The Journal of Supercomputing (2022) 78:8003–8024
https://doi.org/10.1007/s11227-021-04235-z

OFP-TM: an online VM failure prediction and tolerance model towards high availability of cloud computing environments

Deepika Saxena[1] · **Ashutosh Kumar Singh**[1]

Accepted: 26 November 2021 / Published online: 6 January 2022

Abstract
The indispensable collaboration of cloud computing in every digital service has raised its resource usage exponentially. The ever-growing demand of cloud resources evades service availability leading to critical challenges such as cloud outages, SLA violation, and excessive power consumption. Previous approaches have addressed this problem by utilizing multiple cloud platforms or running multiple replicas of a Virtual Machine (VM) resulting into high operational cost. This paper has addressed this alarming problem from a different perspective by proposing a novel 𝕆nline virtual machine 𝔽ailure ℙrediction and 𝕋olerance 𝕄odel (OFP-TM) with high availability awareness embedded in physical machines as well as virtual machines. The failure-prone VMs are estimated in real-time based on their future resource usage by developing an ensemble approach-based resource predictor. These VMs are assigned to a failure tolerance unit comprising of a resource provision matrix and Selection Box (S-Box) mechanism which triggers the migration of failure-prone VMs and handle any outage beforehand while maintaining the desired level of availability for cloud users. The proposed model is evaluated and compared against existing related approaches by simulating cloud environment and executing several experiments using a real-world workload Google Cluster dataset. Consequently, it has been concluded that OFP-TM improves availability and scales down the number of live VM migrations up to 33.5% and 83.3%, respectively, over without OFP-TM.

Keywords Cloud outage · Service availability · Mean-time-between-Failure · Mean-time-to-repair · Servers · Virtual machines

✉ Deepika Saxena
13deepikasaxena@gmail.com

Ashutosh Kumar Singh
ashutosh@nitkkr.ac.in

[1] Department of Computer Applications, National Institute of Technology, Kurukshetra, India

1 Introduction

Commercial cloud services have strengthened the online and digital business enterprises with their surplus benefits including maximum computing at minimum capital investment, reliability, elasticity and scalability of resources [1–4]. The dependency of cloud users on third-party resources has increased manyfolds that will continue to grow in future [5–7]. Undoubtedly, supplying of high availability of cloud services is one of the biggest challenges for the Cloud Service Providers (CSPs) because of the enormously growing demand and dependability of every organization on the cloud infrastructure [8–11]. Though the CSP adheres to Service Layer Agreement (SLA) and take responsibility of cloud infrastructure and ensures service availability and security by all means [12–14] cloud outage occurs, due to fluctuating and dynamic resource utilization [15–19].

A recent survey of COVID-19 pandemic depicted in Fig. 1 reveals that cloud outages have dominated the titans in the market, including Zoom, Microsoft Azure, and Google Cloud Platform, Amazon Web Services, Salesforce, IBM Cloud etc. [20]. For instance, Microsoft Azure faced six hours and around five hours outages on 3*rd* and 24*th* March, 2020 respectively, due to failure of cooling system that caused a reduction in airflow, and the subsequent thermal spikes throughout the data center which have stave off the performance of network compute and storage devices. The users of Google Cloud services and websites including YouTube, Gmail, Google Assistant, and Google Docs have faced downtime approximately for an hour on December 14, 2020, after being hit with a widespread cloud outage that had affected both commercial as well as personal services of Google. Also, the e-commerce giant's cloud divisions such as Amazon Web Services (AWS) and Microsoft were down on November 26, 2020, and April 21, 2020 (for around five hours), respectively.

These cloud outages could increase in frequency and severity which have raised a critical question of reliability on services for cloud users because even the large swathes of internet apps, services, and websites are operating at the mercy of these technical organizations. Furthermore, it raises a biggest challenge for CSP, i.e. how

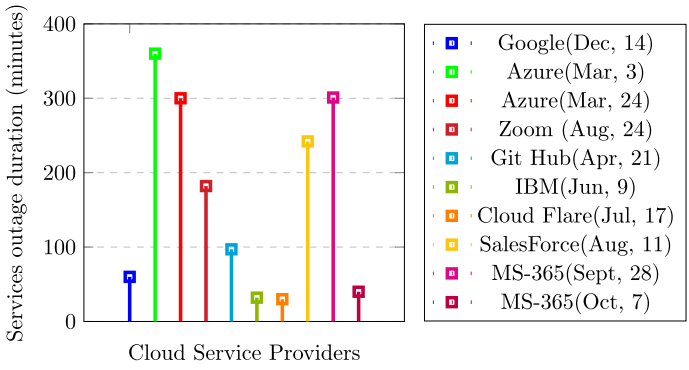


Fig. 1 Cloud outage 2020

to handle these cloud outages and ensure high availability while maintaining energy-efficient load balancing constraints?

Generally, the average availability of services is quantified in terms of mean time between failures ($\mathbb{MTBF}$) and mean time to repair ($\mathbb{MTTR}$) [21, 22]. Therefore, maximizing $\mathbb{MTBF}$ and minimizing $\mathbb{MTTR}$ by proactive detection of resource failures, analysing historical and current performance of the physical machines, and concurrent handling of any outage beforehand provides an efficient solution for improving the availability of cloud services. The existing works have resolved the cloud outages by using proactive as well as reactive methods [23–25]. The proactive methods of failure handling depend on the prior knowledge of the failed VMs. Mostly, the VMs get fail due to over-utilization or limitation of sufficient resources, required softwares that are not installed properly, or execution time exceeding the deadline, physical machine running out of memory/disk space, and so on [26]. Contrary to this, the reactive methods are triggered at the occurrence of actual failure. These methods include checkpointing, replication, retry, application resubmission etc. [27] which delays the execution of user applications.

Our Contributions This paper proposes an $\mathbb{O}$nline virtual machine $\mathbb{F}$ailure $\mathbb{P}$rediction and $\mathbb{T}$olerance $\mathbb{M}$odel (OFP-TM) which ingrains High Availability (HA) enhancing features in servers as well as VMs. The proposed approach integrates proactive VM failure prediction and failure tolerance to provide a comprehensive solution of the resource contention-based cloud services failure by developing and employing an ensemble resource predictor for a proactive estimation of any VM failure due to resource deficiency. The failure-prone VMs are assigned to a failure tolerance unit for handling of such cloud outage and maintain an expected level of cloud services availability for the cloud users.

The key contributions of the proposed framework include:

- An ensemble method-based failure predictor is developed to forecast the multiple resource-based contention respective to each VM for estimation of its failure status proactively.
- A Failure Tolerance Unit is deployed to trigger the necessary failure elimination actions and decide an appropriate allocation for the predicted failure-prone VMs.
- The performance evaluation of OFP-TM by using real benchmark Google Cluster dataset reveals that it outperforms the state-of-art approaches in terms of various performance metrics, like Service availability, VM failure reduction, server overload prediction, resource utilization, reduction of power consumption and VM migration cost.

Organization: The paper is organized as follows: Sect. 2 presents a recent related work. Section 3 entails a comprehensive description of proposed framework. Online failure prediction by developing Ensemble approach-based Resource Predictor (ERP) is discussed in Sect. 4. The description of Failure Tolerance is given in Sect. 5. The operational design and complexity of OFP-TM is entailed in Sect. 6. The performance evaluation and comparison of OFP-TM with state-of-the-arts is presented in Sect. 7. Finally, the paper is concluded with conclusive remarks and future scope of the proposed work in Sect. 8.

2 Recent related work

A Deep Neural Network (DNN)-based failure predictor is trained and utilized to classify the upcoming workload on VMs (i.e. arriving tasks) into 'failure-prone tasks' and 'non-failure-prone tasks' in [28]. Two separate task scheduling algorithms are proposed to allocate both types of tasks to the most suitable server. Three consecutive failure-prone tasks are taken and each task is replicated into three copies which are scheduled by constructing a super task vector on most suitable energy-efficient server. It allows the execution of different copies of the tasks on different servers, thus resisting overlapping and redundant execution. The performance of the work is investigated using Internet and Euler datasets which improves fault tolerance and energy efficiency as well.

A failure aware and energy-efficient (FAEE) VM placement scheme is proposed in [29] which predicts VM failure using an exponential smoothing-based forecasting technique. Accordingly, two fault tolerance methods including VM migration and VM checkpointing are triggered to handle any failure and ensure service availability. A simulation-based evaluation of this work was conducted using Grid5000 Failure Traffic Analysis (FTA) dataset. Furthermore, this work concludes that a significant improvement in terms of energy efficiency and reliability can be obtained by considering the failure characteristics of physical resources.

Bui et al. proposed an early fault detection method in [30], which engaged a fuzzy logic algorithm and Gaussian process prediction technique. This method was based on a rigorous analysis on the characteristics or nature of failures. The method entailed an improved performance in terms of failure prediction accuracy and reaction rate, which subsequently enhanced the reliability of the entire cloud system. Later, Nguyen et al. [31] proposed a dynamic resource allocation scheme to tackle the problem of energy-efficient distribution of load on available hosts. To achieve the objective, an OpenStack-based cloud platform was set up to implement the proposed method with live VM migration. The power distribution and corresponding energy consumption was monitored which revealed that the proposed method proficiently distributes the VM resources and minimizes the energy consumption simultaneously.

A fault prediction system is developed for distributed computing Hadoop clusters using trained Support Vector Machine (SVM) model in [32]. This fault predictor is trained with a non-anomalous dataset during different operation patterns like boot-up, shutdown, idle, task allocation, resource distribution etc. so that the it can adapt its parameters to detect and classify between normal and abnormal situation. Xu et al. proposed a disk failure prediction approach in [33] using Multiple Additive Regression Trees gradient boosting (MART-GB) algorithm. It ranks all disks according to the degree of error-proneness and enable live migration of existing VMs and allocation of new VMs to the healthy disks, thus improving service availability. This work is evaluated using real-world cloud data which shows its superior performance over Random Forest and SVM-based prediction models.

Wang et al. proposed a fault-tolerant elastic scheduling algorithms for real-time tasks (FESTAL) in cloud environment [34]. This work provided a fault-tolerant

VM scheduling algorithms which accommodated virtualization technology and an appropriate VM migration strategy along with battery back-up features. The experiment-based performance evaluation and comparison of FESTAL using synthetic and Google data traces reveals its efficiency in terms of improved performance of cloud services.

Another work based on battery back-up scheduling schemes for fault-tolerant execution of scientific workflows by incorporating task allocation and message transmission features, is presented in [35]. It proposed a dynamic fault-tolerant scheduling algorithm that employed a backward shifting approach to use idle physical resources consciously by inducing task overlapping and VM migration features. Additionally, vertical and horizontal VM scaling-up techniques are utilized to deal with fluctuations and changes of dynamic workflows demands in real-time. This work enhanced the resource utilization and scalability even in the presence of physical node failures.

Sivagami et al. [36] presented an Assured Virtual Cloud Data Center framework for fault tolerance and availability of cloud data centers. A load monitoring and balancing algorithm is proposed to estimate the load on virtual links and distribute the available physical resources concisely among the VMs. In case of VM failure, it selects supporting VM considering network topology, load distribution, and availability of physical resources. The simulation and experimentation of this work entailed that the proposed method outperformed the state-of-the-art techniques by increasing the cloud survivability with reduced complexity.

Vinay et al. [37] have proposed a fault-tolerant scheduling algorithm with a bidding strategy for scientific workflows. This work focused on minimizing the volatility and cost of resource provisioning for scientific workflows. The algorithm used spot and blockspot instances as hybrid instances as compared to on-demand instances to minimize the cost of execution and number of faults while following the deadline constraints. The achieved evaluations demonstrated via experimental simulation reveal that this algorithm has potential and shows robustness under short deadlines with minimal makespan.

Ghoreyshi et al. [38] proposed a VM migration method to mitigate fault problem while satisfying Service Level Agreement (SLA) violation and energy consumption constraints. The VM migration was based on the ratio of least increasing energy consumption and the minimal deadline missed. This work has depicted that an increase in energy consumption and increase in failure rate have an exponential relationship. Its performance evaluation reveals that by adopting this method for VM migration the number of SLA violations would minimize.

Li et al. [39] presented an energy-efficient and fault-tolerant VM replica management policy while meeting the deadline and budget constraints in the edge-cloud environment. This work proposed energy-aware VM cluster scaling strategy for reduction of power consumption and achievement of energy efficiency by activating and deactivating up the data nodes according to the load status of the system. Also, a node recovery method based on availability metrics is employed to handle the node failures. The experimental evaluation and comparison of this proposed method shows its superior performance over existing approaches.

Table 1 compares OFP-TM with the aforementioned state-of-the-art works with respect to different performance metrics.

In the light of above discussion, it can be concluded that existing works have attempted to solve the challenge of VM failures by predicting the resource failures. Others have used reactive and proactive migration of VMs on the occurrence of resource contention disregarding VM availability and SLA features during predictive resource management. In contrast, the proposed OFP-TM provides a comprehensive solution by accurately predicting the VM failures and automatic triggering of VM replication and migration to mitigate its effect proactively. Also, the prediction of resource utilization of VMs assists in alleviating server over/under-load, reducing resource and power wastage as well.

Table 2 shows the list of symbols with their explanatory terms that have been used throughout the paper.

3 Proposed framework

Consider a datacenter containing P physical machines or servers $\{S_1, S_2,..., S_P\} \subseteq \mathbb{S}$ hosts Q VMs such that $\{V_1^{S_1}, V_2^{S_1},..., V_x^{S_1}\}$ hosted on S_1, $\{V_1^{S_2}, V_2^{S_2},..., V_x^{S_2}\}$ and $\{V_1^{S_P}, V_2^{S_P},..., V_x^{S_P}\}$ are deployed on servers S_2 and S_P, respectively. These VMs are employed for execution of different job requests of M users $\{U_1, U_2,..., U_M\} \subseteq \mathbb{U}$ as demonstrated in Fig. 2. Each server consists of a VM Hypervisor layer that enables deployment of different VMs by creating a layer of virtual isolation among them. Below hypervisor,

Table 1 Comparison of performance metrics: OFP-TM model v/s state-of-the-arts

Approach	Failure prediction	Failure tolerance	MTBF, MTTR	Availability	Resource utilization	Power consumption	VM migration
[28]	DNN	✓	×	×	✓	✓	×
[29]	Exponential smoothing	✓	✓	×	×	✓	×
[30]	Fuzzy logic & Guassian process	×	×	×	✓	×	×
[31]	ESNemble time-series	×	×	×	✓	✓	✓
[32]	SVM	×	×	×	×	×	×
[33]	MART-GB	×	×	×	×	×	×
[34]	×	✓	×	×	✓	×	×
[35]	×	✓	×	×	✓	×	×
[36]	×	✓	×	×	✓	✓	×
[37]	×	✓	×	×	×	×	×
[38]	×	✓	×	×	×	✓	✓
[39]	×	✓	×	×	✓	✓	×
OFP-TM	Ensemble prediction	✓	✓	✓	✓	✓	✓

Table 2 Notations

Symbols	Explanation terms
S	Server
V	Virtual machine
V^F	Failure-prone VM
U	User
P, Q, M	Number of servers, VMs, users
ω	Mapping between server and VM
α_i	Status of i *th* server
$\mathcal{RU}$	Resource utilization
$\mathcal{PW}$	Power consumption
y	Resource index
d	Data samples
BP	Base predictor
z_{actual}	Actual output
$z_{predicted}$	Predicted output
m	Number of data samples
$\mathbb{R}$	Resources
$\mathbb{D}$	Normalized data
D^*	Historical data
N	Number of failure-prone VMs
n	Total number of base predictors
$\mathbb{X}$	Number of active resources
UT, DT	Uptime, Downtime
$\mathbb{E}_{rmse}$	Root mean squared error
C, Mem	CPU, Memory
BW	Bandwidth

server comprises of a pool of resources for computation, storage, and networking. The power consumption analyser ($\mathcal{PA}$), and resource analyser ($\mathcal{RA}$) are also accomodated within each server to measure the consumption of power and utilization of resources by the respective server.

The power consumption analyser ($\mathcal{PA}$) analyses the rate of electrical power consumption ($\mathcal{PW}$) of a server over. The power consumption for i *th* server (i.e. $\mathcal{PW}_i$) is estimated using Eq. (1), where $\mathcal{PW}_i^{max}$, $\mathcal{PW}_i^{min}$ and $\mathcal{PW}_i^{idle}$ are maximum, minimum and idle state power consumption of i *th* server and $\mathcal{RU}$ is CPU utilization of the respective server. The total power consumption of entire datacenter ($\mathcal{PW}_{dc}$) over time-interval $\{t_1, t_2\}$ is estimated using Eq. (2).

$$\mathcal{PW}_i = [\mathcal{PW}_i^{max} - \mathcal{PW}_i^{min}] \times \mathcal{RU} + \mathcal{PW}_i^{idle} \tag{1}$$

$$\mathcal{PW}_{dc} = \sum_{i=1}^{P} \mathcal{PW}_i \tag{2}$$

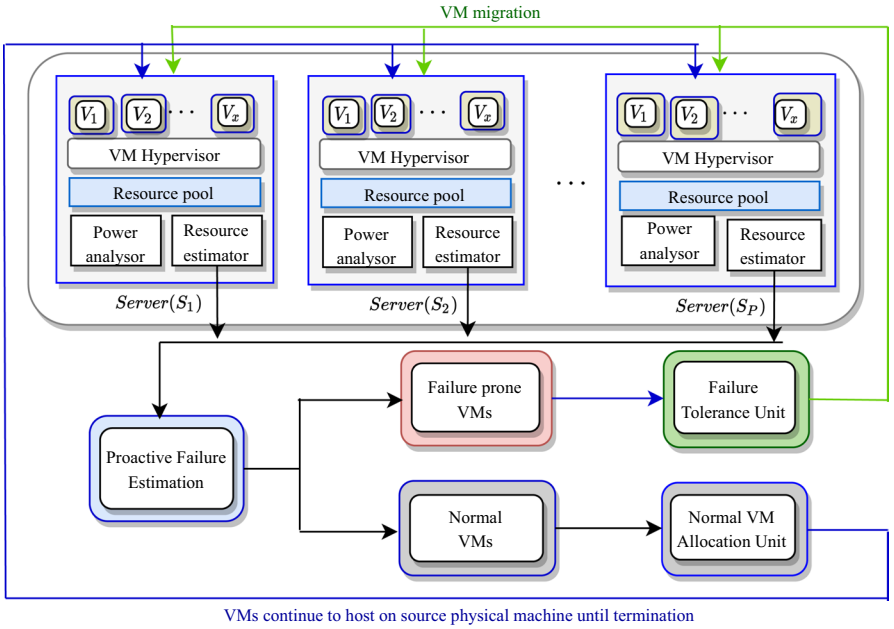


Fig. 2 Online failure prediction and tolerance model

The resource analyser ($\mathcal{RA}$) is employed to forecast the resource usage of each VM on a server and aggregate the expected resource utlilization for estimation of the resource contention i.e. overload status of a server before its actual occurrence. The VMs hosted on overloaded server may fail in future due to resource deficiency and thus becomes 'failure prone'. The detailed description of resource capacity utilization-based failure prediction is provided in Sect. 4. The resource utilization ($\mathcal{RU}$) of datacenter can be obtained using Eqs. (3) and (4), where α_i represents status of i *th* server (i.e. $\alpha_i = 0$ for an inactive server while $\alpha_i = 1$ for an active server), $\mathbb{X}$ is the number of resources. Though in formulation, only CPU (C), bandwidth (BW), and memory (Mem) are considered, this equation is extendable to accomodate any number of resources.

$$\mathcal{RU}_{dc} = \int_{t_1}^{t_2} \left(\frac{\sum_{y=1}^{\mathbb{X}} \mathcal{RU}_y^{\mathbb{R}}}{|\mathbb{X}| \times \sum_{i=1}^{P} \alpha_i} \right) \mathrm{d}t \quad \mathbb{R} \in C, Mem, BW \tag{3}$$

$$\mathcal{RU}_y^{\mathbb{R}} = \sum_{i=1}^{P} \frac{\sum_{j=1}^{Q} \omega_{ji} \times v_j^{\mathbb{R}}}{S_i^{\mathbb{R}}} \quad \mathbb{R} \in C, Mem, BW \tag{4}$$

The resources utilization of all the VMs hosted on each server are periodically estimated and monitored to determine the probability of any VM failure proactively.

Accordingly, the respective set of VMs is categorized into two subsets including *Failure-prone VMs* and *Normal VMs*. The operation and allocation of all the failure-prone VMs are decided by the Failure Tolerance Unit (FTU) which executes operations such as VM replication, recovery and migration to prevent the respective VM failure beforehand. On the contrary, the Normal VMs continue to operate at the same server until it is terminated by the user. The current status of all the failure-prone VMs is replicated into new VM instances to be hosted on the other suitable physical machine. The essential constraints that must be satisfied before each VM placement and migration are stated in Eq. (5), where $V_i^{\mathbb{R}}$ denotes resource usage of *i th* VM; $\mathbb{R}$ represents resources viz., CPU (C), memory (M) and bandwidth (BW), respectively, for assignment of *i th* VM (V_i) at *k th* server (S_k).

$$\sum_{i=1}^{Q} V_i^{\mathbb{R}} \times \omega_{ik} \leq S_k^{\mathbb{R}}; \quad \forall_k \in \{1, 2, \ldots, P\}, \mathbb{R} \in \{C, Mem, BW\} \tag{5}$$

Furthermore, the information collected from $\mathcal{PA}$ and $\mathcal{RA}$ assists in analysing the status of failure for each physical machine over consecutive time-intervals $\{t_1, t_2\}$. If physical machine status predictor anticipates any failure due to resource contention such as hard disk (memory), networking device (network), and CPU (compute) failure, all the failure-prone VMs are migrated from the respective to selected energy-efficient physical machines (PMs).

4 Online failure prediction

The proactive failure of users' VMs is estimated according to the predicted resource requirement of different VMs hosted on a server. An *Ensemble Prediction Model* (EPM) is developed for the prediction of resource requirement of VMs for users' requests execution in future. Figure 3 portrays an ensemble of multiple base predictors-based Failure Prediction Unit (FPU) comprising of three key operations namely *Data preparation*, *Prediction*, and *Output generation*.

Data preparation Initially, the data are prepared from the historical resource usage information of different resources $\{\mathbb{R}_1, \mathbb{R}_2, ..., \mathbb{R}_y\} \in \mathbb{R}$, gathered from the respective users' VMs deployed on different servers, where $\{1, 2, \ldots, y\} \in \mathbb{X}$ is the number of resources. The data preparation comprises three consecutive operations viz., *Attribute selection* from the historical resource usage samples such as CPU and memory utilization, *Aggregation* of usage values of selected resources per unit time $\{t_1, t_2\}$, and *Normalization* of aggregated values in the range [0, 1] by applying Eq. (6).

$$\mathbb{D} = \frac{D_i^* - D_{min}^*}{D_{max}^* - D_{min}^*} \tag{6}$$

where D_{min}^* and D_{max}^* are the minimum and maximum values, respectively, of the input data set (D^*: $\{d_1^*, d_2^*, ..., d_l^*\}$). The normalized vector $\mathbb{D}$ is a set of normalized data values of utilization of a particular resource such that $\{d_1, d_2, \ldots, d_l\} \in \mathbb{D}$.

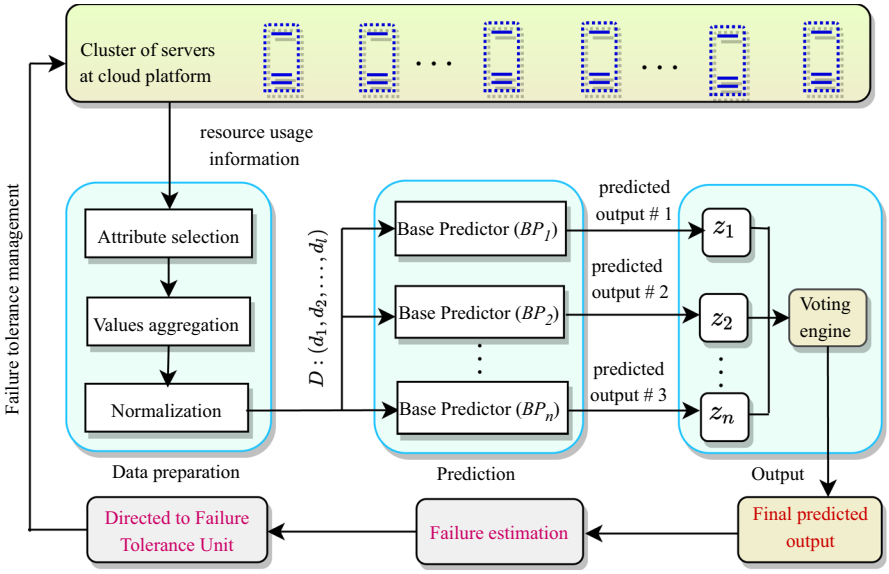


Fig. 3 Failure prediction unit

Let $l+1th$ utilization of j *th* resource ($\mathbb{R}_j$) depends on the previous l utilization of the respective resource such that $\mathbb{D}^{\mathbb{R}_j}$: $\{d_1^{\mathbb{R}_j}, d_2^{\mathbb{R}_j}, \ldots, d_l^{\mathbb{R}_j}\}$ which is arranged in two dimensional input ($\mathbb{D}_{input}^{\mathbb{R}_j}$) and output ($\mathbb{D}_{output}^{R}$) matrix as shown in Eq. (7), where R_i represents capacity utilization of i *th* resource.

$$\mathbb{D}_{input}^{\mathbb{R}_j} = \begin{bmatrix} d_1^{\mathbb{R}_j} & d_2^{\mathbb{R}_j} & .... & d_l^{\mathbb{R}_j} \\ d_2^{\mathbb{R}_j} & d_3^{\mathbb{R}_j} & .... & d_{l+1}^{\mathbb{R}_j} \\ . & . & .... & . \\ . & . & .... & . \\ d_m^{\mathbb{R}_j} & d_{m+1}^{\mathbb{R}_j} & .... & d_{l+m-1}^{\mathbb{R}_j} \end{bmatrix} \mathbb{D}_{output}^{\mathbb{R}_j} = \begin{bmatrix} d_{l+1}^{\mathbb{R}_j} \\ d_{l+2}^{\mathbb{R}_j} \\ . \\ . \\ d_{l+m}^{\mathbb{R}_j} \end{bmatrix} \tag{7}$$

Prediction The ensemble approach utilizes different base predictors or learners to estimate single prediction, i.e. the predicted output of n different base predictors (BP_1, BP_2,...,BP_n) are considered before generation of final output. In the proposed FPU, three base predictors are involved: (i) Feed-forward Neural Network (FNN), (ii) Support Vector Machine (SVM), and (iii) Linear Regression (LR) for estimation of the future resource usage $\mathcal{RU}$ of a VM.

– *FNN* A three-layered feed-forward neural network consisting of 1 input layer with 7 nodes, 1 hidden layer with 10 nodes, and 1 output layer with 1 node. The neural network connection weights are randomly generated in the range of [0, 1]. It receives historical resource usage data samples from the host VM which are normalized in the range [0, 1] to feed normalized 7 values into the input layer. This FNN is optimized using an evolutionary algorithm named

'Differential Evolutionary Algorithm', though any optimization algorithm could be used for training the neural network.

– *SVM* The SVM-based prediction model considered the points lying within the decision boundary lines, where the best fit line is the hyperplane having a maximum number of points. Let Eq. (8) represents a hyperplane (Y) and Eqs. (9) and (10) enumerate $+a$ and $-a$ which are the distances of upper and lower decision boundaries from hyperplane, respectively. Accordingly, any hyperplane that satisfied Eq. (11) builds a satisfactory SVM-based prediction model.

$$Y = wx + b \tag{8}$$

$$wx + b = +a \tag{9}$$

$$wx + b = -a \tag{10}$$

$$-a \leq Y - (wx + b) \leq +a \tag{11}$$

SVM is a supervised learning algorithm that acknowledges the presence of non-linearity in the data samples and establishes a proficient prediction model.

– *LR* A linear prediction model is represented by an equation $Y = a + bX + e$, where a is intercept, b is the slope of the line and e is the error term. This equation is used to forecast the value of a target variable based on the given training input prediction variables.

The proposed ensemble approach then aggregates the prediction of each base prediction models and results in a single prediction model for the unseen data. It allows to reduce the generalized prediction error. Since the above discussed base prediction models are diverse and independent, the prediction error of the final prediction model is decreased using the ensemble approach.

Output generation The final outcome of the ensemble predictor is estimated by combining the outcomes of all the base learners using a voting engine. The final predicted output is potentially more accurate because this ensemble model selects predicted output with least error as the final predicted output everytime. Also, the predictions of base learners in the ensemble model are updated to historical data for future evaluation of the base learners. This approach improves the accuracy of the resource utilization-based failure prediction. The performance and accuracy of each base predictor is evaluated and compared by applying the Root of Mean Squared Error score ($\mathbb{E}_{rmse}$) given in Eq. (12):

$$\mathbb{E}_{rmse} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(Z_{actual} - Z_{predicted})^2} \tag{12}$$

where m is the number of training samples, Z_{actual} and $Z_{predicted}$ are actual and predicted outputs, respectively. The predicted outcome with least fitness value is generated as final predicted output.

5 Failure tolerance

Failure Tolerance Unit (FTU) is portrayed in Fig. 4, where the failure-prone VMs $\{V_1^F, V_2^F,...,V_N^F\} \in \mathbb{V}^F$ (where N is number of failure-prone VMs) and their future resource requirements are arranged into *Resource Provisioning Matrix*.

The resource provisioning matrix is $N \times \mathbb{X}$ matrix composed of the predicted resource usage for $\mathbb{X}$ different resources ($\mathbb{R}_1$, $\mathbb{R}_2$,...,$\mathbb{R}_y$) of N VMs, where $y \in \{1, 2, \ldots, \mathbb{X}\}$. The maximum value out of each resource capacity is selected to decide the effective capacity of $\mathbb{X}$ resources in the *Selection-Box* (*S-Box*). The distinct selection boxes are estimated pertaining to predicted usage of $\mathbb{X}$ resources of all failure-prone VMs. Thereafter, the replicas of failure-prone VMs are assigned to the selected energy-efficient servers according to the resource capacities reserved by the S-Boxes for the respective VMs. For instance, the resource provisioning matrix in Fig. 4 reveals the requirement of $\mathbb{X}$ resources for each VM.

From resource provisioning matrix, the capacity requirement of R_1 resource are {21, 17,...,26} and it is assumed that 26 is the highest demand for the resource R_1. On the same lines, the set of values {43, 24,...,23} and {14, 18,...,41} shows the capacity requirement of resources R_2 and R_y, respectively, for failure prone VMs $\{V_1^F, V_2^F,...,V_N^F\}$. S-Box is composed of highest resource capacities associated to respective resource in resource provisioning matrix. The failure-prone VMs are safely allocated by mapping $\mathbb{X}$ number of S-Boxes to energy-efficient physical machines. The corresponding VMs allocation is performed by applying a Greedy approach such that the VMs are intentionally assigned to already active servers first while satisfying the resource capacity constraints so as to minimize the power consumption. The inactive physical machines are turned to active mode only if the currently active physical machines have insufficient resource capacity to fulfill the requirement of target VM.

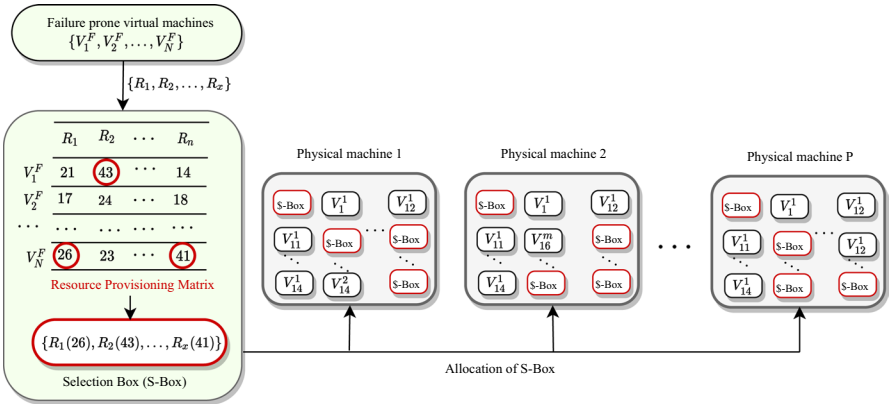


Fig. 4 Failure tolerance unit

6 Operational design and complexity

OFP-TM initializes list of VMs ($List_{\mathbb{V}}$) and list of servers ($List_{\mathbb{S}}$) as per the predefined configuration. The ensemble resource predictor is trained with previous samples of VM resource (viz., CPU and memory) usage. The predictor is trained and re-trained periodically for improved prediction accuracy of future resource usage estimation in real-time. The predicted resource usage outputs of each VM hosted on a server are aggregated to estimate the physical resource requirement of the respective server in next time-interval. This estimation helps to determine any over-/under-load condition on the respective server. The same procedure is followed for each server in the cluster. Accordingly, the VMs are categorized into failure-prone and non-failure-prone VMs.

Algorithm 1 presents the operational summary of OFP-TM.

Algorithm 1: OFP-TM operational summary ()

```
1  Initialize list of VMs List_V and Servers List_S;
2  for each time-interval {t_1, t_2} do
3      for each S_i : i ∈ [1, P] do
4          for each V_j hosted on S_i do
5              Predict multiple resource usage such as V_i^new(R) ⇐ Ensemble Resource
                 Predictor( ) where {R_1, R_2, ..., R_y } ∈ R ;
6              if V_i^new(R) > V_i^current(R) then
7                  V^Failureprone ⇐ V_i^new;
8              else
9                  V^Normal ⇐ V_i^new;
10             end
11         end
12         Aggregate predicted resource capacity of each resource of V_i^new;
13         Tot_c(R) ⇐ V_i^new(R) + Tot_c(R);
14         if S_i(R) > Tot_c(R) then
15             Update V^Failureprone by removing VMs currently placed on S_i ;
16         else
17             S_i will be overloaded;
18         end
19         Prepare resource provisioning matrix of size N × X which maps X resources
             of VMs to N number of VMs ;
20         Determine size of Selection − box_i ⇐ Max{(V_1^R_y, V_2^R_y, ..., V_N^R_y)},
             y ∈ {1, 2, ..., X} ;
21     end
22     Allocate S-boxes on energy-efficient servers satisfying the resource capacity
         constraint mentioned in Eq. (5);
23     Deploy the replicas/active images of failure prone VMs as per the allocation of
         their respective S-boxes;
24 end
```

Step 1 intitializes all the required parameters including $List_{\mathbb{S}}$ and $List_{\mathbb{V}}$ has complexity of O(P) and O(Q), respectively. Steps 2–24 iterate for $\{t: \{t_1, t_2\}\}$ time-intervals with O(t) complexity. Steps 3–21 and steps 4–11 repeat for P and $z = len(\omega_{ji})$) consume time-complexity of O(Pz). Step 5 calls prediction method where the complexity of ensemble predictor is assumed to be O(C^{EPM}). Steps 6–9 differentiate failure and non-failure-prone VMs have complexity of O(1). Steps 12 and 13 aggregate

Table 3 Server configuration

Server	PE	MIPS	RAM(GB)	Memory(GB)	$\mathcal{PW}_{max}$	$\mathcal{PW}_{min}/\mathcal{PW}_{idle}$
S_1	2	2660	4	160	135	93.7
S_2	4	3067	8	250	113	42.3
S_3	12	3067	16	500	222	58.4

Table 4 VM configuration

VM type	PE	MIPS	RAM(GB)	Memory(GB)
$_V small$	1	500	0.5	40
$_V medium$	2	1000	1	60
$_V large$	3	1500	2	80
$_V Xlarge$	4	2000	3	100

the resource usage information for a server have complexity of O(1). Steps 14–18 determine overload condition on a server have complexity of O(1). Steps 19 prepare a resource provisioning matrix of size $N \times \mathbb{X}$ shows a complexity of O(N$\mathbb{X}$) equivalent to O(Q$\mathbb{X}$) because $N << Q$. Step 20 computes the size of S-Box consumes complexity of O(1). Steps 22 and 23 deal with allocation of S-Box followed by assignment of VMs consume complexity of O(1). Hence, the total complexity is O(PQzt$C^{EPM}\mathbb{X}$).

7 Performance evaluation

7.1 Experimental setup

The simulation experiments are executed on a server machine assembled with two Intel® Xeon® Silver 4114 CPU with 40 core processor and 2.20 GHz clock speed. The computation machine is deployed with 64-bit Ubuntu 16.04 LTS, having main memory of 128 GB. The data center environment was set up with three different types of server and four types of VMs configuration shown in Tables 3 and 4 in Python version-3. The resource features like power consumption (P_{max}, P_{min}), MIPS, RAM and memory are taken from real server IBM [40] and Dell [41] configuration where S_1 is'ProLiantM110G5XEON3075', S_2 is 'IBMX3250Xeonx3480' and S_3 is 'IBM3550Xeonx5675'. The VMs configuration is inspired from the VM instances of Amazon website [42].

7.2 Dataset and simulation configuration

The performance of proposed work is evaluated using two realworld benchmark workloads including Google Cluster Data (GCD), which has resources CPU, memory, disk I/O request and usage information of 672,300 jobs executed on 12,500

servers for the period of 29 days [43]. The CPU and memory utilization percentage of VMs are obtained from the given CPU and memory usage percentage for each task in every five minute over period of twenty-four hours. The multiple resource utilization percentage of VMs are obtained from the CPU, network, and memory usage percentage for each job in every five minute over period of twenty-four hours. The experiments are executed with varying size of datacenter such as 200, 400, 600, 800, and 1000 VMs such that the ratio of VMs:servers is 2:1, where VMs can be allocated dynamically as per demand of users with online prediction interval of 'five minutes'. The number of users are not mentioned in the original dataset, therefore, we have created set of user equals to 60% of the number of VMs, requested varying number and type of VMs over time. Each user can hold VMs in the range between 0 and 10 with a constraint that at any instance, the total number of VM requests must not exceed total number of available VMs at the datacenter. In real computing environment, the cloud outages occur in cloud burst situations, massive failure of physical servers, and peak of hours every day causing overloads and resource contention. Accordingly, we consider a sudden peak of aggregated load (or resource demand) of all VMs hosted on a server, which is greater than available resource capacity of the respective server, as a cloud outage. Such outages are predicted periodically, in an online service environment.

7.3 Results

The $\mathbb{MTBF}$ and $\mathbb{MTTR}$ can be computed by applying Eqs. (13) and (14), respectively. Accordingly, the average availability can be calculated using Eq. (15), where nf is total number of failures, $\sum_{i=1}^{M} UT_i$ and $\sum_{i=1}^{M} DT_i$ represent total uptime and downtime, respectively, experienced by M users over time-interval $\{t_1, t_2\}$.

$$\mathbb{MTBF} = \int_{t_1}^{t_2} \left(\frac{\sum_{i=1}^{M} UT_i}{nf} \right) \mathrm{d}t \tag{13}$$

$$\mathbb{MTTR} = \int_{t_1}^{t_2} \left(\frac{\sum_{i=1}^{M} DT_i}{nf} \right) \mathrm{d}t \tag{14}$$

$$A_{avg} = \frac{\mathbb{MTBF}}{\mathbb{MTBF} + \mathbb{MTTR}} \tag{15}$$

Table 5 reports the performance metrics: $\mathbb{MTTR}$, $\mathbb{MTBF}$, average availability (A_{avg}), accuracy of failure prediction (Acc^P), number of predicted failures ($Fail^P$), and number of live VM migration ($Mig\#$) achieved for GCD workload for varying size of datacenter (200 VMs to 1000 VMs) over period of 400 minutes.

The prediction accuracy of multiple resources using ensemble predictor governs the performance of all other metrics. The average of failure prediction accuracy

Table 5 Performance metrics for GCD workloads

VM#	$T(min.)$	$\mathbb{MTTR}$	$\mathbb{MTBF}$	A^a_{avg}	Acc^{Pb}	$Fail^{Pc}$	$Mig\#^d$
200	100	1.47	2757.14	99.94	95.5	75	7
	200	1.68	2400.00	99.93	95.0	86	8
	300	1.47	2757.14	99.94	95.5	72	7
	400	1.26	3233.33	99.96	99.3	96	6
400	100	4.41	1804.76	99.76	94.8	119	21
	200	3.57	2252.94	99.84	95.8	395	17
	300	3.78	2122.22	99.83	95.5	262	18
	400	3.15	2566.67	99.88	96.3	140	15
600	100	4.83	2508.70.89	99.81	96.1	583	23
	200	3.99	3057.90	99.90	96.8	411	19
	300	3.99	3057.90	99.90	96.8	544	19
	400	5.88	2042.86	99.71	95.3	536	28
800	100	6.09	2658.62	99.25	96.38	733	29
	200	4.41	3709.52	99.88	97.38	725	21
	300	5.25	3100.00	99.83	96.88	696	25
	400	3.78	4344.44	99.91	97.75	713	18
1000	100	6.93	3233.33	99.79	96.7	906	33
	200	7.77	2602.70	99.70	96.3	607	37
	300	7.14	2841.18	99.75	96.6	999	34
	400	5.88	3471.43	99.83	97.2	972	28

[a]Av.avg.: Average Availability, [b]Acc^P:Failure prediction accuracy, [c]$Fail^P$: Number of predicted failures, [d]$Mig\#$: Number of VM migrations

varies from 86 to 99.3% for GCD. The $Fail^P$ and $Mig\#$ vary directly but non-uniformly depending upon the size of datacenter and errors in failure prediction.

Furthermore, to be observed that the obtained values of both $\mathbb{MTBF}$ and $\mathbb{MTTR}$ depend on the number of failures (nf) as depicted in Eqs. (13) and (14). The values of UT are obtained by computing product of *number of successfully deployed VMs* and *time-interval* over period $\{t_1, t_2\}$. The $\mathbb{MTTR}$ value associated to a VM is 0.21 minutes which is utilized from [44, 45]. Accordingly, the values of $\mathbb{MTTR}$ are computed for different number of VM migrations which varies with the number of unpredicted failures. The resultant availability values are computed by applying Eq. (15) and utilizing $\mathbb{MTBF}$ and $\mathbb{MTTR}$ values recorded during respective time-interval $\{t_1, t_2\}$.

7.4 Comparison

OFP-TM is compared to three state-of-the art works which predicts failure to provide high availability: Prediction-based Energy-aware Fault-tolerant Scheduling scheme (PEFS) [28], Hadoop Distributed Computing Clusters for Fault Prediction (HDCC) [32], and Cloud Disk Error Forecasting (CDEF) [33]. Further, the comparison

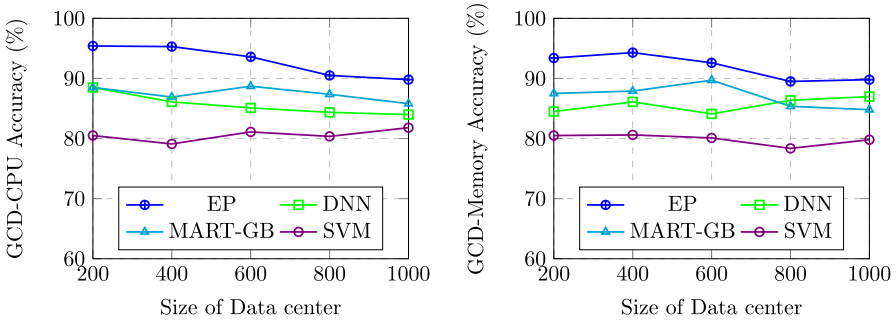
among OFP-TM, OFP-TM without S-Box, and Without OFP-TM is presented for the evaluation of various performance metrics. OFP-TM specifies proposed framework including VM Failure Ensemble prediction and failure-tolerance unit which is a combination of Resource Provision Matrix and Selection Box (S-Box) mechanism. The S-Box enables safer allocation of VMs alleviating maximum chances of failure. While OFP-TM without S-Box (where S-Box lies within Failure Tolerance Unit) specifies VM Failure Ensemble prediction and resource provisioning only. It means VMs failure is predicted but failure tolerance mechanism is not applied and VMs are provisioned as per the VM placement constraints mentioned in Eq. (5) in the manuscript. Without OFP-TM specifies neither failure prediction nor tolerance mechanisms are applied.

7.4.1 Failure prediction

Figure 5 compares the accuracy of failure prediction of ensemble predictor and the three comparative methods. On average, the prediction accuracy of ensemble approach varies from 93 to 97.8%, while the average accuracy of DNN [28], MART-GB [33], and SVM [32] ranges from 86 to 94%, 81 to 92%, and 76 to 84%, respectively.

7.4.2 Failure tolerance

Figure 6 compares the failure tolerance with respect to average number of VM migrations which is least for OFP-TM, consecutively followed by OFP-TM without S-Box (i.e. with ERP prediction) and without OFP-TM approaches. The achieved results depict that the number of VM migration increases non-uniformly with the size of the datacenter. The average number of VM migration using OFP-TM varies from 3 to 6% of the size of the datacenter. However, OFP-TM significantly reduces live VM migrations up to 72.6% and 83.3% over OFP-TM without S-Box and without OFP-TM, respectively for GCD workload. It can be observed from Fig. 6 that the values of average number of VM migration for data center of size 1000 VMs



(a) GCD-CPU **(b)** GCD-Memory

Fig. 5 Failure prediction accuracy for GCD workload

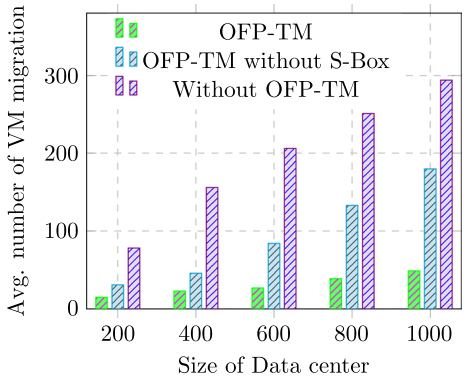


Fig. 6 Average number of VM migration for GCD workload

with respect to OFP-TM, OFP-TM without S-Box, and Without OFP-TM are 49, 180, and 294, respectively. By applying percentage improvement (which is a reduction of the number of VM migrations) formula using Eq. (16).

$$Reduction = \frac{(old\_value - new\_value)}{(old\_value)} \times 100 \tag{16}$$

In case of OFP-TM without S-Box, Reduction $= (180 - 49)/180 \times 100 = 72.6\%$. Similarly, in case of Without OFP-TM, Reduction $=(294 - 49)/294 \times 100 = 83.3\%$.

7.4.3 Availability

The availability varies with the values of $\mathbb{MTBF}$ and $\mathbb{MTTR}$, obtained during online processing over time-interval $\{t_1, t_2\}$. The variations observed (during experimental simulation) in the values of $\mathbb{MTBF}$ and $\mathbb{MTTR}$ are shown in Fig. (7). It is to be noticed that $\mathbb{MTTR}$ decreases when $\mathbb{MTBF}$ increases, which specifies a inverse relation between them.

The $\mathbb{MTBF}$ values decrease while the $\mathbb{MTTR}$ increases with growing size of the datacenter because of the slight decrease in the percentage of accuracy during prediction of failures (Table 5). The $\mathbb{MTBF}$ decreases and $\mathbb{MTTR}$ increases with following the trend: OFP-TM < OFP-TM without S-Box < OFP-TM without ERP. Figure 8 entails the availability for GCD where OFP-TM outperforms OFP-TM without S-Box and without OFP-TM by 17.2% and 33.5%, respectively, for GCD online workload distribution.

8 Conclusions and future work

A novel online failure prediction and tolerance model is proposed which inculcates high availability in VMs as well as servers by predicting any failure proactively and triggering necessary failure tolerance actions. An ensemble failure predictor is utilized to predict multiple resource usage of VMs concurrently and periodically to estimate their failure status. The operational status of physical machines is

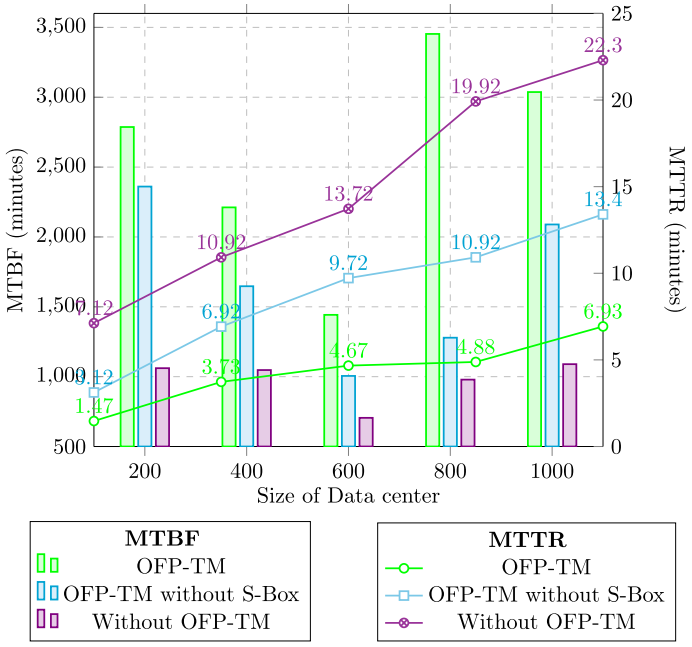
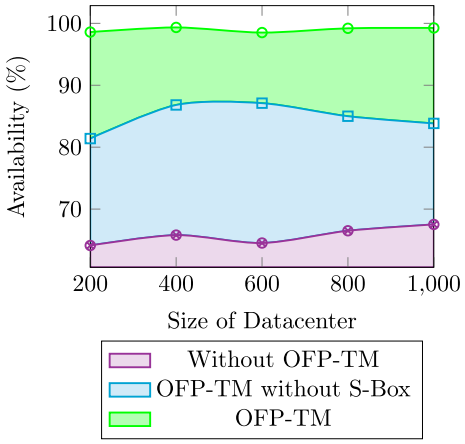


Fig. 7 MTTR and MTBF of GCD workload

Fig. 8 Average availability



monitored by analysing their resource usage and power consumption periodically to detect any failure in advance. The model employs a failure tolerance unit to decide an appropriate allocation of failure-prone VMs and migrate them to selected server using clustering-based S-Box. The performance evaluation of the proposed model reveals that it maximizes service availability and minimizes performance degradation due to overloads, cloud outages, SLA violations and excess power consumption. All the results are supported by the simulation and experiments executed on

benchmark real workload traces. The comparison with state-of-the-art approaches states that the OFP-TM can significantly improve the availability and optimize resource management cost of cloud datacenter.

Currently, the proposed model includes proactive failure tolerance only which can be extended in future with reactive failure tolerance strategies including N-Version programming, Parallel Execution etc. The combination of both reactive and proactive failure tolerance approach will build a more robust resource management model achieving high availability for cloud environments.

Acknowledgements This work is financially supported by National Institute of Technology Kurukshetra, Haryana, India.

References

1. Saxena D, Singh AK, Buyya R (2021) OP-MLB: an online VM prediction based multi-objective load balancing framework for resource management at cloud datacenter. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2021.3059096
2. Singh AK, Saxena D, Kumar J, Gupta V (2021) A quantum approach towards the adaptive prediction of cloud workloads. IEEE Trans Parallel Distrib Syst 32(12):2893–2905. https://doi.org/10.1109/TPDS.2021.3079341
3. Saxena S, Saxena D (2015) EWAS: an enriched workflow scheduling algorithm in cloud computing. In 2015 International Conference on Computing, Communication and Security (ICCCS), pages 1–5. IEEE
4. Saxena D, Singh AK (2021) Workload forecasting and resource management models based on machine learning for cloud computing environments. arXiv preprint arXiv:2106.15112
5. Saxena D, Singh AK (2021) OSC-MC: online secure communication model for cloud environment. IEEE Commun Lettvol. 25(9):2844–2848. https://doi.org/10.1109/LCOMM.2021.3086986
6. Singh AK, Saxena D (2021) A cryptography and machine learning based authentication for secure data-sharing in federated cloud services environment. J Appl Secur Res 1–24
7. Saxena D, Gupta R, Singh AK (2021) A survey and comparative study on multi-cloud architectures: emerging issues and challenges for cloud federation. arXiv preprint arXiv:2108.12831,
8. Kumar J, Saxena D, Singh AK et al (2020) Biphase adaptive learning-based neural network model for cloud datacenter workload forecasting. Soft Comput 24:14593–14610. https://doi.org/10.1007/s00500-020-04808-9
9. Li Z, Yang Y (2017) A novel network structure with power efficiency and high availability for data centers. IEEE Trans Parallel Distrib Syst 29(2):254–268
10. Saxena D, Chauhan RK, Kait R (2016) Dynamic fair priority optimization task scheduling algorithm in cloud computing: concepts and implementations. Int J Comput Netw Inf Secur 8(2):41
11. Saxena D, Vaisla KS, Rauthan MS (2018) Abstract model of trusted and secure middleware framework for multi-cloud environment. In: International Conference on Advanced Informatics for Computing Research, pages 469–479. Springer
12. Saxena D, Singh AK (2020) Security embedded dynamic resource allocation model for cloud data centre. Electron Lett 56(20):1062–1065
13. Saxena D, Gupta I, Kumar J, Singh AK, Wen X (2021) A secure and multiobjective virtual machine placement framework for cloud data center. IEEE Syst J. https://doi.org/10.1109/JSYST.2021.3092521
14. Gupta R, Saxena D, Singh AK (2021) Data security and privacy in cloud computing: concepts and emerging trends. arXiv preprint arXiv:2108.09508
15. Saxena D, Singh AK (2020) Auto-adaptive learning-based workload forecasting in dynamic cloud environment. Int J Comput Appl 1–11
16. Saxena D, Singh AK (2020) A proactive autoscaling and energy-efficient VM allocation framework using online multi-resource neural network for cloud data center. Neurocomputing 426:248–264

17. Saxena D, Saxena S (2015) Highly advanced cloudlet scheduling algorithm based on particle swarm optimization. In 2015 Eighth International Conference on Contemporary Computing (IC3), pages 111–116. IEEE
18. Saxena D, Singh AK (2021) Energy aware resource efficient-(eare) server consolidation framework for cloud datacenter. Advances in communication and computational technology. Springer, Singapore, pp 1455–1464
19. Zhang Q, Li S, Li Z, Xing Y, Yang Z, Dai Y (2015) Charm: a cost-efficient multi-cloud data hosting scheme with high availability. IEEE Trans Cloud Comput 3(3):372–386
20. CRN (2020) Ten biggest cloud outages of 2020. https://www.crn.com/slide-shows/cloud/the-10-biggest-cloud-outages-of-2020/11?itc=refresh
21. Endo PT, Gonçalves GE, Rosendo D, Gomes D, Santos GL, Moreira ALC, Kelner J, Sadok D, Mahloo M (2017) Highly available clouds: system modeling, evaluations, and open challenges. Research Advances in Cloud Computing. Springer, Singapore, pp 21–53
22. Jammal M, Kanso A, Heidari P, Shami A (2017) Evaluating high availability-aware deployments using stochastic petri net model and cloud scoring selection tool. IEEE Trans Serv Comput 14(1):141–154. https://doi.org/10.1109/TSC.2017.2781730
23. Mukwevho MA, Celik T (2018) Toward a smart cloud: a review of fault-tolerance methods in cloud systems. IEEE Trans Serv Comput
24. Endo PT, Rodrigues M, Gonçalves GE, Kelner J, Sadok DH, Curescu C (2016) High availability in clouds: systematic review and research challenges. J Cloud Comput 5(1):1–15
25. Gill SS, Buyya R (2018) Failure management for reliable cloud computing: a taxonomy, model, and future directions. Comput Sci Eng 22(3):52–63
26. Jhawar R, Piuri V, Santambrogio M (2012) Fault tolerance management in cloud computing: a system-level perspective. IEEE Syst J 7(2):288–297
27. Costa Carlos HA, Park Y, Rosenburg BS, Cher C-Y, Ryu KD (2014) A system software approach to proactive memory-error avoidance. In SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pages 707–718. IEEE
28. Marahatta A, Xin Q, Chi C, Zhang F, Liu Z (2020) Pefs: AI-driven prediction based energy-aware fault-tolerant scheduling scheme for cloud data center. IEEE Trans Sustain Comput. https://doi.org/10.1109/TSUSC.2020.3015559
29. Sharma Y, Si W, Sun D, Javadi B (2019) Failure-aware energy-efficient VM consolidation in cloud computing systems. Future Gener Comput Syst 94:620–633
30. Bui D-M, Lee S et al (2018) Early fault detection in IAAS cloud computing based on fuzzy logic and prediction technique. J Supercomput 74(11):5730–5745
31. Nguyen HM, Kalra G, Kim D (2019) Host load prediction in cloud computing using long short-term memory encoder-decoder. J Supercomput 75(11):7592–7605
32. Pinto J, Jain P, Kumar T (2016) Hadoop distributed computing clusters for fault prediction. In: 2016 International Computer Science and Engineering Conference (ICSEC), pages 1–6. IEEE
33. Xu Y, Sui K, Yao R, Zhang H, Lin Q, Dang Y, Li P, Jiang K, Zhang W, Lou J-G et al. (2018) Improving service availability of cloud systems by predicting disk error. In 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18), pages 481–494
34. Wang J, Bao W, Zhu X, Yang LT, Xiang Y (2014) Festal: fault-tolerant elastic scheduling algorithm for real-time tasks in virtualized clouds. IEEE Trans Comput 64(9):2545–2558
35. Zhu X, Wang J, Guo H, Zhu D, Yang LT, Liu L (2016) Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds. IEEE Trans Parallel Distrib Syst 27(12):3501–3517
36. Sivagami VM, Easwarakumar KS (2019) An improved dynamic fault tolerant management algorithm during VM migration in cloud data center. Future Gener Comput Syst 98:35–43
37. Vinay K, Kumar SM Dilip, Raghavendra S, Venugopal KR (2018) Cost and fault-tolerant aware resource management for scientific workflows using hybrid instances on clouds. Multimed Tools Appl 77(8):10171–10193
38. Ghoreyshi SM (2013) Energy-efficient resource management of cloud datacenters under fault tolerance constraints. In: 2013 International Green Computing Conference Proceedings, pages 1–6. IEEE
39. Chunlin L, YaPing W, Yi C, Youlong L (2019) Energy-efficient fault-tolerant replica management policy with deadline and budget constraints in edge-cloud environment. J Netw Comput Appl 143:152–166
40. IBM (1999) Power model. [online]. https://www.ibm.com/

41. Dell (1999) Power model. [online]. https://www.dell.com/systems/power/hardware/
42. Amazon (199) Amazon ec2 instances. [online]. https://aws.amazon.com/ec2/instance-types/
43. Charles R, John W, Hellerstein JL (2011) Google cluster-usage traces: format+ schema. Google Inc., White Paper, pp 1–14
44. Araujo J, Maciel P, Torquato M, Callou G, Andrade E (2014) Availability evaluation of digital library cloud services. In: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pages 666–671. IEEE
45. Santos GL, Endo PT, Gonçalves G, Rosendo D, Gomes D, Kelner J, Sadok D, Mahloo M (2017) Analyzing the it subsystem failure impact on availability of cloud services. In: 2017 IEEE Symposium on Computers and Communications (ISCC), pages 717–723. IEEE