Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 07, No. 02, June 2021.

FAULT TOLERANCE USING SELF-HEALING SLA AND LOAD BALANCED DYNAMIC RESOURCE PROVISIONING IN CLOUD COMPUTING

Mayank Sohani and S. C. Jain

(Received: 13-May-2021, Revised: 22-May-2021, Accepted: 24-May-2021)

ABSTRACT

Over the internet, application efficiency management has recently emerged as an essential service cloud computing. The Cloud Service Provider (CSP) gives various cloud services based on pay per use, which requires efficient monitoring and measuring of services delivered for management of Quality of Service (QoS) through the Internet of Things (IoT) and therefore needs to fulfil the Service Level Agreements (SLAs). However, avoiding SLA violations and ensuring a user's dynamic demands as per QoS fulfilment are challenging in cloud computing while delivering dedicated cloud services. Cloud environment intricacy, heterogeneity and dynamism are expanding quickly, making cloud frameworks unmanageable and unreliable. Cloud systems need self-management of services to overcome these issues. Therefore, there is a need to develop a resource-provisioning scheme that automatically fulfils cloud user's QoS requirements, thus helping the CSP accomplish the SLAs and avoid SLA violations. This paper presents a prediction-based resource management technique called Predictive Cloud Computing Systems (PCCSs). Focus is on the self-healing-based prediction that handles unexpected failures and self-configuration-based prediction of resources for applications. The Predictive Cloud Computing System (PCCS) performance is evaluated in the cloud simulator. The simulation results revealed that Predictive Cloud Computing Systems (PCCSs) achieve better results than existing techniques, in terms of execution time, cost-effectiveness, resource conflict and SLA breach while delivering reliable services.

KEYWORDS

Quality of service, Cloud-service provider, Service-level agreement, Service-level objective, Predictive cloud computing system.

1. INTRODUCTION

Cloud computing focuses on QoS parameters, such as throughput, response time, availability, capability, service cost and reliability, among others. The QoS parameters play a vital role in the ranking of service providers. QoS parameters are continuously monitored and controlled by service providers to avoid SLA breaches. According to the study and analysis, it is reported that the Virtual Machine (VM) requires different durations of boot time before it is ready to operate [1] [2] [3]. The VM needs 5 to 15 minutes to get started; therefore, during this time, system resources would not be available and the requests could not be served due to lack of resources. This leads to an infringement of SLA and due to this, penalties on cloud providers are imposed. Our objective is to design a solution for provisioning and predicting the need for a VM in advance. Making it available only on time could maintain the level of availability and prevent violations of the SLA [4]. This research will discuss cloud computing advantages, such as up-front costs, lower infrastructure maintenance and ease of resource scaling for the users. Cloud computing has various benefits and many issues of energy consumption, resource utilization, VM migration and service-level agreement (SLA) violations [5]-[6]. In this paper, we are using a threshold-based Virtual Machine Consolidation (VMC) strategy. Many issues of the resources need to be addressed. Therefore, VM consolidation (VMC) is the best way to solve them.

In cloud computing, unpredictable situations are handled by an intelligent autonomic system that keeps the system stable based on human guidance and easily adapted to new environmental conditions, such as hardware, software failures, …etc. This system can quickly handle the heterogeneity, availability, reliability and dynamism problems. The system works through monitoring, analyzing, planning and execution phases in a controlled way in order to achieve the application execution goal within the deadline by fulfilling the user's defined QoS parameters with minimum complexity. Virtual machine

M. Sohani and S. C. Jain are with the Department of Computer Science and Engineering, Rajasthan Technical University, Kota-324010, India. Emails: sohanimayank@gmail.com and scjain1@gmail.com

optimum utilization is highly desirable to maintain the required SLA and it is achieved by virtual machine dynamic consolidation. The live VM migration is used for VM reallocation as per current resource workload demands of users and reduces energy consumption [7] [8] [9] [10] [11]. However, virtual machine migration tends to increase application execution latency and infrastructure energy overheads. Many factors are considered in migration cost, such as the number of virtual machines considered for migration, network bandwidth viewed for migration, memory content update rate of the virtual machine, source and destination servers' workload at the time of migration [12]-[13]. During migration cost investigation quantitatively, the power consumption and time of migration linearly increase as network bandwidth and size of VM increase, respectively, whereas migration time decreases if the increase in bandwidth and increases if the VM memory size increases [14]-[15].

Various research literature focuses only on the current resource requirements of the destination host. The future utilization has not been discussed more at the time of the VM allocation stage. That will generate needless VM migrations that would result in more energy consumption and increase SLA violations in the data center [16]. This paper proposes a new prediction-based method for different resource utilizations; i.e., CPU, memory and network. The work focused on the memory utilization of these resources on the hosts at the time of VM placement. Our proposed method is a prediction model based on feedforward neural networks with backpropagation for linear regression-based prediction models. Our detection technique is responsible for current and future resource utilization on the hosts before placing VMs.

As per QoS requirements, a predictive cloud computing system provides self-management of resources that fulfils the following properties of self-management:

- This paper presents a detailed analysis of selected resource provisioning techniques that work for QoS requirements, VM migration strategies, load balancing techniques and SLA violation monitoring schemes.
- It proposes and implements an algorithm for predicting the workload in cloud computing systems.
- The proposed algorithm improves self-healing in a predictive cloud computing system as a capability of the system to identify, analyze and recover from unfortunate faults automatically.
- It proposes and implements self-configuration in a predictive cloud-computing system, which is an indicator of the capability of the system to adapt to the changes in the cloud environment.
- It proposes and implements a new VM migration and load balancing scheme for the cloud-computing system.

In our earlier work, QoS-based Predictive Priority-based Dynamic Resource Provisioning Scheme [17] is proposed. The Predictive Priority-based Dynamic Resource Provisioning Scheme is a novel approach for predicting priority-based scheduling schemes. This explores a new approach that is an efficient emergency priority-aware algorithm. In this scheme, we consider the emergency cloud request and priority is given to load that emergency cloud requests for execution. This will ensure the load request availability and longevity of more sophisticated requests in heterogeneous cloud computing environments without SLA violation monitoring [18]. To realize this, QoS-aware autonomic resource management of cloud services needs to be considered as a crucial aspect that reflects the cloud management complexities. To design a resource management approach which can work as a QoS-based autonomic approach, Predictive Priority-based Dynamic Resource Provisioning Scheme has been further extended by proposing Predictive Cloud Computing System (PCCS). In this research work, a resource management approach which can work as a QoS-based autonomic approach has been proposed which offers fault tolerance using self-healing SLA and load balanced dynamic resource provisioning in cloud computing, to handle sudden failures and provide cloud resource maximum utilization by self-optimization.

The motivation of this paper is to design an intelligent cloud-based and QoS-aware autonomic resource management approach called Fault Tolerance Using Self-healing and Load Balanced Dynamic Resource Provisioning in Cloud Computing. This offers handling of sudden failures of resources through self-healing, resource self-configuration for applications and maximum resource utilization through self-optimization features. The proposed scheme works to minimize SLA violation rate, execution cost, execution time and resource contention and maximize energy efficiency and resource utilization. The PCCS performance is tested with a CloudSim simulation environment using PlanetLab workload traces.

PCCS increases service availability and reliability and improves satisfaction of cloud users. The rest of the paper is organized as follows: Section 2 describes the related work, while the proposed model is presented in Section 3. Section 4 presents the simulation setup, results and discussion. Section 5 presents conclusions and future scope.

2. RELATED WORK

Many studies have investigated the SLA management systems in cloud computing, but SLA enforcement is covered only by a few of them. Without considering enough cloud requirements, other environments, such as grid computing and service-oriented architecture, applied the SLA models into cloud computing as per most related works. In the self-healing system, the central part consists of system monitoring and reacting procedures [19]. The Federated Cloud Trust Management Framework (FCTMF) model resolves trust issues. It evaluates trust on the basis of SLA parameters and by customer and CSP feedback [20]. SH-SLA models enforce the SLA monitoring and reacting procedures based on SLA violations in cloud computing. Each SLA is connected with its related SLAs in different layers of the SH-SLA model of cloud computing, so that all corresponding SLAs can notify their status to each SLA. So, cloud service providers can prevent SLA violations before sensing by the end-users without addressing cost and energy consumption QoS parameters [21]. RADAR technique performs autonomic-management properties for self-healing and self-configuration handled during unexpected failures of service and resource configurations, respectively, with minimum human intervention and gives better results for QoS parameters along with managing hardware, software or network faults, but the study unable to address the self-protecting property [22]. Existing approaches consider a host overloaded detection based on threshold-based host CPU utilization and consider available bandwidth equal to base bandwidth, thus leading to performance degradation. The overloaded host VM migration or reallocation towards another under loaded host machine is not addressed in this study [23]. Previous proposed work assumptions are not based on energy consumption and violations of SLA considering network traffic. Energy consumption can be minimized by existing methods considering the size and current utilization of VM, but network traffic can also affect SLA violations [24].

In the cloud environment, this will provide capable monitoring that would be able to share resources in Clouds. In [25], the authors offer the solution cloud federalism, where the different cloud vendors the cloud services in an integrated manner. The Cloud Burst is the best example of cloud federalism. In [26], the authors' discussion is about resource management's performance with the help of live migration. This feature is added in the cloud system that has to provide excellent services into the cloud environment of active fault tolerance by flawless Virtual Machine movement. The consumer is not being aware of any change in a virtualized environment from wavering hardware to unwavering hardware. In these models, virtual technologies have provided resource consolidation with minimum energy consumption and are unable to address the issue of self-management [27]-[28]. Resource over-provisioning can be solved by VM placement as per the VM resource requirements independently based on their requests. Placing more VMs on the same PM by sharing hardware resources exceeds its physical capacity [29]. Unfortunately, over-commitment affects the application performance with QoS violations and SLA penalties by congesting limited PM resources [30]. In UP-VMC, resource requirements for current and future utilization consolidate the VMs with the minimum quantity of active PMs. It uses regression-based prediction for future and current resource utilization, enhancing the QoS and minimizing the number of VM migrations, but application scalability and network resource utilization factors are not addressed in this study [31].

In cloud computing, the overall response time of the system is reduced by load balancing and this policy of workload distribution fulfills the QoS requirements along with efficient cloud resource utilization. Several techniques were proposed; however, VM migration and fault tolerance issues are not still fully addressed [32]. An ideal framework PRMF can identify current workload and future workload prediction for provisioning/deprovisioning cloud resources as per the demand of application users. This framework identifies given workload patterns with key evaluation metrics using statistical techniques. It applies best-fit algorithms from algorithms using predictive methods to provision/de-provision VM instances, but is unable to address issues, such as cost, makespan time and energy consumption [33]. Resource provisioning techniques are working based on predetermined considerations, are reactive and are provided with leading CSPs. Under-or over-provisioning of resources is done in reactive approaches that have time-lag in resource demand and provisioning. The study proposed a predictive technique for

cloud resource management to overcome these limitations [34]. In cloud computing, search optimization methods are introduced by many studies, but there is still some scope to get enhanced search for optimal solutions. To achieve this solution, many functions need to be involved; i.e., execution time, power consumption, performance, QoS and SLA violation rate [35]. However, in some earlier works, the maximum three objective functions are taken into consideration to get the optimal solution in cloud computing. The ESCORT framework addresses these issues to optimize execution cost, energy consumption and SLA violation rate [36].

A secure resource provisioning model with SLA integration is proposed to achieve many benefits for cloud users' and cloud service providers' points of view. This secure provisioning model is used by cloud service providers for the security parameters' fulfilment purpose without considering other major QoS parameters, such as execution time, cost, throughput, energy consumption [37], …etc. In cloud computing applications, workload changes as per time and to fulfil such workload resource requirements, cloud service providers dynamically allocate the resources. Dynamic resource provisioning aims to improve resource utilization and reduce resource usage costs for cloud users [38]. To achieve profit-aware resource provisioning, the cloud service provider must provide less renting cost with proper resource utilization to meet the QoS requirements. The dynamic resource provisioning technique works as an effective technique for utilization of resources without considering energy consumption and SLA violations. The goal is to minimize the resource rental cost and maximize resource utilization for profit earning [39]. The CHOPPER framework works based on self-protection, self-healing, self-optimization and self-configuration using three phases of self-management; i.e., Monitor, Analyze and Plan & Execute to address different QoS parameters, but it is unable to calculate the workload resource demand in advance [40]. The increase of cloud users with peak time demands makes the risk of resource faults during interactions with the cloud infrastructures match the execution deadline. That can lead to resource contentions and damage the reputation of cloud service providers due to non-consideration of cost and energy consumption in the study [41]. The authors propose an MASA framework that works based on a healing agent and a consistency manager agent to handle the runtime issues of resource provisioning and SLA violations, but it is unable to manage adaptive fault tolerance scheme for cloud security solution [42].

3. PROPOSED MODEL

SLA is the most important part between the cloud service provider and the customer. SLA is a mutual agreement between the cloud service provider and the customer. This Service Level Agreement (SLA) is the official negotiation document at the service level and shall contain performance parameters along with the minimum level of service quality. Our proposed SLA is including an automated cloud healing process based on the above description and prediction. In the proposed method, each service has its function of automatic healing and reaction. This SLA-based prediction will work on the threshold value and related SLAs on the cloud user service. This threshold value helps prevent breaches of the SLA and the specific QoS threshold. If the QoS value is higher than the threshold value, the state of violation prevention shall be shown as active and autonomous healing gets activated.

The proposed prediction-based model optimizes cloud computing energy-efficient resources automatically and considers essential aspects, such as configuration, prediction-based recovery, optimization and protection and automatic QoS-aware resource management. Our most essential contributions offer prediction-based intuitive design of cloud applications and resources by installing missed or old H_Components. Prediction-based automatic healing is provided by handling sudden failures, automatic protection against security attacks and automatic optimization as the resources are being used optimally.

3.1 System Architecture

The system behavior and its entire structure are represented ultimately with the help of system architecture only. That can define the system's architectural overview of the whole system. The main aim of the proposed Predictive Cloud Computing System is to predict the future workload and ensure resource provisioning in advance with the best suitable pair of resources to fulfil QoS requirements and avoid any SLA violations occurring due to resource provisioning. The proposed model ensures resource provisioning with less power consumption under low execution cost with the best reliable resource pairs

for allocation. Figure 1 represents the predictive cloud computing system's architecture concept map.

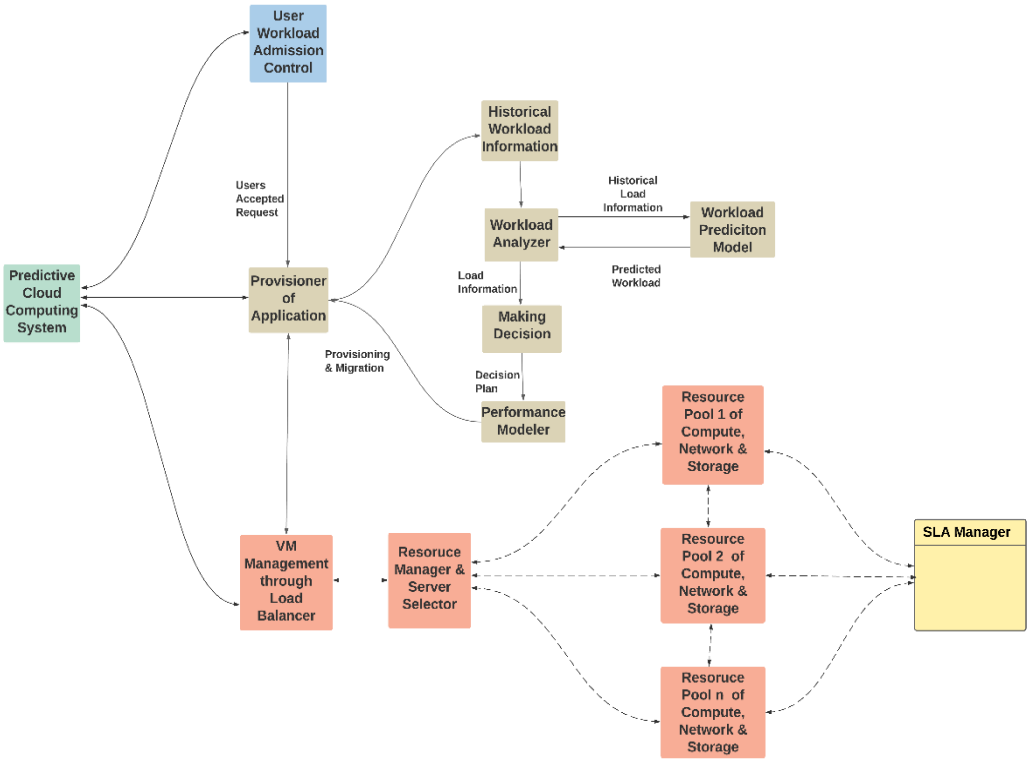

Figure 1. Predictive cloud computing system's architecture concept map.

The workload traces of PlanetLab are taken as an input dataset. The load analyzer performs analysis on workload data based on time series and converts it from unstructured data into structured data as per real-time workload traces received. The workload predictor predicts the future workload and based on this prediction, cloud providers perform arrangement and provisioning of the resources as per QoS requirements. The load analyzer takes care of the current resource utilization of all resource cloud nodes available in the system.

The future workload is predicted based on current workload traces of Planet Lab and prepared structured data based on time series. This predicted workload is used to maintain the SLA commitment towards the cloud user service quality and availability. After predicting the future resource requirements, we can ensure the availability of optimal VM resources and provision them under effective load balancing techniques. The workflow diagram for a Predictive Cloud Computing System is represented in Figure 2. In PCCS, the user submits the request for services based on service types and their properties and negotiation occurs between the user and the CSP. SLA is signed between users and CSP as per QoS requirements and SLA terms. Now, CSP arranges the specific type of resources and sub-resources as per user QoS requirements and provisions these resources for the services used by cloud users. Suppose that required resources are not available in the resource pool. In that case, either renegotiation occurs based on available resources in the resource pool or CSP finds new resources. If resources are available, then the resource configuration is performed using a workflow template. The monitoring unit monitors the entire execution process for user-submitted workload and the workload analyzer prepares a historical workload database. The proposed PCCS applies a predictive cloud computing model on a historical workload database and prepares predicted resources in advance to provide them shortly without violation of the SLA. This PCCS prediction scheme ensures that required resources are ready to be used in advance to save the extra time of provisioning users' workload requests. Application workload is executed using PCCS-provisioned resources. If any demand of current workload is remaining for execution and the same notified by the monitoring unit, then the same is repeated to execute the application workload. This entire process follows four phases; i.e., monitoring, analysis, planning and execution concerning time *t* and updating estimates and actual resource consumption and workload status. Resource configuration upgradation or reconfiguration is performed based on monitoring and analyzing phase inputs for QoS, fault tolerance and SLA fulfilment. VM migration and load balancing

are automatically performed based on VM threshold values as per input given by monitoring, analysis, planning and execution phases to the VM load balancer. The entire process gets stoped after the execution of the user-submitted application workload.

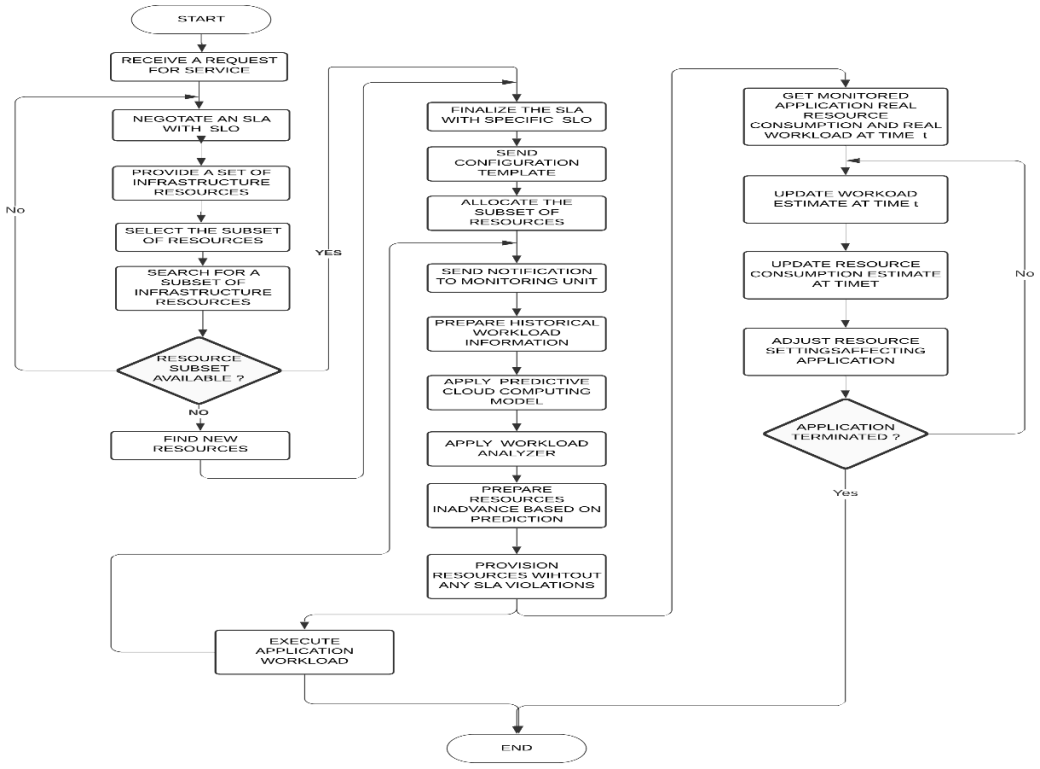

Figure 2. Predictive cloud computing system's workflow.

3.2 Workload Analyzer and Predictor

The workload analyzer plays an essential role in the framework to analyze the task load that can only be done after the establishment of cloud infrastructure. The analyzer arranges the unstructured data into structured data based on time series. As unstructured data is no more helpful to get the desired future workload forecasting, the information that can help future workload forecasting is sequential structured data.

Workload predictor is used to take this sequential structured data provided by the workload analyzer as an input to forecast the future workload. Here, backpropagation-based prediction methodology is used, where several input nodes are used to calculate the output using the process of training the workload dataset by the supervised learning methodology. When the task load data formatting is done, it can be used as structured data for quick forecasting purposes, which gives an accurate-manner prediction. This process provides the advantages of automatic learning and reduces the time to fit the analyzer data for every prediction purpose.

3.2.1 Host Server Selector and Manager

All the host machine details are listed and managed by the server. As per the workload request, the best possible resources are ranked based on their configuration and performance metrics. These cloud resources are then provisioned based on their performance ranks. The best VM is provisioned for new user workload requests from the load balancer.

3.2.2 Forecasting of Predicted Workload

Workload prediction is made on the historical workload data history using the backpropagation algorithm. This algorithm works internally and is used to predict the workload of the near future as a tool.

3.2.3 Host State Management

As per the predefined SLA requirements, the host state management performs the cloud resource management as per the resultant prediction without compromising the SLA violations. The host resources are continuously monitored by the host-state manager for their health parameters; i.e., time to start, time to stop, total uptime and total downtime for every physical server.

3.2.4 Resource Requirement Forecasting Based on Prediction

In this scheme for forecasting, we are using the backpropagation algorithm, where for the forecast, the future demand of resources is based on the past downtime history. The resource forecasting is carried out using the following equation:

$$\mathrm{RE(t)} = \alpha * \mathrm{RE(t-1)} + (1-\alpha) * \mathrm{RO(t)} \qquad 0 \leq \alpha \leq 1, \tag{1}$$

Here, *RE(t)* expresses the estimated resource and *RO(t)* represents the observed resource load during the time *t.* α is portrayed as a constant that emulates the trade-off between constancy and communion. The proposed model used it to forecast CPU utilization load, predicting the future load calculated in every minute and forecasting immediately.

If the observed load R*O(t)* prediction is in sequential order; i.e., 30, 40, 50 and 60, then prediction for the next term would be more accurate as 70 and then, the algorithm works correctly. Intermediate load values are not forecast by the algorithm. To decrease and increase both order negative value depiction purposes, the formula below for -10 is used in place of the above formula.

$$\mathrm{RE(t)} = -|\alpha| * \mathrm{RE(t-1)} + (1+|\alpha|) * \mathrm{RO(t)} \tag{2}$$

The Depiction for decreasing and increasing order creates confusion to select the exact value between these two. To predict more accurate future workload, the formula is modified as shown below:

$$\mathrm{RE(t)} = \mathrm{m} * \mathrm{RA(t-1)} \tag{3}$$

where m is the multiplier and its value is calculated as:

$$\mathrm{m} = \frac{\mathrm{RA(t-1)}}{\mathrm{RA(t-2)}} \tag{4}$$

Here, the resource-estimated load is calculated using actual resource load concerning time and a multiplier value; i.e., m. This forecasting of workload based on prediction is very near to the relative value of required resources by the cloud users in the near future. However, this prediction model needs resources to predict the forecasting of required resources. To overcome the resource provisioning wastage issues, this model gives perfect VM allocation requirements using backpropagation learning.

The above-defined model provides self-healing for sudden failures, self-protection against security attacks and self-optimization as the resources are being used optimally.

(i) Our proposed model will have no human intervention requirements and will enhance the users' satisfaction level. In this prediction-based model, SLA would efficiently control cloud users' QoS needs and improve the load balancing of the provisioned cloud resource utilization (CPU and memory). Our proposed model optimizes execution cost, time and energy efficiency.

(ii) This model proposed the phases based on prediction-based properties and prediction is based on the regression model. In the execution time of loads, prediction-based performance (QoS value) continuously analyzes the plans and action to handle that message and executes the procedure to maintain efficiency.

(iii) We have classified the load into different categories based on deadline emergency. That will help investigate the impact of various workloads on different QoS parameters. The execution of workloads would enhance the availability of cloud-based services and secure energy-efficiency reliability.

3.3 Metrics Based on QoS

The QoS parameters; i.e., waiting time, execution time, energy consumption and execution cost are calculated for user-submitted workload as per the cloud environment consideration.

$$WETi = \sum_{i=1}^{n} \left(\frac{WFTi - WESTi}{n} \right) \tag{5}$$

where $WETi$= workload execution time, $WFTi$= workload finish time, $WESTi$= workload execution start time and n= number of workload.

$$WWTi = \sum_{i=1}^{n} \left(\frac{WESTi - WSTi}{n} \right) \tag{6}$$

where $WWTi$= workload waiting time, $WESTi$= workload execution start time, $WSTi$= workload submission time and n= number of workload.

$$WCTi = WETi + WWTi \tag{7}$$

where $WCTi$= workload completion time.

$$EC = ECdc + ECm + ECse + ECe \tag{8}$$

where EC= energy consumption, $ECdc$= energy consumption of data center, ECm= storage-device energy consumption, $ECse$= switching-equipment energy consumption and ECe= extra energy consumption.

$$AC = RC + PC \tag{9}$$

$$RC = WETi \times Price \tag{10}$$

$$PC = \sum_{i=1}^{c} (PCi) \tag{11}$$

where AC= average cost, RC= resource cost, PC=penalty cost and $c \in PC$ = penalty cost set.

$$RUi = \sum_{i=1}^{n} \left(\frac{Resource\ Actual\ Time\ Spent\ to\ Execute\ Workload}{Resource\ Total\ Uptime} \right) \tag{12}$$

where RU= resource utilization.

$$FDR = \frac{Number\ of\ Faults\ Detected}{Total\ Number\ of\ Faults} \tag{13}$$

where FDR= fault detection rate.

3.4 Fault Tolerance Using Self-healing SLA and Load Balancing

In this paper, we are working on a prediction-based self-healing that is a part of proactive fault tolerance (FT) in high-performance computing that prevents computing node failures from affecting running parallel applications, so that nodes would be in the failure process [43]. In this research, our main objective is to work on a prediction-based fault tolerance scheme in cloud computing. Our cloud model approach will work based on the above-described cloud computing and prediction base technique and improve the quality of services.

SLA is an essential document of mutual agreement between the cloud service provider and the customer. In this SLA (Service Level Agreement), the official negotiation document at the service level, the QoS and its service costs shall be agreed upon and shall contain performance parameters and a minimum level of service quality. Our SLA is a proposed SLA, including an automated cloud healing process based on the above description and prediction. In the proposed method, each service has its function of automatic healing and reaction. This SLA-based prediction will work on the threshold value and related SLAs on the cloud service for users. This threshold value helps prevent breaches of the SLA and the specific QoS threshold. The QoS threshold value is compared to the SLO value recorded in the SLA contents. If the QoS value is higher than the threshold value, the state of violation prevention shall be shown as active and autonomous healing. In this paper, prediction-based self-optimization of cloud computing energy-efficient resources, proposing QoS-aware autonomic resource management, considers other essential aspects, such as self-configuration, prediction-based self-healing, self-optimization and self-protection as proposed in algorithms 1, 2 and 3.

The significant contributions of this paper offer prediction-based self-configuration. Algorithm 1 describes the makespan as per MIPS of VM using machHigh and machLow. The task finish time is calculated and is considered as makespan time. This calculated makespan time is then added to predicted task execution time of a VM. If [P_comTimejj< makespan & P_comTimej< makespan] both are true

for makespan time, then swap respective parameters; i.e., makespan, task and machine, get executed. After complete execution, makespan time is returned as a value. Our proposed model works for cloud applications and resources on a prediction base that would be installed if the component or cloud application is missing.

Algorithm 1: Prediction of Workload

```
1.  For all Mi ∈ machHigh do
2.    For all Mj ∈ machLow do
3.      For Tk ∈ Mi do
4.        P_comTime^ij ← computeFinishTime (Mj)
5.        P_comTime^ij ← P_comTime^ij + PredictiveExecutionTime of Tk on Mj
6.        P_comTime^j ← makespan_PredictiveExecutionTime of Tk on Mi
7.          If P_comTime^ij < makespan then
8.            If P_comTime^j < makespan then
9.              Makespan ← P_comTime^ij
10.                   task ← Tk
11.             Machine ← Mj
12.           End if
13.         End if
14.       End for
15.     End for
16.   End for
17. return makespan
```

In this prediction-based SLA, we aim to predict based healing; therefore, our scheme is a prediction-based algorithm. SLA is a document that is a mutual agreement between client and provider. Our proposed SLA relies on prediction based on load balancing of resources between client and cloud provider. In this framework for load balancing, we create a priority queue of loads and, based on the prediction priority queue, add the load into the priority queue according to a criterion. This criterion is based on threshold values. This load priority queue manages the resources according to predicted execution time and energy consumption and maintains the priority queue; therefore, deadlock is solved according to prediction-based priority queue. In this scheme, we propose prediction-based self-healing as we have all prediction-based priority queues for loads and the resources are allocated according to priority. As that proposed scheme provides the resources based on prediction-based requirements, our proposed system provides the prediction-based resources such as hardware, CPU and memory that our prediction-based SLA can quickly add.

Prediction-based self-optimization, self-healing and auto-configuration are monitored by the monitoring unit and resource performance management is executed using self-management properties as per Algorithm 2. All processing nodes' performance is monitored through QoS agent. The load priority queue considered for workload set (Wpq= {Wp1, Wp2…., Wpm}) is submitted to the load priority queue. The workloads are executed as per QoS and resource availability needs. After provisioning, QoS parameters (execution time, cost and energy consumption) were calculated for every workload using QoS metric equations 5 to 13. Alert is generated if any condition fails [([PET ≤ Dt && PC ≤ BE] = = 'TRUE') or ([PEC≤PTH] = = 'TRUE')]. Further, in self-healing, the system checks the status of all the components and if any faults are found, it will replace the required components. This entire process maintains log information for current device status and updates resource utilization information. If usage of resources is more than the threshold value [(CurrentStatus ['CPU' || 'MEMORY'] > Value of THRESHOLD)], then alert is generated. All the software versions' status is checked for hardware components in the system. If [(Component version status = OLD || Not-VALID)] is true for OLD or Not-VALID, then generate alters and install the new replacing the old version. The [H_Component_Name and H_Compoenent_Id] is updated based on log information.

Our resources represent nodes that have a state as activate and deactivate. Within this prediction-based SLA, healing uses a hybrid tool in a diagnostic approach. This hybrid tool is used for diagonal purposes and combines analytical methods that cooperate with a common goal. We apply VMC (VM consolidation) based on current and future VM migration in this proposed model. We are using a prediction regression-based model. Algorithm 3 performs the load balancing and VM migration operation. Workload is assigned to a VM as per VM allocation policy and CPU utilization is calculated

Algorithm 2

1. **# Phase One**: ***Prediction based Self-Optimization***
2. *Begin*
3. *Load Priority Queue: $Wp_q = \{Wp_1, Wp_2, \ldots\ldots\ldots\ldots, Wp_m\}$*
4. *Add Loads into Priority Queue: $Wp_a = \{Wp_1, Wp_2, \ldots\ldots\ldots\ldots, Wp_o\}$ where $p_o \leq p_m$*
5. *Allocate resources to task loads based on Quality of Services parameters*
6. *Loop until all Predict base queue loads (Wp_a), where Predict average cost (P_C) , Predict energy consumption (P_{EC}) and Predict execution time (P_{ET}) for execution*
7. *If ([$P_{ET} \leq D_t$ && $P_C \leq B_E$] = = 'TRUE') then*
8. *If ([$P_{EC} \leq P_{TH}$] = = 'TRUE') then*
9. *Schedule execution according to prediction-based priority queue resources*
10. *Else*
11. *Alert Message*
12. *End if*
13. *Else*
14. *Alert Message*
15. *End if*
16. *End loop*
17. ***# Phase Two: Perdition based Self-Healing***
18. *Begin*
19. *Set of Prediction based Priority Queue Nodes: $PNode_{set} = \{PNode_1, PNode_2, \ldots\ldots\ldots, PNode_n\}$, where $PNode_c$ represents current node of queue.*
20. ***If** (Predictive Priority Queue == Empty) **then***
21. *Scan drives and check replica of original driver*
22. *Add node into node set from the current node number*
23. ***Else***
24. *Generate alert for Priority queue node is already exist*
25. ***End if***
26. *Repeat loop until all hardware priority queue node (Status of Node)*
27. *Get detail of current status [EVENTTYPE, TIMESTAMP, EVENTID]*
28. ***If** (EVENTTYPE = = 'EMERGENCY' OR 'ERROR') **then***
29. *Database is updated by using log information [NodeName and address of MAC]*
30. ***End if***
31. ***End loop***
32. *Loop until repeat Software Monitoring [Resource utilization (MEMORY and CPU)]*
33. ***If** (CurrentStatus ['CPU' || 'MEMORY'] > Value of THRESHOLD) **then***
34. *Generate alert message*
35. *Update Resource utilization (Memory and CPU) information*
36. ***End if***
37. ***End loop***
38. ***# Prediction Based Auto Configuration for Self-Healing Process***
39. *Begin*
40. *Prediction base Priority Queue of H_Components: = $\{Hc_1, Hc_2, \ldots\ldots, Hc_p\}$*
41. *Priority Queue of Active H_Components:= $\{ Hc_1, Hc_2, \ldots\ldots, Hc_q \}$, where $q \leq p$*
42. ***While** true **do***
43. *Repeat loop for all software S_Components*
44. *Repeat loop to get all Priority Queue of Active H_Components version status*
45. ***If** (Component version status = OLD || Not-VALID) **then***
46. *INSTALL the new version for replacing the old version using the process of uninstall*
47. ***End if***
48. ***End if***
49. ***End loop***
50. *Repeat loop all hardware H_Components then Track Log Register*
51. *Repeat loop to get all detail of Priority Queue of Active H_Components status [EVENTTYPE, TIMESTAMP, EVENTID]*
52. ***If** (EVENTTYPE == 'EMERGENCY' || 'ERROR') **then***
53. *Database is updated by using log information [H_Component_Name and H_Compoenent_Id]*
54. ***End if***
55. ***End loop***
56. ***End loop***

and monitored. If host CPU utilization < 0.21, then CPU is added to the underutilized host list. If host CPU utilization > 0.79, then the CPU is added to the over-utilized host list; otherwise, the host resides in a safe host list. Now, check which VM is maximally utilized and then upgrade VM configuration if possible; otherwise, migrate VM towards a safe host. Consolidate underutilized host VMs and either shut down the VM or add them to the migration list. Prepare safe host list, increasing order of CPU utilization and performing VM migration.

Algorithm 3: VM Migration and Load Balancing

1. *Based on the VM allocation policy schedule the load on the VM.*
2. *Repeat loop for every host to calculate the CPU utilization*
3. *Repeat until*
   - *3.1 Get the first host in the list*
   - *3.2 if host CPU utilization < 0.21, then host add into underutilized host list*
   - *3.3 if host CPU utilization > 0.79, then host add into over utilized host list*
   - *3.4 otherwise host add into safe host list close the condition of the loop*
4. *Repeat until over utilized host list get the VM with maximum utilization*
   - *4.1 get the available MIPS from the host of maximum utilized VM*
   - *4.2 if MIPS is available, then add available MIPS to over utilized VM*
   - *4.3 otherwise, migrate the VM to a safe host based on a safer policy*
5. *Repeat until for each underutilized host*
   - 5.1 *Consolidate every VM on the underutilized host and move those*
   - 5.2 *VMs to migration list close the condition of the loop*
6. *Organize the safe host in increasing order based on CPU utilization and migrate all the VMs based policy*

4. SIMULATION SETUP, RESULTS AND DISCUSSION

Our proposed Predictive Cloud Computing System is modeled and simulated using CloudSim. This research is carried out using the CloudSim toolkit [44]. The system modeling and behavior of cloud system components; i.e., VMs, Datacentre and RP rules are fully supported by the CloudSim toolkit [7]. The standard resource scheduling methods' implementation can be done with little effort and method extension is possible. The inter-networked and distinct clouds are contained in the cloud environment simulation using the toolkit.

Furthermore, the toolkit supports VM provisioning under an inter-networked cloud environment to implement resource scheduling techniques through custom interfaces. Toolkit benefits provide the performance with time effectiveness, flexibility and applicability for test results. The heterogeneous workload of clouds is considered for experimental results. Each available resource contains one or more processing elements with different Million Instructions Per Second (MIPS). In this outcome, we assume that every workload admitted to the Predictive Cloud Computing System (PCCS) contains a workload of fluctuating sizes of inputs and execution times. These workloads are considered in the form of Cloudlets [44].

The resource configurations for testbed are: 2.4 GHz, Intel Core 2 Duo, 160 GB HDD, 1 GB RAM, with Windows operating system, 2.9 GHz, Intel Core i5-2310, 160 GB HDD, 1 GB RAM, with Linux operating system, 2.0 GHz, Intel Core i7-8550, 256 GB HDD, 4 GB RAM, with Linux operating system. This paper simulates our results with four SLA self-healing MASA, SH-SLA, CHOPPER and RADAR with our new proposed works. According to our simulation results, the proposed prediction-based PCCS SLA is the best. Table 4 gives details of workload types along with missing deadline compensation provided. We have shown the different testbed results in the Table 1, Table 2 and Table 3 given below.

In this simulation, two different cloud infrastructures through different processor configurations (4-core processor and 8-core processor) have been considered to measure the variation of different QoS parameters; i.e., energy efficiency, execution cost, resource utilization, throughput, SLA violation rate, resource contention, waiting time, fault detection rate, reliability, availability, intrusion detection rate and turnaround time. The different QoS parameters Improvement Rate (IR) percentage and simulation statistics summary are described in the tables. The CloudSim simulation environment has been

considered with 3000 same-type workload traces of PlanetLab for performance testing. The PCCS is validated for different QoS parameters through autonomic resource management existing techniques, such as CHOPPER [21], SH-SLA [8], RADAR [9] and MASA [22]. The PCCS performance is more stable and efficient in resource management for changing cloud workloads using the coefficient of variation with a small value. The simulation results are presented in Table 1, Table 2 and Table 3 for all the different cloud infrastructures.

Table 1. Simulation results and improvement rate (IR) of PCCS and CHOPPER.

QoS Parameters	4-Core Processor			8-Core Processor		
	PCCS	CHOPPER	IR (%)	PCCS	CHOPPER	IR (%)
Energy consumption (kWh)	88.91	117.61	24.4	124.46	162.13	23.23
Execution cost (C$)	94.2	128.71	26.81	162.7	219.56	25.90
Resource utilization (%)	79.11	71.01	10.24	83.66	77.761	7.05
Energy efficiency (%)	89.81	82.85	7.75	81.45	73.89	9.28
Throughput (workload/sec)	549.8	559.19	1.68	669.43	619.55	7.45
SLA violation rate (%)	28.15	36.56	23.00	41.46	47.91	13.46
No. of missed deadlines	28.11	34	17.32	44	49	10.20
Resource contention (sec)	3416.56	4180.48	18.27	4830.78	5461.45	11.55
Waiting time (sec)	299.32	306.69	2.40	268.69	266.15	0.95
Fault detection rate (%)	67.48	64.78	4.00	74.98	71.12	5.15
Reliability (%)	7.91	6.23	21.24	8.42	8.18	2.85
Availability (%)	86.39	82.71	4.26	89.77	89.22	0.61
Intrusion detection rate (%)	27.98	26.48	5.36	48.78	44.69	8.38
Turnaround time (sec)	622.15	651.45	4.50	561.89	593.28	5.29

Table 2. Simulation results and improvement rate (IR) of PCCS and RADAR.

QoS Parameters	4-Core Processor		
	PCCS	RADAR	IR (%)
Energy consumption (kWh)	88.91	110.61	19.62
Execution cost (C$)	94.2	118.71	20.65
Resource utilization (%)	79.11	69.01	12.77
SLA violation rate (%)	28.15	35.56	20.84
Fault detection rate (%)	67.48	64.78	4.00
Turnaround time (sec)	622.15	651.45	4.50

Table 3. Simulation results and improvement rate (IR) of PCCS, MASA and SH-SLA.

QoS Parameters	4-Core Processor				
	PCCS	MASA	IR (%)	SH-SLA	IR (%)
Energy consumption (kWh)	88.91	121.61	26.89	122	27.12
Execution cost (C$)	94.2	125.71	25.07	129	26.98
Resource utilization (%)	79.11	66.01	16.56	63	20.36
SLA violation rate (%)	28.15	37.56	25.05	38	25.92
Fault detection rate (%)	67.48	61.78	8.45	60.78	9.93
Turnaround time (sec)	622.15	658.45	5.51	666.3	6.63

Table 4. Workload urgency details with their types.

Load type	Emergency deadline (P_*Du*)	Slack time (seconds)	Delay time (seconds)	Deviation status	Minimum penalty	Penalty rate
Emergency Deadline	P_Du < 0.25	10	0–50	5 %	200 s	5 %
			51–100	10 %	400 s	6 %
			101–150	15 %	600 s	7 %
Medium Deadline	0.25≤P_Du ≤0.75	30	0–50	5 %	100 s	4 %
			51–100	10 %	200 s	5 %
			101–150	15 %	300 s	6 %
Deadline	P_Du > 0.75	60	0–50	5 %	50 s	2 %
			51–100	10 %	100 s	3 %
			101–150	15 %	150 s	4 %

The results demonstrate that PCCS improves average resource utilization by 13.40%, average energy efficiency by 8.52%, average fault detection rate by 6.31%, average intrusion detection rate by 6.87%, average throughput by 6.39%, average reliability by 12.04%, average availability by 2.44% and minimizes average SLA violation rate by 20.44%, average energy consumption by 24.25%, average execution cost by 23.82%, average number of missed deadlines by 11.34%, average resource contention by 10.36%, average waiting time by 2.59% and average turnaround time by 5.59% as likened to existing resource management techniques. As per the simulation results, it is clearly shown that PCCS outperforms existing techniques in terms of QoS parameters, as PCCS achieves every situation automatically.

Figures 3, 4, 5 and 6 represent the proposed PCCS scheme results of different QoS parameter comparison with the existing techniques of SH-SLA, MASA, RADAR and CHOPPER. The proposed Predictive Cloud Computing System (PCCS) performs better in terms of energy consumption, execution cost, resource utilization, fault detection rate, turnaround time and SLA violation rate for SLA-aware autonomic resource management and gives better results for SLA violation rate along with different QoS parameters.

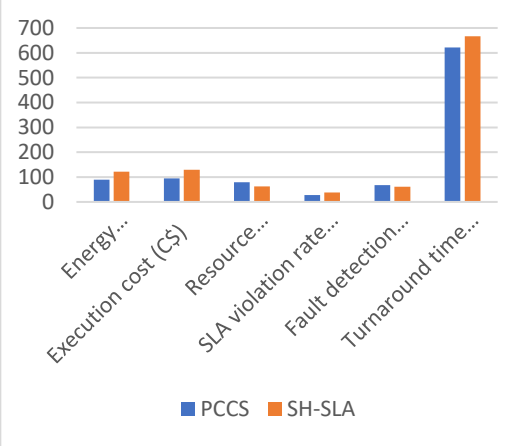

Figure 3. SH-SLA and proposed PCCS comparison on different QoS parameters.

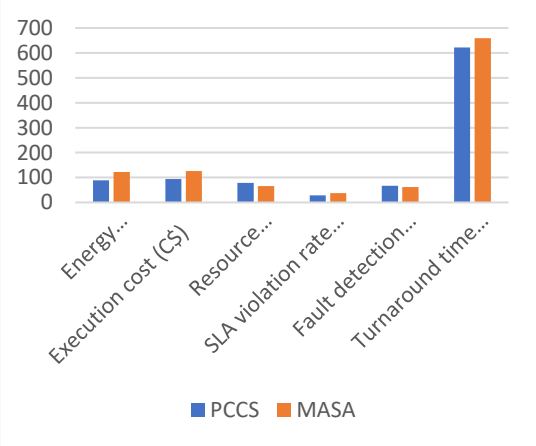

Figure 4. MASA and proposed PCCS comparison on different QoS parameters.

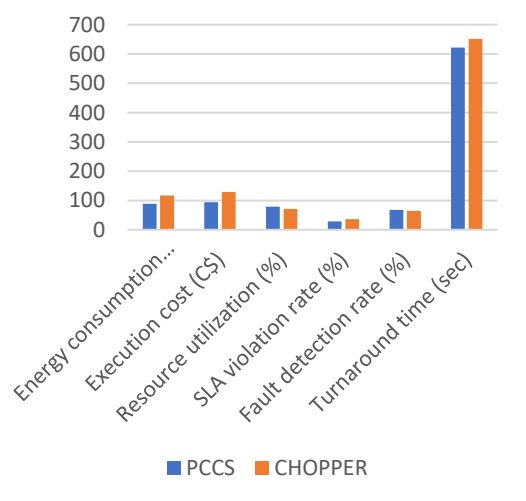

Figure 5. CHOPPER and proposed PCCS comparison on different QoS parameters.

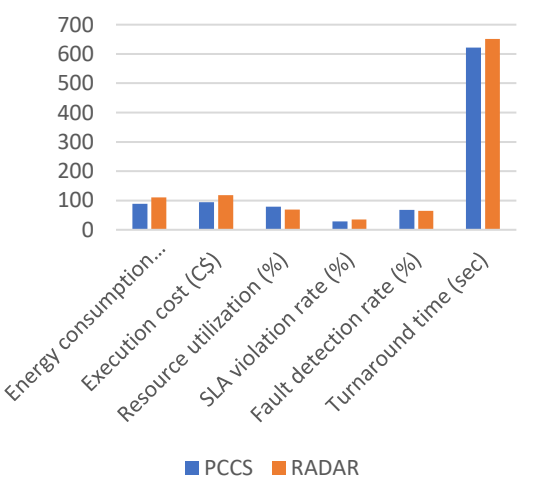

Figure 6. RADAR and proposed PCCS comparison on different QoS parameters.

Figure 3 clearly shows that the proposed PCCS improves resource utilization by 20.36%, fault detection rate by 9.93% and decrease energy consumption by 27.12%, execution cost by 26.98%, turnaround time by 6.63%, an SLA violation rate by 25.92% in comparison to SH-SLA. In Figure 4, the proposed PCCS technique comparative analysis simulated with MASA scheme and simulation results show that the

proposed scheme gives better results for fault detection rate by 8.45%, resource utilization by 16.56%, turnaround time by 5.51%, cost of execution by 25.07%, consumption of energy by 26.89% and rate of SLA violation by 25.05%. Figure 5 represents the simulation results of CHOPPER with PCCS proposed technique which justify that comparatively the proposed PCCS takes less energy consumption by 23.81%, execution cost by 26.35%, turnaround time by 4.89%, SLA violation rate by 18.23%, resource utilization by 8.64% and the fault detection rate by 4.57%. Figure 6 clearly shows that the PCCS improves resource utilization by 12.77% and fault detection rate by 4% and decreases energy consumption by 19.62%, execution cost by 20.65%, turnaround time by 4.50% and SLA violation rate by 20.84% in comparison to RADAR.

Our paper has discussed how the cloud provider provides better quality in a Cloud Environment during the user request for resources and management. As the SLA between user and provider is the most crucial document, the proposed prediction base management and PCCS have presented a predictive approach to resource management using a VM migration policy. That will effectively address the overloading problem and provide cloud resource prediction as per SLA for user QoS requirements, where no human intervention will improve user satisfaction.

The proposed model works based on prediction; therefore, configuration, healing, protection and optimization have been automatically done. Our simulation is done on CloudSim in terms of various parameters, such as throughput, reliability, fault detection rate, turnaround time, waiting time and SLA violation rate. According to the results, the proposed prediction-based approach is better than the existing SLA frameworks. Our proposed framework leads to improve the scalability of cloud-based services. Our proposed algorithm simulation reduces the cost and execution time; therefore, this will lead to saving energy. The simulation is done based on the number of VM migrations and SLA violations. Comparison of the results with those of the existing frameworks shows a reduction in VM migrations in energy consumption in the data center to measure energy consumption in terms of idle hosts.

5. CONCLUSION

In this paper, fault-tolerance using a self-healing SLA-based Predictive Cloud Computing System (PCCS) has been proposed with self-management property for heterogeneous workload execution. The main goal of our PCCS is to minimize the SLA violation rate and increase user satisfaction levels by fulfilling their QoS requirements. We propose a new model that uses VMs as a resource allocation unit that provisions threshold-based dynamic allocation of cloud computing resources that performs prediction on the future need of resources by the PCCS scheme. This scheme will prepare resources required as per the future need of the users' applications. The proposed method predicts the future demand of user applications based on historical databases of workload demands. The scheme makes resources ready for provision after predicting the required resource demands and fulfils the actual needs of the application without SLA violation. The proposed method can dynamically configure the necessary resources based on the threshold-based load balancing technique and maximize available cloud resource utilization with reduced user-usage cost. The PCCS improves average resource utilization, energy efficiency, fault detection rate and throughput. It minimizes average energy consumption, execution cost, missed deadlines, resource contention, waiting time and turnaround time. The simulation results show that the proposed PCCS performs better than existing resource provisioning techniques in terms of SLA violation rate.

Our work presented resource requirements prediction, but we have not included hard disk, traffic, network utilization and bandwidth for the forecast. Therefore, future research could focus on having more resources, such as hard disk and bandwidth, for the prediction model. The prospective study consists of work on network utilization and network traffic to maintain scalability of the proposed model.

REFERENCES

[1] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam and N. Sharma, "Towards Autonomic Workload Provisioning for Enterprise Grids and Clouds," Proc. of the 10th IEEE/ACM International Conference on Grid Computing, pp. 50-57, Banff, AB, Canada, Oct. 2009.

[2] T. Lorido-Botran, J. Miguel-Alonso and J. A. Lozano, "A Review of Auto-scaling Techniques for Elastic

Applications in Cloud Environments," Journal of Grid Computing, vol. 12, no. 4, pp. 559-592, Oct. 2014.

[3] Md. Toukir Imam, S. F. Miskhat, R. M. Rahman and M. A. Amin, "Neural Network and Regression-based Processor Load Prediction for Efficient Scaling of Grid and Cloud Resources," Proc. of the 14th IEEE International Conference on Computer and Information Technology (ICCIT 2011), pp. 333-338, Dhaka, Bangladesh, Dec. 2011.

[4] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A. A. Alelaiwi and F.-M. Li, "Minimizing SLA Violation and Power Consumption in Cloud Data Centers Using Adaptive Energy-aware Algorithms," Future Generation Computer Systems, vol. 86, pp. 836-850, 2018.

[5] J. Zhu, P. He, Z. Zheng and M. R. Lyu, "Online QoS Prediction for Runtime Service Adaptation *via* Adaptive Matrix Factorization," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 10, pp. 2911-2924, Oct. 2017.

[6] Shalu and D. Singh, "Swarm Intelligence Based Virtual Machine Migration Techniques in Cloud Computing," Proc. of the International Conference on Computation, Automation and Knowledge Management (ICCAKM), pp. 120-124, Dubai, United Arab Emirates, 2020.

[7] A. M. R. AlSobeh, S. AlShattnawi, A. Jarrah and M. M. Hammad, "WEAVESIM: A Scalable and Reusable Cloud Simulation Framework Leveraging Aspect-oriented Programming," Jordanian Journal of Computers and Information Technology (JJCIT), vol. 06, no. 02, pp. 182-201, June 2020.

[8] R. Yadav, W. Zhang, K. Li et al., "Managing Overloaded Hosts for Energy-efficiency in Cloud Data Centers," Cluster Computing, vol. 2021, DOI: 10.1007/s10586-020-03182-3, Feb. 2021.

[9] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah and M. A. Alzain, "A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications," IEEE Access, vol. 9, pp. 41731-41744, 2021.

[10] S. K. Pande, S. K. Panda, S. Das, K. S. Sahoo, A. K. Luhach et al., "A Resource Management Algorithm for Virtual Machine Migration in Vehicular Cloud Computing," Computers, Materials & Continua, vol. 67, no.2, pp. 2647–2663, 2021.

[11] M. A. Shahid, N. Islam, M. M. Alam, M. M. Su'ud and S. Musa, "A Comprehensive Study of Load Balancing Approaches in the Cloud Computing Environment and a Novel Fault Tolerance Approach," IEEE Access, vol. 8, pp. 130500-130526, 2020.

[12] Z. Chen, K. Lin, B. Lin, X. Chen, X. Zheng and C. Rong, "Adaptive Resource Allocation and Consolidation for Scientific Workflow Scheduling in Multi-cloud Environments," IEEE Access, vol. 8, pp. 190173-190183, 2020.

[13] B. Gul et al., "CPU and RAM Energy-based SLA-aware Workload Consolidation Techniques for Clouds," IEEE Access, vol. 8, pp. 62990-63003, 2020.

[14] R. Yadav, W. Zhang, K. Li et al., "An Adaptive Heuristic for Managing Energy Consumption and Overloaded Hosts in a Cloud Data Center," Wireless Networks, vol. 26, pp. 1905–1919, April 2020.

[15] W. Dargie, "Estimation of the Cost of VM Migration," Proc. of the 23rd IEEE International Conference on Computer Communication and Networks (ICCCN), pp. 1-8, Shanghai, China, 2014.

[16] S. Singh, I. Chana, M. Singh et al., "SOCCER: Self-optimization of Energy-efficient Cloud Resources," Cluster Computing, vol. 19, no. 4, pp. 1787–1800, Sep. 2016.

[17] M. Sohani and S. C. Jain, "A Predictive Priority-based Dynamic Resource Provisioning Scheme with Load Balancing in Heterogeneous Cloud Computing," IEEE Access, vol. 9, pp. 62653-62664, April 2021.

[18] F. Yao, C. Pu and Z. Zhang, "Task Duplication-based Scheduling Algorithm for Budget-constrained Workflows in Cloud Computing," IEEE Access, vol. 9, pp. 37262-37272, 2021.

[19] H. M. Khan, G. Chan and F. Chua, "An Adaptive Monitoring Framework for Ensuring Accountability and Quality of Services in Cloud Computing," Proc. of the International Conference on Information Networking (ICOIN), pp. 249-253, Kota Kinabalu, Malaysia, 2016.

[20] R. Latif, S. H. Afzaal and S. Latif, "A Novel Cloud Management Framework for Trust Establishment and Evaluation in a Federated Cloud Environment," The Journal of Supercomputing, vol. 2021, DOI: 10.1007/s11227-021-03775-8, April 2021.

[21] A. Mosallanejad, R. Atan, M. Azmi Murad and R. Abdullah, "A Hierarchical Self-healing SLA for Cloud Computing," International Journal of Digital Information and Wireless Communications (IJDIWC), vol. 4, no. 1, pp. 43-52, 2014.

[22] S. S. Gill, I. Chana, M. Singh and R. Buyya, "RADAR*:* Self-configuring and Self-healing in Resource

Management for Enhancing Quality of Cloud Services," Concurrency and Computation: Practice and Experience, vol. 31, no. 1, DOI: 10.1002/cpe.4834, Aug. 2018.

[23] S. Banerjee, S. Roy and S. Khatua, "Efficient Resource Utilization Using Multi-step-ahead Workload Prediction Technique in Cloud," The Journal of Supercomputing, vol. 2021, DOI: 10.1007/s11227-021-03701-y, March 2021.

[24] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy and Y. Tian, "Adaptive Energy-aware Algorithms for Minimizing Energy Consumption and SLA Violation in Cloud Computing," IEEE Access, vol. 6, pp. 55923-55936, 2018.

[25] S. Sotiriadis, N. Bessis and R. Buyya, "Self-managed Virtual Machine Scheduling in Cloud Systems," Information Sciences, vol. 433-434, pp. 381–400, 2018.

[26] A. Paya and D. C. Marinescu, "Energy-aware Load Balancing and Application Scaling for the Cloud Ecosystem," IEEE Transactions on Cloud Computing, vol. 5, no. 1, pp. 15-27, 2017.

[27] I. Odun-Ayo, B. Udemezue and A. Kilanko, "Cloud Service Level Agreements and Resource Management", Advances in Science, Technology and Engineering Systems Journal, vol. 4, no. 2, pp. 228-236, 2019.

[28] R. Yadav, W. Zhang, H. Chen and T. Guo, "MuMs: Energy-aware VM Selection Scheme for Cloud Data Center," Proc. of the 28th IEEE International Workshop on Database and Expert Systems Applications (DEXA), pp. 132-136, Lyon, France, 2017.

[29] M. Dabbagh, B. Hamdaoui, M. Guizani and A. Rayes, "Toward Energy-efficient Cloud Computing: Prediction, Consolidation and Over-commitment," IEEE Network, vol. 29, no. 2, pp. 56-61, 2015.

[30] E. Torre, J. J. Durillo, V. de Maio, P. Agrawal, S. Benedict, N. Saurabh and R. Prodan, "A Dynamic Evolutionary Multi-objective Virtual Machine Placement Heuristic for Cloud Data Centers," Information and Software Technology, vol. 128, DOI: 10.1016/j.infsof.2020.106390, 2020.

[31] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu and H. Tenhunen, "Energy-aware VM Consolidation in Cloud Data Centers Using Utilization Prediction Model," IEEE Transactions on Cloud Computing, vol. 7, no. 2, pp. 524-536, 2019.

[32] D. Abdulkareem Shafiq, N. Z. Jhanjhi and A. Abdullah, "Load Balancing Techniques in Cloud Computing Environment: A Review," Journal of King Saud University - Computer and Information Sciences, DOI: 10.1016/j.jksuci.2021.02.007, 2021.

[33] M. Balaji, Ch. Aswani Kumar and G. Subrahmanya V. R. K. Rao, "Predictive Cloud Resource Management Framework for Enterprise Workloads," Journal of King Saud University - Computer and Information Sciences, vol. 30, no. 3, pp. 404-415, 2018.

[34] F. Ebadifard and S. M. Babamir, "Autonomic Task Scheduling Algorithm for Dynamic Workloads through a Load Balancing Technique for the Cloud-computing Environment," Cluster Computing, vol. 24, pp. 1075-1101, June 2021.

[35] N. Chaurasia, M. Kumar, R. Chaudhry et al., "Comprehensive Survey on Energy-aware Server Consolidation Techniques in Cloud Computing," The Journal of Supercomputing, vol. 2021, DOI: 10.1007/s11227-021-03760-1, March 2021.

[36] B. K. Dewangan, A., M., V. Agarwal and A. Pasricha, "Energy-aware Autonomic Resource Scheduling Framework for Cloud," International Journal of Mathematical, Engineering and Management Sciences, vol. 4, no. 1, pp. 41-55, 2019.

[37] A. A. Hassan, B. M. Bai and T. J. Gandomani, "An Integrated Model for Secure-on-Demand Resource Provisioning Based on Service Level Agreement (SLA) in Cloud Computing," Journal of Theoretical and Applied Information Technology, vol. 65, no. 2, July 2014.

[38] M. Sohani and S. C. Jain, "State-of-the-art Survey on Cloud Computing Resource Scheduling Approaches," Proc. of Ambient Communications and Computer Systems, Part of the Advances in Intelligent Systems and Computing Book Series, vol. 696, pp. 629-639, March 2018.

[39] W. Lin, J. Z. Wang, C. Liang and D. Qi, "A Threshold-based Dynamic Resource Allocation Scheme for Cloud Computing," Procedia Engineering, vol. 23, pp. 695-703, 2011.

[40] S. S. Gill, I. Chana, M. Singh et al., "CHOPPER: An Intelligent QoS-aware Autonomic Resource Management Approach for Cloud Computing," Cluster Computing, vol. 21, pp. 1203–1241, 2018.

[41] H. Alhussian et al., "Investigating the Schedulability of Periodic Real-time Tasks in Virtualized Cloud

Environment," IEEE Access, vol. 7, pp. 29533-29542, 2019.

[42] M. Azaiez and W. Chainbi, "A Multi-agent System Architecture for Self-healing Cloud Infrastructure," Proceedings of the International Conference on Internet of Things and Cloud Computing (ICC'16), pp. 1-6, DOI: 10.1145/2896387.2896392, March 2016.

[43] S. Talwani and I. Chana, "Fault Tolerance Techniques for Scientific Applications in Cloud," Proc. of the 2nd International Conference on Telecommunication and Networks (TEL-NET), pp. 1-5, Noida, India, 2017.

[44] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," Software – Practice and Experience, vol. 41, no. 1, pp. 23–50, August 2010.

ملخص البحث:

ظهرت حديثاً إدارة فاعلية التطبيقات كجزء أساسي من خدمات الحوسبة السّحابية على الشبكة العنكبوتية. ويقدّم مزود الخدمة السّحابية خدماتٍ متنوعة على أساس الدّفع مقابل الاستخدام؛ الأمر الذي يستلتزم الرّصد والقياس للخدمات المقدمة من أجل إدارة جودة الخدمة من خلال إنترنت الأشياء ومن ثم تحقيق الشّروط الواردة في اتفاقية مستوى الخدمة. ومع ذلك، فإنّ تجنُّب خرق الاتفاقية وضمان تحقيق متطلبات المستخدمين الديناميية يشكلان تحدياً في الحوسبة السّحابية. ويتعاظم التّعقيد وعدم التجانس والديناميية بسرعةٍ مما يجعل أُطر العمل المتعلقة بالحوسبة السّحابية عصيّة على الإدارة الفاعلة وغير موثوقة. وإنّ الأنظمة السّحابية تحتاج الى إدارة ذاتية للخدمات للتغلُّب على تلك المشكلات. لذا، فإنّ ثمّة حاجة الى تطوير خطة لاتّخاذ التّدابير اللازمة التي من شأنها أن تلبّي متطلبات المستخدمين لأنظمة الحوسبة السّحابية فيما يرتبط بجودة الخدمة وتحول دون خرق اتفاقية مستوى الخدمة. هذه الورقة تقدم تقنيةً مقترحة لإدارة المصادر قائمة على التّوقُّع تسمى (النّظام التّوقُّعي للحوسبة السّحابية). وفي هذا النظام، يتم التركيز على التّوقُّع المستند على الالتئام الذّاتي الذي يُعالج الأخطاء غير المتوقعة، وعلى التّوقُّع المستند على التّرتيب الذاتي للمصادر المتعلقة بالتّطبيقات. وقد جرى تقييم النّظام المقترح في المحاكي السّحابي. وكشفت نتائج المحاكاة أنّ النّظام المقترح تفوّق من حيث الأداء على تقنياتٍ قائمة من حيث زمن التّنفيذ، والجدوى المتعلقة بالتكلفة، وتعارُض المصادر، وخرق اتفاقيات مستوى الخدمة؛ في الوقت الذي قدّم فيه خدماتٍ موثوقة.

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).