If linear regression predicted the CPU to be underloaded, then the migration was carried out and the underloaded server was turned off to sleep mode. Simulations carried out in CloudSim showed that 51% of energy can be saved by using this approach in VM migrations.

Several researchers have used prediction techniques to estimate the amount of resources to be dynamically allocated. For example, Aazam *et al.* [6] carried out a unique study in which they predict the amount of resources to be allocated to the user based upon its historical usage records. The authors suggest that the relinquish probability of the users must be checked before allocating all the resources that are requested by the users. Based upon the historical relinquish probabilities, they predicted the current relinquish probability which is used to estimate the amount of resources to be allocated. The history of the user is divided into two parts: Service Oriented Probabilities (SOP) and Average Overall Probabilities (AOP). SOP is the probability of using the requested service and AOP is the average probability of using all the services collectively. Both of these probabilities are used in an approximation algorithm to make predictions. This unique work sets the basis of our study to include user behavior as an important parameter for estimating the resources. Moving along, Hu [7] improved the model from [6] by considering an extra parameter (current utilization of the system) before assigning resources. For example, the amount of allocated resources will be greater in case of low server utilization but smaller when the server is over utilized. Overall, this study managed to improve the server utilization, but lacked mention of the provider's net profit. The list of future work suggested by Hu in [7] highlights that rather than using average approximation algorithms, the quality of prediction could be improved if machine learning algorithms were used. Therefore, in this study, we decided to look at the concept of linear regression to make predictions.

Putting more weight on the same idea of [6] and [7], Moreno *et al.* [11] state that user behavior is important to study while characterizing the workload of the system. Their results show that the task and the user dimension vary significantly every day. This diversity of users have a momentous impact on resource utilization and energy costs. Furthermore, they claim that in most cases, users overestimate the amount of resources requested. This overestimation impacts the utilization if they end the service before their requested duration. Based on this, the authors tend to improve the utilization and energy efficiency by understanding the relationship between the user and the kind of tasks within a workload. A similar kind of study was presented by Fang *et al.* [12] in which they predict the load using an Autoregressive Integrated Moving Average (ARIMA) model. The resources were allocated according to the normal and sudden spikes in utilization. For predicted normal workload, the resources were scaled accordingly while the coarse-grained capacity scaling approach was used in case of sudden spikes. The main idea of their study was to propose a prediction framework which can handle sudden hike of the workload during peak times.

Sadeka *et al.* [13] use time-series analysis for adaptive resource allocation. A neural network algorithm was used to predict the future surge in resource demands for proactive scaling of the resources. The future utilization was predicted for varying sliding window sizes to assure the accurate predictions with respect of time. The accuracy was further tested by means of error correction methods for neural network algorithms. Resource scaling improved the performance and generated higher profit for the CSP.

B. Resource management schemes for batch workloads

The authors in [14] mention that batch jobs are generally delay tolerant up to a relatively loose deadline. The deadlines can be met until the fulfillment ratio is guaranteed. The fulfillment ratio is the ratio of execution time of a job to its scheduling duration. The prices charged by the provider are set according to the fulfillment ratio. The problem stated in their study addresses the lack of a cost effective solution for processing a batch of users with deadline guarantees. As a result, the authors propose a flexible instance which uses service fulfillment ratio as a pricing factor, and at the same time, guarantees that the jobs will be executed within a given deadline. The users are free to decide the fulfillment ratio and they will be charged accordingly. So as to automatically adapt resource prices to the demand supply relation, and maximize provider revenue, pricing schemes were derived from a well-designed pricing curve and the Nash bargaining solution, respectively. Moreover, the flexible instance allows providers to utilize the ideal resources, as users can ask to scale up the resources at a reasonable price. The results showed the increase in utilization, which eventually increased the net profit of the provider. Moreover, electricity cost has become a big concern for commercial cloud service providers with the rapid expansion of network-based cloud computing. Hence, Li *et al.* [15] addresses the issue of electricity expenses by limiting the delay in response by following the price-sensitive and cooling efficiency-enabled batch computing workload dispatch approach. The results of a Mixed Integer Programming (MIP)-based resource demand management solution show that aggregating the batch requests with similar deadlines can decrease the energy consumption by 30%. In the long run, it adds up to the saving of the CSP and increases the net profit.

C. Concluding remarks

Different resource management schemes for predicting the workload were discussed in the previous sub-sections however, most of the work amongst them is done for predicting the collective workload on the system. Furthermore, the resource management techniques using batch processing do not carry out prediction of user behavior to allocate resources. Instead, most of the researchers worked in the area of scheduling the jobs after the batch is processed. Additionally, the techniques using the batch workload showed an increase in CSPs profits by scaling the resources. However, none of the approaches increased the profit by considering the finite capacity environment of a CSP. Therefore, it becomes necessary to propose a resource management mechanism which maximizes the net profit in a finite capacity environment. This is the focus of attention for this paper along with the objective of maintaining the market value by maximizing the number of users to be processed in each batch, All in all, none of the existing works have dealt with all the aspects of the problems of concern of our study.

3. Linear Regression to Predict User Behavior

Many researchers have focused on the allocation problem while considering various aspects such as VM migration [16], VM consolidation [17], and QoS [18]. Besides that, studies related to resource estimation suggest that the amount of resources to be allocated could be estimated based upon the user's history. In the context of this work, as users keep using cloud services, it is assumed that cloud providers store information about the behavior of the users over time. Therefore, we use linear regression to accurately predict the user behavior.

Linear regression [19] is a form of supervised learning that involves predicting the value of a continuous output variable based on one or more input feature variables. In our work, we use a single input feature: history length of each user. Given a data set about the history of a user, one may want to predict his relinquish probability. More precisely, for an incoming user returning for the $(m+1)th$ time, a matrix is loaded with the m values from his history (history length). Here, predicting the relinquish probability as a function of the history length (feature) is a continuous output.

Two methods can be used to make predictions, namely the gradient descent and the normal equation method. The gradient descent method is generally used when there are multiple features and a large history (>100 k) [20]. On the other side, the normal equation method is efficient for a smaller history as it requires less computational time. Since users have a history length smaller than 100*k*, the normal equation method is used in our work. For more details on how the cost function is solved using these two methods, please refer to [20] [21].

4. Proposed Optimization Model

In this section, a mathematical model is proposed to optimize the net profit of cloud providers in an environment where users can leave the system at any point in time. The model is derived from the RACE (Relinquishment-Aware Cloud Economics) model that was published in [1]. As a reminder, the RACE model is used to calculate the overall net profit earned by the cloud providers. It is general enough to evaluate and compare different resource allocation schemes. For the proposed optimization model, the parameters used to calculate the net profit in [1] are reused and expanded to estimate the optimal amount of resources to assign to users with known history. The history represents the user behavior which is used to calculate the relinquish probability using the linear regression model discussed in the previous section.

To formulate the mathematical model, we assume the following:

- Users are able to determine and specify the amount of resources required to process and complete their jobs.
- Cloud providers store information about the behavior of their users over

time. This information is used to calculate the relinquish probability.

- Cloud providers know, in real-time, the remaining amount of resources (*i.e.* the remaining capacity), the amount of power used by their servers and the price of electricity in their region.

In the following sections, we present the notation, derive the profit function, and present the objective function and the constraints of the model.

A. Notation

1) Sets

- *J*, set of all resource types *j* where $j \in J$.
- *N*, set of users in a given batch where $n \in N$. It is assumed that users collectively arrive into the system and are divided into batches.
- *L*, set of geographical locations for the data centers in a cloud federation where $l \in L$.

2) Constants

- p_j, price for service $j \in J$.
- C_l, unit price of electricity at location $l \in L$ in $/kWh.
- P_l, power used to operate all the servers at location $l \in L$ when fully utilized.
- *V*, remaining capacity of the cloud when an incoming batch is processed.

3) Input Parameters

- R_j^n, amount of resources requested by user $n \in N$ for service $j \in J$.
- r^n, predicted value of the relinquish probability for user *n* (done through linear regression as discussed in the previous section).
- D^n, requested duration by user *n*.

4) Decision Variables

- a_j^n, amount of allocated resources for service $j \in J$ to user $n \in N$ of the batch.
- x_j^n, binary variable such that $x_j^n = 1$ if and only if user $n \in N$ in the batch is allocated resources for service $j \in J$, otherwise $x_j^n = 0$ if the request is not processed.

B. Objective Function

The aim of the objective function is to maximize the net profit by allocating the optimal amount of resources while considering the current available capacity of the resource pool, the relinquish probability, and the arrival rate of the requests. The objective function is first divided into three different parts: income, electricity expenses and relinquishment loss.

The first term of the objective function, shown in Equation (3), calculates the expected income generated by user *n*. When users arrive, they request resources for a specific duration D^n. At the same time, the cloud provider will look at the history of the users and will predict their relinquish probability (r^n). Then, the users will be assigned ressources (a_j^n) and will be charged according to the price of ressources (p_j).

$$Income_n = \sum_{j \in J} p_j \left(1 - r^n\right) D^n a_j^n \tag{1}$$

The second term, shown in Equation (2), represents the electricity expenses generated by user *n* (EE_n) for the utilized duration $D^n\left(1-r^n\right)$ and the idle duration $D^n r^n$. Both are calculated based on the predicted relinquish probability.

$$EE_n = \sum_{l\in L} C_l P_l \left(\sum_{j\in J} \left(D^n\left(1-r^n\right)u_j^n + \frac{2}{3} D^n r^n \left(1-u_j^n\right) \right) \right) \tag{2}$$

where

$$u_j^n = \frac{a_j^n}{\text{total resources}} \tag{3}$$

The third term, shown in Equation (4), calculates the relinquishment loss for a given user (RL_n). The model uses the predicted relinquish probability to calculate the relinquishment loss for the duration ($r^n \cdot D^n$) *i.e.* the duration for which the user would not use the service.

$$RL_n = \sum_{j\in J} D^n p_j a_j^n r^n \tag{4}$$

Hence, the overall profit function for a given user *n*, denoted $P(r)_n$, can be represented as:

$$P(r)_n = Income_n - EE_n - RL_n \tag{5}$$

Moreover, to maintain a good reputation, cloud providers always want to process the maximum number of user requests. Therefore, an additional function is added to reflect this objective. In conclusion, the final objective function, shown in Equation (6), simultaneously maximizes the profit and the number of requests served.

$$\max\left(\sum_{n\in N} \left(P(r)_n + \sum_{j\in J} x_j^n \right) \right) \tag{6}$$

C. Constraints

The objective function shown in Equation (6) is subject to different constraints expressed in Equations (7) to (9).

$$\sum_{n\in N} a_j^n \le V \quad \left(j\in J\right) \tag{7}$$

$$a_j^n \le R_j^n \quad \left(j\in J, n\in N\right) \tag{8}$$

$$a_j^n \ge \beta_j\left(x_j^n R_j^n\right) \quad \left(j\in J, n\in N\right) \tag{9}$$

Equation (7) specifies that the sum of all allocated resources should be less than or equal to the remaining capacity of the system (*V*). Additionally, as we try to maximize profit, it is important not to overestimate the amount of resources. Therefore, Equation (8) specifies that the amount of allocated resources should be less than or equal to the requested amount of resources by the user. Moreover, since different services have different minimum requirements, Equation (9) ensures that at least the minimum amount of resources to run the service will be provided. In our work, and without loss of generality, we used $\beta_j = 0.2$ although this value can be tuned to the requirements of the different services. A typical example could be a user requesting a 4 k videos. If, for any reasons (e.g. unreliable user, not enough remaining capacity, etc.), the algorithm decides to not allocate the requested amount of resources, then in order to be able to see the video, the user should at least get enough resourced to play the standard definition (SD) version (which requires less resources) of the video.

Based on the equations and the constraints mentioned above, the model finds the optimal amount of resources to be assigned (a_j^n) to each user in the batch. It is important to note that the calculations of the model are mainly dependent upon the predicted relinquish probabilities as all other input parameters such as the amount of requested resources, the duration and the arrival rate of users are kept constant.

5. Performance Evaluation

In this section, we compare the performance of the proposed model against other models from the literature. The simulations of all the proposed algorithms were implemented in Java using the CloudSim toolkit [22]. The tools (simulator and computer) used during the simulations are outlined in **Table 1**. The simulations are carried with fixed resource pool of the CSP in an environment where users can relinquish. Our results are deduced based upon the pricing of utilizing an On-Demand windows-based general purpose t2.medium vCPU from Amazon Web Services in Canada (Central) on a usual business day [23]. Besides, the prices of electricity vary during different times of the day. Therefore, the mean price of the electricity of summers in Ontario is considered [24].

A. Generating the workload

To investigate the effectiveness of all the models, we generate a synthetic workload. The most important input parameter to generate the workload is user behavior. **Figure 1** depicts the process of generating and populating the history of the users. By definition, the relinquish probability is the ratio between the duration for which the user has not used the service and the total requested duration. More specifically, the value of the Average Relinquish Probability (ARP) gives an idea about the general behavior of the user who is requesting the resources based on how much, on average, it has used in the past. A user with an ARP ≥ 0.7 is considered disloyal whereas a user with an ARP ≤ 0.3 is considered loyal. Users that fall between 0.3 and 0.7 are considered average users. Therefore, to obtain realistic ARP data, a mean of 0.5 and a standard deviation (SD) of 0.3

Table 1. Simulation environment.

Operating System	Ubuntu 16.04 LTS
Processor	Intel i7 - 3.6 GHz
Memory	16 GB
Simulator	CloudSim 4.0
Implementation Language	Java SE 7

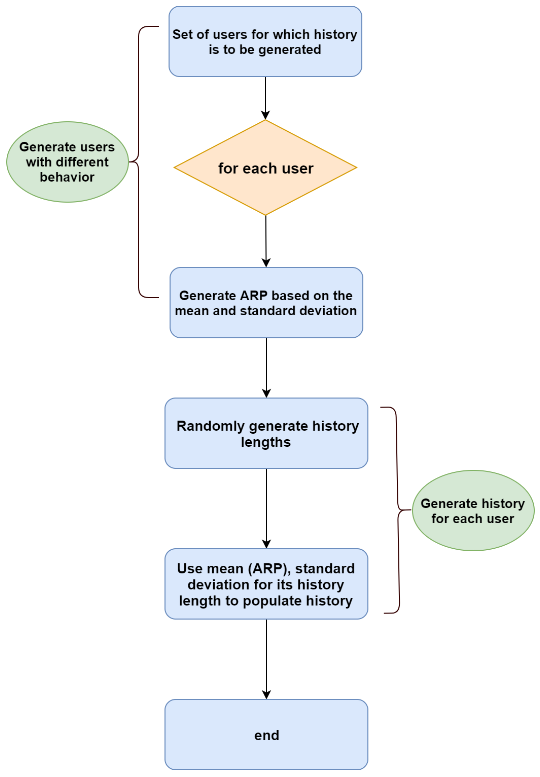

Set of users for which history is to be generated
for each user
Generate ARP based on the mean and standard deviation
Generate users with different behavior
Randomly generate history lengths
Use mean (ARP), standard deviation for its history length to populate history
Generate history for each user
end

Figure 1. Process to generate history of users.

are used to generate values using the normal distribution. Values greater than one are considered as 1 and values less than one are given the value 0.

Using ARPs to populate the history: A user can be a first-time user or could have used the service multiple times. In this study, the history length of each user is set between $U(1,60)$. The first-time users are treated as loyal users. For returning users, based upon the history length of each user (the number of time the user is returning for), the corresponding value of the user's ARP is used as the mean for the normal distribution to populate the relinquish probabilities for each time the user relinquished. For example, if a user is returning for the 5th time and its ARP is 0.59, then using a mean of 0.59 and a standard deviation of 0.2, its usage history is populated for the last 4 times.

To check the reliability, the model is tested with different arrival rates of users. In this study, we assume that users submit requests based on the M/M/1 queuing model. The set of arrival rates (λ) used to test the models is {0.008, 0.016, 0.032, 0.064} req/sec. Hence, the interarrival times of the requests are generated randomly using the exponential distribution with mean $E[x]=1/\lambda$ secs. For simplicity and without loss of generality, we assume that the user requests a single service type, that is, vCPU with a minimum of 500 GHz and a maximum of 900 GHz for a specific duration between the limit of 2 - 4 hours.

The input parameters used are user history, interarrival time of user requests, start time of the users, end time of the users, amount of requested resources by the users, requested duration by users and used duration by the users. Table 2

Table 2. Parameters used to evaluate the proposed optimization model.

Arrival rate of users	0.008, 0.016, 0.032, 0.064
History length of users	$U(1,60)$
Total size of resource pool	Fixed at 200k
Power of server	1 server of 525W
Total duration	25 hours
Requested duration by user	$U(2,4)$ hours
Requested resources by user	$U(500,900)$
Requested service type by user	vCPU
On-demand price of service	CAD 0.1/hour
Total no of users arrived	750, 1450, 2700, 5500
Mean electricity price in Ontario in summers	CAD 0.132/kwh
Schedule end time of a user	Sum of start time and requested duration of respective user
Relinquish duration of a user	Relinquish probability times the actual requested duration.

summarizes all the input parameters used during the simulations. The start time of the user is the time when the user begins using the resources and start getting charged for the service. The difference between two start times is the interarrival duration generated by the exponential distribution described earlier. The requested durations and the requested resources of the users are randomly generated within the limits stated in Table 2. The current relinquish probability is still generated by the normal distribution. The scheduled end time of a user is the sum of its start time and the requested duration. If the user relinquishes, the used duration of the user is the requested duration times the current relinquish probability. The relinquish time then becomes the sum of the start time and the relinquish duration. Else, if the user does not relinquish (*i.e.* the relinquishing probability is zero), then the requested duration is considered as the duration used by the user.

B. Evaluation of various methods to predict relinquish probability

This section presents the comparison results of the various prediction techniques used to predict the user behavior given their history. The outcomes of this section will tell us which method can make better predictions about the user behavior. The prediction will let the provider know how long the user is going to use the allocated resources based on his history. To perform the evaluation, Java and Octave platforms were used to generate the scripts, data sets and carry out the calculations respectively.

In this section, we use linear regression and compare the results with the two methods mentioned in [6] and [7]. Table 3 provides an example of an average user whose history length (m) is five. The data in Table 3 shows the usage history of the user for the last five times. Similar kind of data was used to carry out a separate univariate regression analysis for a single user returning with various history lengths.

To evaluate the performance of the different methods, we used three user categories: loyal, disloyal and average. To simulate loyal, disloyal, and average users, different ARPs (0.3, 0.7 and 0.5 respectively) were used and the history was populated randomly using the normal distribution with these ARPs as mean and a standard deviation of 0.2. As illustrated in Figure 2, seven different loyal users were simulated with different history lengths (*i.e.* $m = 5$, 10, 20, 40, 60, 80 and 100). Each user has ARP = 0.3 with different history lengths. To get reliable results and avoid randomness, the process of generating history and making prediction for a single user was repeated 50 times. Then, the mean of the predicted values is taken and plotted along with the 95% confidence interval in Figures 2-4. A similar process is followed for the other two types of users (average and disloyal). Further, the history length is denoted as m and the $(m+1)th$ value is plotted for each prediction approach. For instance, if history length is $m=5$, it states that the user is returning for the $(m+1)th$ (*i.e.* 6th) time. The relinquish probability for the $(m+1)th$ time is plotted (*i.e.* each user is returning for the 6th, 11th, 21st, 41st, 61st, 81st and 101st time respectively). As seen from the graphs, predicted values from linear regression (LR), BaaS [1] and RP [7] are compared with the observed relinquish probability for $(m+1)th$ time.

Table 3. Example of an average user with $m = 5$.

History Length ($m = 5$)	Relinquish Probability
1	0.35
2	0.56
3	0.28
4	0.67
5	0.45
6 ($m + 1$)	To be predicted by LR

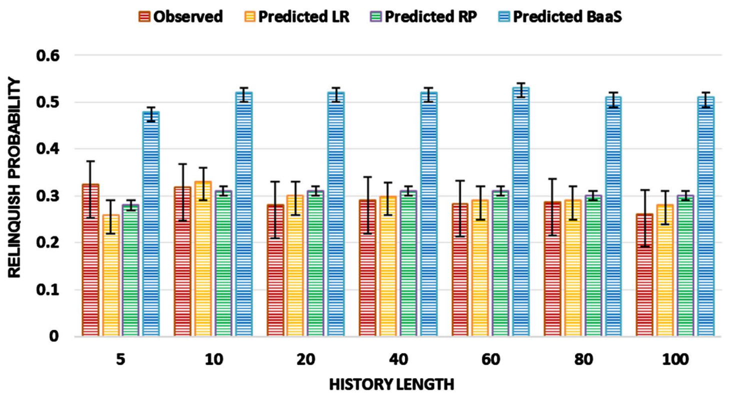


Figure 2. Prediction results for loyal users (ARP ≤ 0.3).

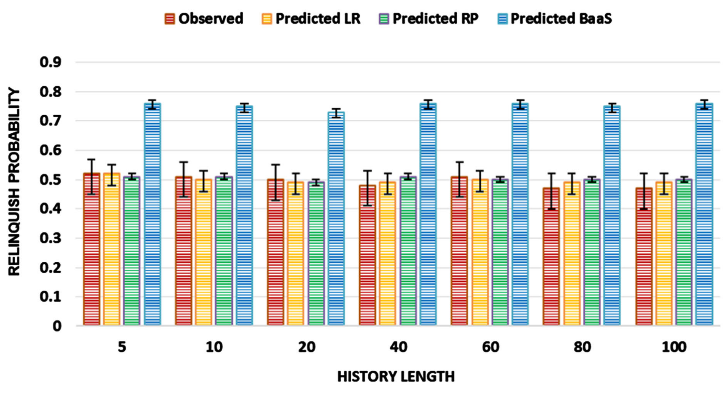


Figure 3. Prediction results for average users (0.3 < ARP < 0.7).

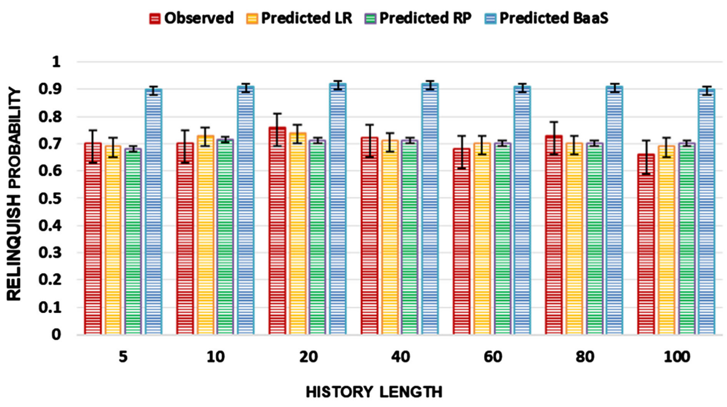


Figure 4. Prediction results for disloyal users (ARP ≥ 0.7).

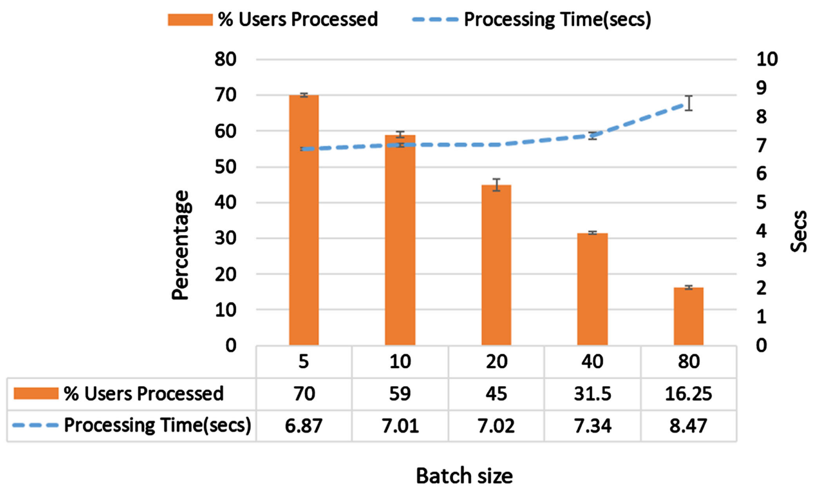
To evaluate the prediction accuracy of the three methods, we compare the predicted value with the actual observed value from the user. When comparing the observed relinquish probability with the predicted relinquish probability, we can evaluate the accuracy of the various prediction methods. The observed relinquish probability is the real amount of resources that was used over the amount of resources that was initially requested. It can be seen that BaaS shows the biggest difference (around 20%, 25% and 20%) between the observed value and the predicted value for loyal, average and disloyal users respectively. This is because the authors use SOP and AOP for predicting the user behavior. Furthermore, to cope with changes in the behavior of users each time they return, the authors use the variance as an additional parameter, but this does not help in making the results more accurate. On the other hand, RP (which is the improved version of BaaS) is able to make better predictions with a difference of approximately 5%, 4% and 4% between the observed value and the predicted value for loyal, average and disloyal users respectively. Moreover, for small training sets (*i.e.* history lengths ≤ 20), the prediction method used in [7] provides better results (approximately 5% better) compared to linear regression. However, as the history length grows, the accuracy of linear regression outperforms the predictions made by the RP method by approximately 2%, 1.7%, 1% for loyal, average and disloyal users respectively. The major reason for the improved accuracy of the linear regression algorithm can be regarded as the minimized squared error function which provides the minimum value of the squared difference between the errors. In general, even with a single parameter (history length), linear regression proves to be the most effective tool amongst the three to make predictions about the user behavior. As a result, linear regression is used to make predictions as improving predictions can also improve the profit generated by the provider.

C. Selecting the batch size for the proposed optimization model

This section presents and discusses the experimental results of the relinquishment aware resource optimization model. This model is solved with IBM CPLEX [25] which is a well-known tool used in Operations Research (OR) for solving optimization problems. For a given batch size and resource capacity, the performance of the model is tested in terms of two parameters. The resultant values of the parameters are used to decide which batch size gives the best results in terms of processing maximum user requests under the finite capacity. The parameters are:

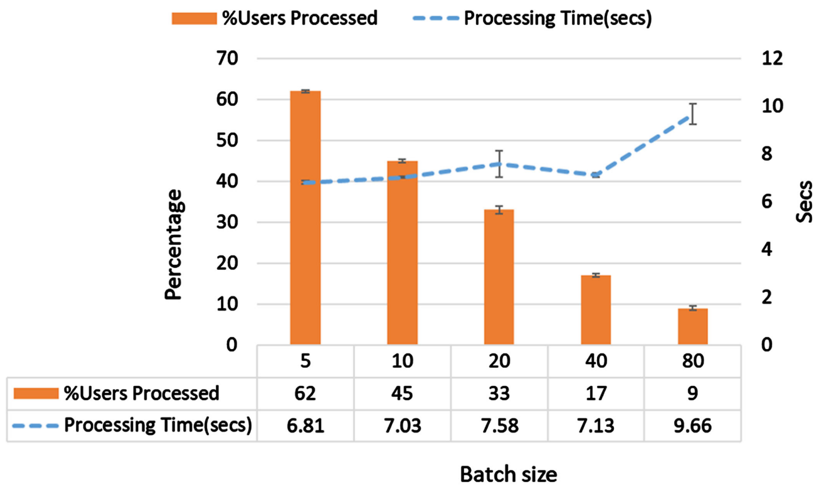
- *Processing time of a workload batch*: The time required by the solver (CPLEX) to read the input data, generate the model, solve the model and produce the output in the form of decision variables.
- *Number of users processed in a batch*: This parameter gives us the percentage of users processed for the batch size under evaluation with the given capacity of resources.

The batch sizes used in this study are based upon the workload batch sizes used by other researchers. For example, the study that was done in [26] considers a batch of 5 requests to be of small size whereas a batch with 10 jobs is considered as medium. Therefore, to have a complete analysis, batch sizes of 5, 10, 20, 40, and 80 requests are tested under different remaining capacities of 2000, 4000 and 8000 vCPUs respectively. The input parameters of the model are predicted relinquish probability, requested duration, requested amount of resources, price of resource and electricity. The experiment for each set of batch size with individual capacity value was run 20 times. The mean values of the percentage of users processed and the CPU processing times taken by the solver are plotted along with the 95% confidence intervals. The built-in timer provided by CPLEX is used to measure the processing time. **Figure 5** depicts the results when the model was tested with a remaining capacity of 8000. Based on the predicted relinquish probability and the input parameters, the solver only allocates resources to users who would potentially generate more income for the provider and minimize the relinquish loss at the same time. Simultaneously, the solver also tends to maximize the number of users to be processed within that batch. As expected, the batch with the large workload of 80 user requests takes the longest time to process all the inputs and provides results in around 8 seconds. The small and the medium sized batches are processed in almost the same amount of time. Moreover, for small and medium workloads, around 70% of the users were processed with the least processing times. In **Figure 6**, the remaining capacity



	5	10	20	40	80
% Users Processed	70	59	45	31.5	16.25
Processing Time(secs)	6.87	7.01	7.02	7.34	8.47

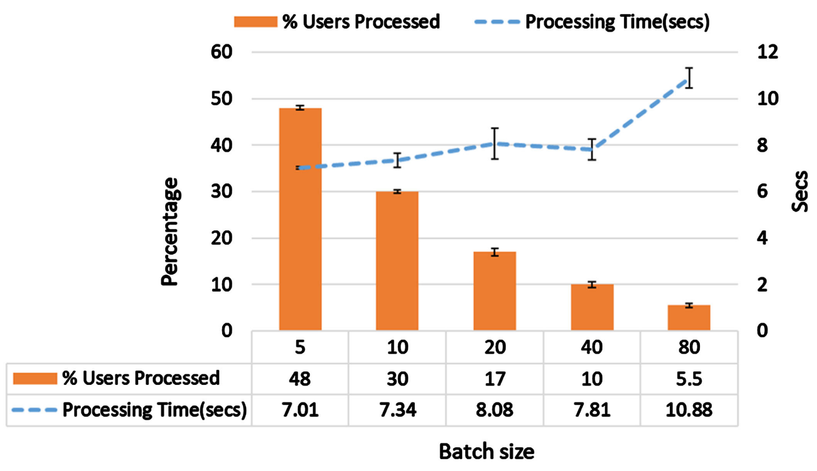
Figure 5. Percentage of users processed and processing times for different batch sizes with a remaining capacity of 8000 resources.



	5	10	20	40	80
%Users Processed	62	45	33	17	9
Processing Time(secs)	6.81	7.03	7.58	7.13	9.66

Figure 6. Percentage of users processed and processing times for different batch sizes with a remaining capacity of 4000 resources.

was reduced by half to 4000 resources. Small and medium workloads performed the best in terms of percentage of users processed and time taken to process the batch. Whereas, the least percentage of users was processed in the large workload with a batch size of 80. This is because the system runs out of capacity for larger batch sizes as the sum of the requested resources by the entire batch greatly exceeds the available capacity. The reason for this is the set limits of the requested amount of resources for the user in Table 2. Hence, the percentage of users processed keeps decreasing with an increase in the batch size for a given finite capacity. To understand this phenomenon, the batch was finally tested with a remaining capacity of 2000. As shown in Figure 7, when the batch size is increased, the percentage of users processed keeps decreasing while the processing time keeps increasing accordingly. As usual, the small batch size shows the best performance in terms of processing time and percentage of users processed.



Batch size	5	10	20	40	80
% Users Processed	48	30	17	10	5.5
Processing Time(secs)	7.01	7.34	8.08	7.81	10.88

Figure 7. Percentage of users processed and processing times for different batch sizes with a remaining capacity of 2000 resources.

To summarize, the experimental results demonstrate that a batch size with five user requests performs better than all the other batch sizes under various remaining capacities. The objective of using different batch sizes under varying capacity was to figure out the optimal batch size which could be used for the simulations where users enter the system in an open stream and the remaining capacity is constantly updated. Therefore, a batch size of five will be used in the next section to assess the performance of the proposed optimization model and carry out the comparison between state-of-art algorithms.

D. Evaluation of the proposed optimization model versus state-of-art algorithms

To carry out the evaluation, the proposed optimization model was integrated into the DatacenterBroker class of CloudSim, along with all the scenarios mentioned in [5], by importing the Java libraries provided with the CPLEX package to call the various CPLEX functions. In addition, the predictions were made by writing the linear regression algorithm in Java and integrating it into the DatacenterBroker class of CloudSim. Following the predictions, the model collectively decides the amount of resources to be allocated to all the requests in the batch and simultaneously updates the current capacity available for the next batch of requests. The outcomes are then compared with the related work. In our previous work, the RACE model was analyzed using four different scenarios. We use the same scenarios (mentioned below for completeness) to evaluate the proposed optimization model.

- In Scenario 1, users are assigned resources based on the service they requested. In other words, complex resource estimation is not required since users get exactly what they requested and they do not relinquish.
- Scenario 2 is similar to Scenario 1, except for the fact that users can relinquish before the scheduled end time.
- In Scenario 3, resources are assigned based on the BaaS method described in [6]. In the BaaS model, the main idea is to assign less resources to disloyal

users.

- In Scenario 4, resources are assigned based on the RP method mentioned in [7]. In this study, resources are allocated based on the predicted user behavior and the current server utilization.

Furthermore, we term our proposed optimization model as Scenario 5. To compare Scenario 5 with the other four scenarios, all models were run for a specified duration of 25 hours. It can be noted that 25 hours is not the actual clock time but the pre-configured simulation time on CloudSim for which the models are compared at different arrival rates. The workload was generated using the same input parameters mentioned in Table 2 and each simulation was repeated 20 times. The confidence intervals were also calculated but since they were very small, they are not shown in the graphs. The overview of the simulation for a batch is shown in Figure 8. When the requests start arriving (unlike Scenarios 3 and 4 where users are processed one by one), we assume that all users in the incoming batch do not have a job completion deadline. This assumption is necessary since when user requests arrive in the system, they are queued in the network until the required batch size is formed (step 1). Then, the cloudlet submits the request to the broker (step 2). The broker asks for the history of each user (step 3a) and gets the history of each user from the host (step 3b). The history is fed to the prediction module where the three different prediction algorithms (depicted in Figure 9) are executed (step 4a). Then, the prediction values are passed to the resource assignment module (step 4b). The resource assignment schemes in this module (as shown in Figure 9) use the predicted value to estimate the amount of resources. This information is passed to the broker (step 4c).

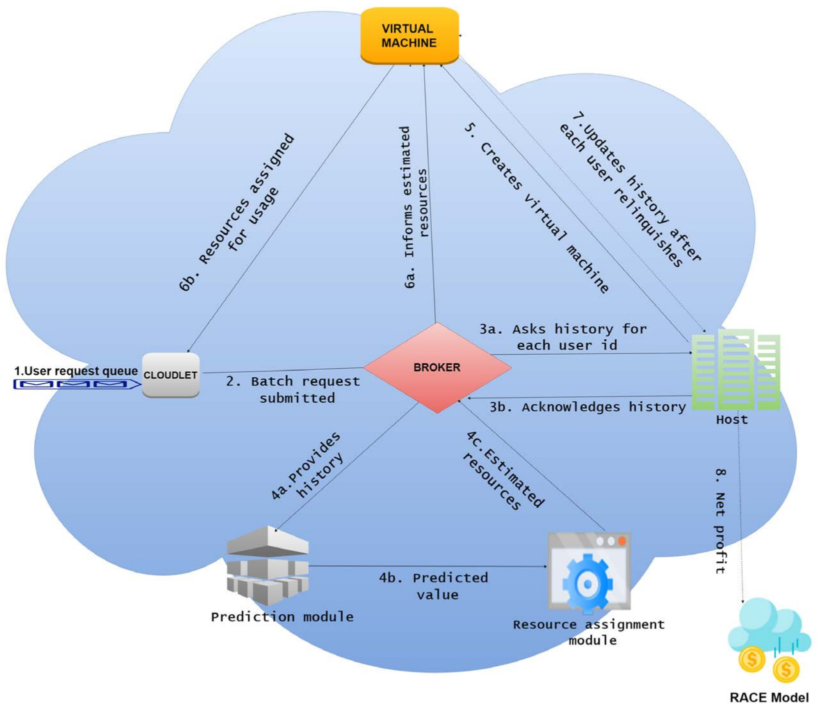

VIRTUAL MACHINE
6b. Resources assigned for usage
6a. Informs estimated resources
5. Creates virtual machine
7.Updates history after each user relinquishes
3a. Asks history for each user id
1.User request queue
CLOUDLET
2. Batch request submitted
BROKER
3b. Acknowledges history
Host
4a.Provides history
4c.Estimated resources
8. Net profit
4b. Predicted value
Prediction module
Resource assignment module
RACE Model

Figure 8. Steps involved to perform the simulation in CloudSim.

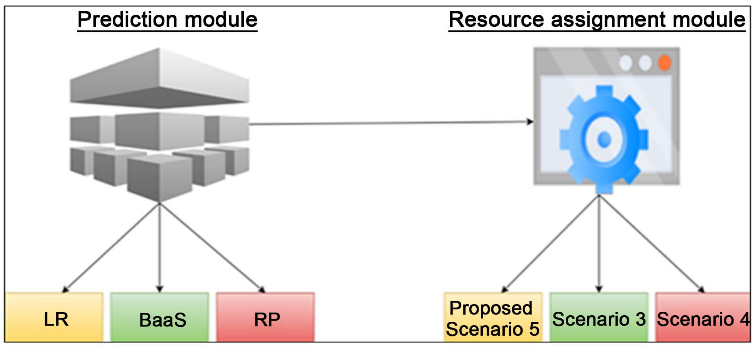


Figure 9. Components of the different predictions modules used in the comparison.

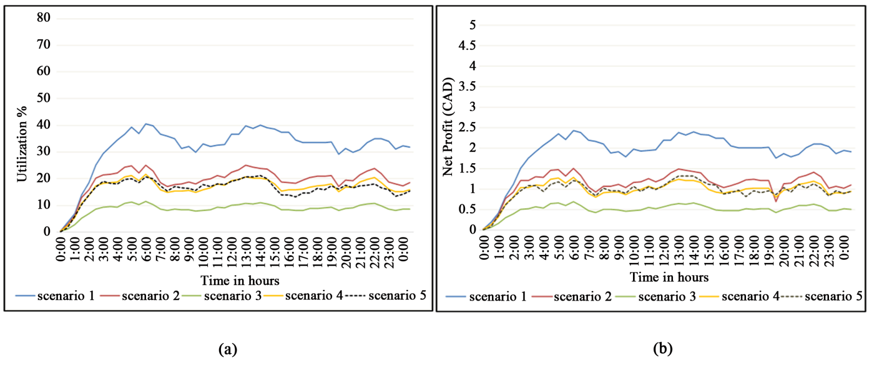


Figure 10. Utilization (a) vs net profit (b) for all scenarios at $\lambda = 0.008$ req/sec.

The host creates a virtual machine (step 5) which further processes the request from broker (step 6a) to assign the resources to the users (step 6b). It is important to note that before every batch is processed, the value of the current remaining capacity of the system is passed into the resource assignment module. It is also assumed that there is no processing delay experienced once the batch enters the system. The remaining capacity is then updated and users are blocked if the system is running at maximum capacity. Finally, the history of each user is updated with a new entry once the user leaves the system (step 7).

The major outcomes for the comparisons of the Average Overall Utilization (AOU) and the Overall Net Profit (ONP) are outlined in Table 4. For a given arrival rate, each scenario was simulated 20 times. Therefore, to calculate the AOU and ONP, we simply took the average over the 20 different simulations. As demonstrated in Figure 10(a) and Figure 10(b), with a total of only around 750 users entering at $\lambda = 0.008$ req/sec, the system is underutilized for all the scenarios. It can be seen that, from time 14:00 to 16:00 when utilization is dropping due to users relinquishing, the impact of relinquishment of users on the net profit of Scenario 5 is less as compared to other scenarios. This is because in Scenario 5, if within a batch, users are predicted to relinquish, loyal users are given the greater share of the resources whereas disloyal users are given either 20% share, or are declined depending on how they are estimated to make profit. Therefore, at time 16:00, with the least utilization of Scenario 5, the corresponding net profit is greater than Scenario 4 and almost comparable to Scenario 2. It can also be noticed that for each scenario, the net profit at each time is generally reflected by the utilization, but at some points, for example from time 20:00 to 23:00, the net profit for Scenario 5 is not identical to the line of the corresponding utilization. This would likely be due to some instances where the linear regression estimates a larger amount of resources for some loyal users but they relinquish before their predicted duration of using the resources. This anomaly in prediction could be due to the fact that the history length of these users was too small for the linear regression algorithm to show its effectiveness. However, the improved performance of the prediction technique and Scenario 5 was seen with the increase in arrival rates of the users.

In the next experiment, the arrival rate was doubled to $\lambda = 0.016$ req/sec and 1450 users entered the system in this case. From Figure 11(a) and Figure 11(b), it is observed that, the net profit and the utilization curves for Scenario 5 are higher than Scenario 4. This is unlike the previous case when the arrival rate of users was smaller. As discussed earlier, this improvement can be attributed to the approach of Scenario 5 that assigns more resources to loyal users and tightens up for predicted disloyal users. Therefore, the difference between the income and the overall expenses is minimized and the Overall Net Profit (ONP) for Scenario 5 (outlined in Table 4) is increased significantly as compared to Scenario 4. Further, as seen in Figure 12(a) and Figure 12(b) when the arrival rate was set to $\lambda = 0.032$ req/sec, the performance of the proposed optimization model increases as the difference between the net profit of Scenario 5 and Scenario 2 decreases. The Average Overall Utilizations (AOUs) and ONPs in Table 4 of Scenarios 2 and 5 show that with comparative profits for both scenarios, a provider using Scenario 5 can still acquire more requests. Therefore, with Scenario 1 running almost at full capacity and Scenario 2 operating at nearly full capacity, it is most likely that Scenario 5 can yield a substantial increase in profit. To see that, the simulations were finally run for $\lambda = 0.064$ req/sec. As it is clearly seen in Figure 13(a) and Figure 13(b), for most of the times, the utilization rate of Scenarios 2 and 5 remains stable as requests continuously keep on arriving

Table 4. Major outcomes for the comparisons of AOU, ONP and P_b for different arrival rates.

	Arrival Rate											
	$\lambda = 0.008$			$\lambda = 0.016$			$\lambda = 0.032$			$\lambda = 0.064$		
	AOU (%)	ONP (CAD)	P_b	AOU (%)	ONP (CAD)	P_b	AOU (%)	ONP (CAD)	P_b	AOU (%)	ONP (CAD)	P_b
Scenario 1	28.7	189 k	0	57.7	357 k	0	84.7	537 k	0.4	88.2	545 k	0.54
Scenario 2	17.4	108 k	0	35.3	209 k	0	68.7	418 k	0	84.4	505 k	0.4
Scenario 3	7.8	47 k	0	15.7	94 k	0	30.8	189 k	0	59.2	370 k	0
Scenario 4	14.7	91 k	0	22.6	139 k	0	41.2	251 k	0	66.2	403 k	0
Scenario 5	14.4	93 k	0	28.3	177 k	0	55.8	349 k	0	85.1	512 k	0.32

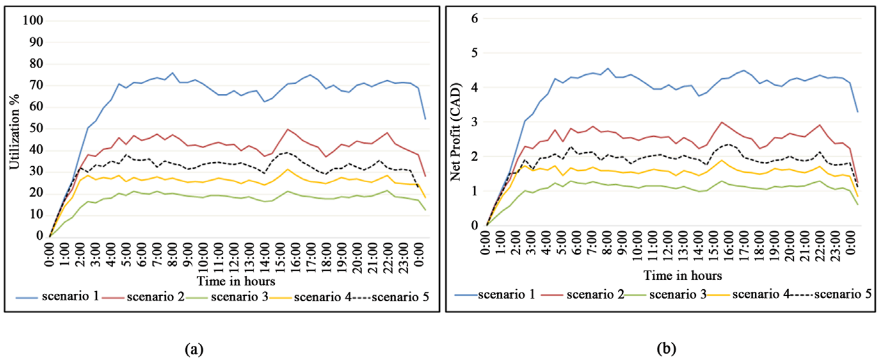


Figure 11. Utilization (a) vs net profit (b) for all scenarios at $\lambda = 0.016$ req/sec.

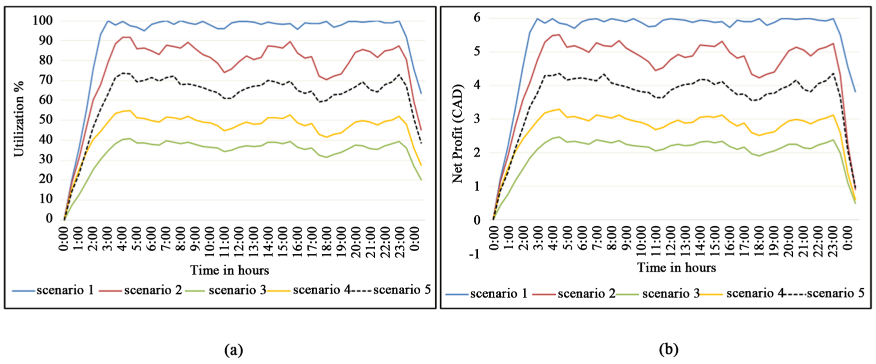


Figure 12. Utilization (a) vs net profit (b) for all scenarios at $\lambda = 0.032$ req/sec.

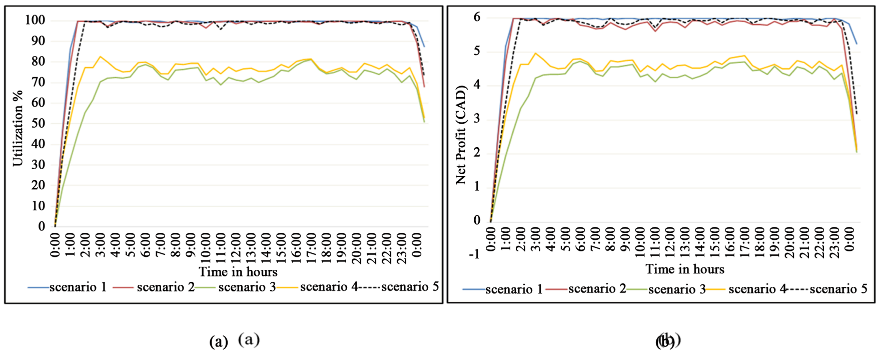


Figure 13. Utilization (a) vs net profit (b) for all scenarios at $\lambda = 0.064$ req/sec.

but the corresponding net profit varies due to users relinquishing their service.

Moreover, because of the high incoming stream of users (around 5500 in 25 hours), the available capacity remains contented most of the time as servers run at high utilization. Noticeably in Figure 13(a) and Figure 13(b), from time 4:00 till the end, Scenario 5 generates slightly greater profit than Scenario 2 which leads to a greater ONP for Scenario 5. As discussed earlier, this is because the optimization model used in Scenario 5 uses linear regression to predict the behavior of users and intelligently assigns the resources so that the impact of relinquishment of users is minimized.

Thus, as per the deductions of [1], the objective of maximizing profit is achieved when resources in a finite resource pool get contented with users relinquishing their services. In addition to that, though the proposed model tends to process the maximum number of users, the predicted loyal users within the batch are given more priority in terms of allocating the share of remaining capacity to maximize profit. Consequently, some disloyal or average users were denied the service. Interestingly, as compared to its counterparts from [1], the AOU has also improved in all the cases for Scenario 5. On the other hand, when the resources are fully utilized at high arrival rates, many user requests were also blocked in some of the scenarios. Since the arrival of users does not depend upon the server capacity or total available resources, the blocking ratio varies and depends upon the arrival rates of the incoming requests.

As [2], this study concentrates on the economic aspects of cloud computing. Therefore, from a business point of view, it is important to know the blocking rate as denying the service leaves a negative impact on the users and deteriorates the market-value of the CSP. Therefore, the blocking probability gives another piece of information to the CSP on evaluating the performance of the resource allocation policy being used. The blocking probability (P_b) is calculated as:

$$P_b = \frac{\text{Number of users blocked due to full capacity}}{\text{Total number of users arrived}} \tag{10}$$

The blocking probabilities for all scenarios are outlined in **Table 4**. It can be clearly seen that no users were blocked for lower arrival rates. Noticeably, for $\lambda = 0.064$ req/sec, the blocking probability of Scenario 5 is less than Scenario 2 even with a slightly higher AOU as Scenario 5 makes better resource assignment decisions during the contention of resources. Additionally, the proposed model shows an improved utilization at higher arrival rates as compared to the similar models proposed in the literature. In a nutshell, it can be said that rather than assigning whatever is requested or using approximate algorithms, providers must assign resources using the proposed optimization technique to maximize their net profit.

6. Conclusions

Since users can relinquish their service at any point in time, it is essential to have ways in which cloud providers can predict the behavior of users. Different ways were proposed in the literature but they were mainly using some sort of average based on the historical behavior of users. Seeing this, the idea was to figure out a better prediction technique such that CSPs could make accurate decisions while estimating the amount of resources to be allocated. As a result, we developed a prediction algorithm based on the concept of linear regression. Users with different usage behaviors (loyal, average and disloyal) were evaluated for different history lengths. The results showed that the technique based on linear regression outperformed all the user behavior prediction techniques proposed in the literature. Lastly, a relinquishment aware resource optimization model was also proposed to optimally estimate the amount of resources such that the net profit of

CSPs can be maximized. The model uses the prediction results from the linear regression technique to make the optimal resource estimation. To carry out the experimental results, variable user batch sizes were tested for the processing times and the number of requests processed per batch. The performance analysis carried out in CPLEX depicts that a batch size of five user requests took the minimum processing time while processing the maximum percentage of requests per batch. Furthermore, the proposed optimization model is also compared against two algorithms from the literature. The simulations are implemented in CloudSim to carry out the comparison. The results of the simulations show that the optimization model generates the maximum profit when resource contention occurs at high user arrival rate.

To the best of our knowledge, since this study is unique, there are many issues that can be addressed to extend this work. Linear regression was modeled using only a single feature: relinquish probability. We believe that including additional features such as the profit generated from the users at each instance could potentially improve the performance of the linear regression algorithm. In this paper, the resource price of Amazon EC2 instance was used. In the future, the price of resources could also be predicted based upon the user history. This can further help CSPs to increase the net profit and compensate the relinquishment loss. Further, based upon the new pricing strategy, a reimbursement strategy can be added to have a complete pricing model.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Singh, S., Aazam, M. and St-Hilaire, M. (2017) RACE: Relinquishment-Aware Cloud Economics Model. *24th International Conference on Telecommunications (ICT)*, Limassol, 3-5 May 2017, 1-6. https://doi.org/10.1109/ICT.2017.7998279

[2] Alhamad, M., Dillon, T. and Chang, E. (2010) Conceptual SLA Framework for Cloud Computing. *4th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, Dubai, 13-16 April 2010, 606-610. https://doi.org/10.1109/DEST.2010.5610586

[3] Zhang, S., Yuan, D., Pan, L., Liu, S., Cui, L. and Meng, X. (2017) Selling Reserved Instances through Pay-as-You-Go Model in Cloud Computing. *IEEE International Conference on Web Services (ICWS)*, Honolulu, 25-30 June 2017, 130-137. https://doi.org/10.1109/ICWS.2017.25

[4] Calheiros, R.N. and Buyya, R. (2012) Cost-Effective Provisioning and Scheduling of Deadline-Constrained Applications in Hybrid Clouds. *International Conference on Web Information Systems Engineering*, 171-184. https://doi.org/10.1007/978-3-642-35063-4_13

[5] Zhang, S., Qian, Z., Luo, Z., Wu, J. and Lu, S. (2016) Burstinessaware Resource Reservation for Server Consolidation in Computing Clouds. *IEEE Transactions on Parallel and Distributed Systems*, **27**, 964-977.

https://doi.org/10.1109/TPDS.2015.2425403

[6] Aazam, M. and Huh, E.-N. (2014) Broker as a Service (BaaS) Pricing and Resource Estimation Model. *IEEE 6th International Conference on Cloud Computing Technology and Science* (*Cloud-Com*), Singapore, 15-18 December 2014, 463-467. https://doi.org/10.1109/CloudCom.2014.57

[7] Hu, Q. (2016) Reactive Prediction Models for Cloud Resource Estimation. Master's Thesis, Carleton University, Ottawa.

[8] Hu, Q., Aazam, M. and St-Hilaire, M. (2018) Cloud Resource Allocation Based on Historical Records: An Analysis of Different Resource Estimation Functions. 2018 *World Congress on Services* (*Services* 2018), Seattle, June 2018, 118-129.

[9] Pandey, N.K., Chaudhary, S. and Joshi, N.K. (2016) Resource Allocation Strategies Used in Cloud Computing: A Critical Analysis. 2*nd International Conference on Communication Control and Intelligent Systems* (*CCIS*), Mathura, 18-20 November 2016, 213-216.

[10] Farahnakian, F., Liljeberg, P. and Plosila, J. (2013) LiRCUP: Linear Regression Based CPU Usage Prediction Algorithm for Live Migration of Virtual Machines in Data Centers. 39*th Euromicro Conference on Software Engineering and Advanced Applications*, Santander, 4-6 September 2013, 357-364. https://doi.org/10.1109/SEAA.2013.23

[11] Moreno, I.S., Garraghan, P., Townend, P. and Xu, J. (2013) An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models. *IEEE 7th International Symposium on Service Oriented System Engineering* (*SOSE*), Redwood City, 25-28 March 2013, 49-60. https://doi.org/10.1109/SOSE.2013.24

[12] Fang, W., Lu, Z., Wu, J. and Cao, Z. (2012) RPPS: A Novel Resource Prediction and Provisioning Scheme in Cloud Data Center. 2012 *IEEE Ninth International Conference on Services Computing* (*SCC*), Honolulu, 24-29 June 2012, 609-616. https://doi.org/10.1109/SCC.2012.47

[13] Islam, S., Keung, J., Lee, K. and Liu, A. (2012) Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud. *Future Generation Computer Systems*, **28**, 155-162. https://doi.org/10.1016/j.future.2011.05.027

[14] Yi, X., Liu, F., Li, Z. and Jin, H. (2016) Flexible Instance: Meeting Deadlines of Delay Tolerant Jobs in the Cloud with Dynamic Pricing. *IEEE 36th International Conference on Distributed Computing Systems* (*ICDCS*), Nara, 27-30 June 2016, 415-424. https://doi.org/10.1109/ICDCS.2016.35

[15] Li, J., Bao, Z. and Li, Z. (2015) Modeling Demand Response Capability by Internet Data Centers Processing Batch Computing Jobs. *IEEE Transactions on Smart Grid*, **6**, 737-747. https://doi.org/10.1109/TSG.2014.2363583

[16] Chen, J., Qin, Y., Ye, Y. and Tang, Z. (2015) A Live Migration Algorithm for Virtual Machine in a Cloud Computing Environment. *IEEE 12th International Conference on Ubiquitous Intelligence and Computing*, Beijing, 10-14 August 2015, 1319-1326. https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.239

[17] Kaur, S. and Bawa, S. (2016) A Review on Energy Aware VM Placement and Consolidation Techniques. *International Conference on Inventive Computation Technologies* (*ICICT*), Coimbatore, 26-27 August 2016, 1-7. https://doi.org/10.1109/INVENTIVE.2016.7830219

[18] Wang, L., Xu, J., Duran-Limon, H.A. and Zhao, M. (2015) QoS-Driven Cloud Resource Management through Fuzzy Model Predictive Control. *IEEE International Conference on Autonomic Computing* (*ICAC*), Grenoble, 7-10 July 2015, 81-90.

https://doi.org/10.1109/ICAC.2015.41

[19] MathWorks. Linear Regression. https://www.mathworks.com/discovery/linear-regression.html

[20] Coursera, Machine Learning. http://www.coursera.org/learn/machine-learning/home/welcome

[21] Lubis, F.F., Rosmansyah, Y. and Supangkat, S.H. (2014) Gradient Descent and Normal Equations on Cost Function Minimization for Online Predictive Using Linear Regression with Multiple Variables. *International Conference on ICT for Smart Society* (*ICISS*), Bandung, 24-25 September 2014, 202-205. https://doi.org/10.1109/ICTSS.2014.7013173

[22] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A. and Buyya, R. (2011) CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software*: *Practice and Experience*, **41**, 23-50. https://doi.org/10.1002/spe.995

[23] Amazon EC2 Pricing. https://aws.amazon.com/ec2/pricing/on-demand

[24] Ontario Electricity Board, Electricity Rates. https://www.oeb.ca/rates-and-your-bill/electricity-rates

[25] Ibm ilog cplex Optimization Studio v12.5.1. http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r5/index.jsp

[26] Lim, N., Majumdar, S. and Ashwood-Smith, P. (2014) Engineering Resource Management Middleware for Optimizing the Performance of Clouds Processing Mapreduce Jobs with Deadlines. *Proceedings of the* 5*th ACM*/*SPEC International Conference on Performance Engineering*, Dublin, March 2014, 161-172. https://doi.org/10.1145/2568088.2576796