EAI Endorsed Transactions
on Cloud Systems
Research Article EAI.EU

Modelling an Efficient Three-Tier Fault Tolerance Approach for Resource Provisioning over Cloud

Suvarna S. Pawar[1,*], Y. Prasanth[1]

[1]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation Vaddeswaram - 522502, Guntur District, Andhra Pradesh, India

Abstract

INTRODUCTION: Nowadays, the cloud computing paradigm encounters newer challenges in offering fault tolerance methods during service provisioning. The failures during service provisioning are unavoidable in the large-scale heterogeneous network. Therefore, the adoption of appropriate fault tolerance techniques can improve the provided service's efficiency and reliability.
OBJECTIVES: Thus, fault tolerating metrics give better accuracy to enhance the QoS, where the three-tier fault-tolerance approach is proposed to resolve the various failures in service provisioning.
METHODS: Initially, a Collector Model collects the request and ranks it based on the service to be provided. Secondly, the redundancy filter module is designed to filter out the request replication and avoid the unnecessary process to be carried out. Finally, the fairness resource allocation module is designed to perform the prominent request received from the users based on the available resources without service congestion.
RESULTS: This three-tier model operates concurrently to handle the multiple requests from the users of various connected nodes. The experimental analysis demonstrates that the three-tier fault tolerance model can enhance the cloud reliability over the large-scale heterogeneous network by ensuring QoS.
CONCLUSION: The well-realized fault tolerance approach can efficiently demonstrate the structural model and fault tolerance process over the computing environment, therefore enhancing the cloud extendibility. Moreover, the computing environment's failure is hugely complex, and failure has to be handled efficiently.

Keywords- Cloud computing, service provisioning, fault tolerance, three-tier model, replication avoidance.

Received on 04 July 2021, accepted on 31 August 2021, published on 14 October 2021

doi: 10.4108/eai.14-10-2021.171320

1. Introduction

Cloud serves as a hierarchical distributed collaboration and scalable paradigm, which is considered to project a superior service architecture merged with cost reduction process for providing the storage and computing resources [1]. Moreover, cloud reliability has become a massive obstacle that prevents the extensive spread of adoption over the computing paradigm. For instance, the computing engine of Google fails in March'15, which hinders the service accessibility for nearly about minutes [2]. Similarly, in the same month, Apple's cloud service is also failed where the millions of Apple users are hampered to buy digital music from the apple play store [3]. Additionally, various applications are carried out over the infrastructure where the system realizes the essential function and fulfils real-time cloud requirements [4]. With the vast amount of available research institutes and enterprises, the cloud's necessity makes them migrate their corresponding applications towards the cloud. The major challenge that relies on the cloud is: How the construct provides feasible, reliable, and real-time cloud applications with fault tolerance ability?

The author must consider allocating resources with a task scheduling nature to determine the task request proportion towards the virtual machine (VM) based on runtime context. A vast amount of techniques are

* Corresponding author: Email: sspawar.scoe@sinhgad.edu

modelled by the investigators to enhance the CC reliability. Sharafeddine et al. [5] elaborate a system architecture that gives fault tolerance ability to the CC paradigm. The fault tolerance and resource allocation process are anticipated to attain fault tolerance by performing massive copies of every task to subsequent computing nodes. Su et al. [6] anticipate a model-based cloud-driven approach to automatic and resource provisioning among the public and private cloud environment.

The provisioning of a well-analysed dynamical fault-tolerance approach for the real-time CC environment is considered a complex issue [7]. Initially, cloud applications are generally composed of a vast number of distributed nodes. It is incredibly complex to determine the general form of language modelling and adopts it to classify the fault tolerance and complex structural behaviour of cloud applications [8]. Subsequently, the cloud applications are accommodated to VM dynamically to fulfil diverse requests generated from the end-users.

The author discusses the number of user requests from various resources like storage, bandwidth allocation, and VM requirements in [9]. However, it is vital to maintain the competency to provide the fault-tolerance strategy dynamically to avoid redundant data with the most acceptable resource allocation [10]. This kind of model is adopted to manage the fault of passive and active tasks, which fulfils the real-time and feasible requirements of the cloud environment [11]. Finally, the fault tolerance mechanism over cloud applications includes various available resources, uncertainties, and concurrency of resource scheduling that triggers the complexity of model validation [12].

It is highly complex to demonstrate a cloud model that offers the model's completeness and soundness with fault-tolerance characteristics [13]. This investigation concentrates on fulfilling a research objective to provide a fault tolerance technique using the three-tier model and ensure the QoS functionality, characterized by concurrency and asynchrony, which is used to design the primary components towards computing paradigm [14]-[15]. The following are the research objectives:

1) Here, a novel fault-tolerance approach is proposed to compute the response time, time is taken for a response, task size, count, and host size. When a failure is encountered in VM, the proposed three-tier model (collector, redundancy filter, and resource allocation module) can dynamically manage the fault tolerance ability during task allocation over an active and a passive manner.

2) A three-tier model is proposed with a hierarchical manner to schedule fairness allocation based on users' requests, fault tolerance over VM failure, and redundant data filtering to avoid congestion over the CC environment. This model is used to integrate the primary components with fault tolerance ability and fulfil the QoS requirements.

3) The proposed three-tier model is used to validate and analyse the soundness of the fault-tolerance model. The proposed three-tier enforcement algorithm is provided to fulfil the need for cloud application and acquire superior reliability over the deadline (final time value). The simulation is carried out in a MATLAB environment to compute the functionality of the proposed model. The outcomes validate the provided objective and give a promising solution.

The remainder of the work is organized as Section 2 is background studies related to CC's fault tolerance approaches. Section 3 analyses the cloud model's three-tier architecture, which includes the collector module, redundancy filter module, and fairness resource allocation module. Section 4 is numerical results and discussion to project the soundness of the proposed three-tier module. Finally, section 5 is a conclusion with future research directions.

2. Related work

Marahatta et al. [16] discuss that when the cloud environment receives massive task instances from diverse applications, the functionality is initiated over various hosts. However, some hosts pretend to fail accidentally. It results in fault over the system. Various fault tolerance methods generally eliminate this process. An enormous factor leads to host failure, and certain faulty events and conditions generally trigger these failure events. The failures encountered in this method include crash over the operating system, hardware malfunctioning, software failures, network partitioning, short power outages, and so on.

Kushwah et al. [17] discuss various prevailing fault-tolerance approaches over data centres, including replication, task re-submission, retry, job migration, check-point, and so on. Specific investigations are carried out by Kushwah et al., which initiates some techniques over principle execution, migration, re-submission, software renovation, and replication to complement diverse fault-tolerance methods over data center-based task scheduling. Moreover, in various distributed and parallel computing approaches, the extensively utilized methods are used to replicate over multiple hosts.

Guo et al. [18] discuss a rearrangement-based enhanced fault tolerance scheduling approach to handle various dynamical scheduling process-based issues over cloud systems. The preliminary backup modelling is used to examine fault tolerance. The appropriate backup copy is validated after a specific performance evaluation to project that the resources are too occupied. Additionally, waiting tasks are re-arranged to adopt the realized sources. On the contrary, when the task is transmitted to the awaited queue of the VM where the sequence of execution is completely fixed, which cannot be modified. Soniya et al. [19] discuss overlapping backup approaches and the strategies for the virtual machine migration over the cloud environment to enhance the resource utilization process. It is preliminarily a backup approach, like the approach mentioned above stated by Guo et al. [18]. Yadav et al. [20] discuss a scheduling algorithm named as dynamically merged task scheduling process. This

model is the modified version of the breadth-first search process to predict the complete optimal functionality of VM for every task. The technique provided by Guo et al. [18] fails to reduce the response time and make span computation process. However, it is not easily merged with VM management approaches to diminish energy utilization and enhance fault tolerance.

The constant monitoring of resource characteristics during the execution of jobs with diverse probability-based and statistics-based approaches predicts the job failure rate can leverage the observations with resource failure for appropriate selection of fault-tolerance scheduling and resource allocation. It is based on various scheduling indicators for the generation of the decision scheduling process where the grid scheduler handles jobs' arrivals.

The scheduling process needs to select the available resources with least failure rate. The multi-constraint fault tolerance/load balancing process is anticipated to diminish the make span computation, task failure, and cost while enhancing resource utilization. The selections of resources are made with specific actual failure rates, the number of jobs provided, the processing ability of the accessible resources, and effective job execution. Duan et al. [21] discuss a novel scheduling process termed as Pre-Ant policies composed of various prediction models that rely on the scheduler and fractal mathematics to adopt enhanced ant colony algorithms. The model prediction demonstrates whether the execution is triggered based on the scheduler with load calibration. The scheduler is accountable for scheduling to reduce energy consumption over the fulfilled QoS. The integration of consolidation algorithm and energy-based optimal allocation strategy is modeled as the bin-packing problem with reduced power consumption model, as demonstrates by Ghribi et al. [22]. Nazir et al. [23] provide a novel adaptive fault tolerance-based job scheduling process. It is also considered as a check-point mechanism. The anticipated method is dynamically updated with a failure index that relies on resourceful completion of task allocation to preserve the failure index. The available resource brokers adopt fault index from scheduler to adopt diverse scheduling intensities during task arrival. The resources' fault and success index value are diminished when the task is finished within the given time limit.

Based on these analyses and the best of the knowledge retrieved with the extensive analysis, no existing works have been performed fault tolerance with multi-tier cloud model to guarantee fault-tolerance, elimination of data redundancy, and fairness based resource allocation [24] – [25]. This model intends to predict the system fault and provides the resistance towards the fault by avoiding non-essential data over the cloud environment. The resource allocation based on the received request is done with a fairness strategy to guarantee QoS.

3. Methodology

This section provides a better understanding of the fault tolerance mechanism with an appropriate system model. Here, the proposed three-tier model with three different modules (collector module, redundant data filter, and fairness resource allocation) is explained elaborately.

3.1. System model

The proposed three-tier model includes three modules: Initially, a Collector Model collects the request and ranks it based on the service to be provided. Secondly, the redundancy filter module is designed to filter out the request replication and avoid the unnecessary process to be carried out. Finally, the fairness resource allocation module is designed to perform the prominent request received from the users based on the available resources without service congestion. This three-tier model is adopted to predict the failure and the system's ability to handle failure. The analysis is done with an online available dataset QoS dataset with three different web services (Cloud Service 1-3). The values over the dataset are normalized using the Eq. (1):

$$Z_i = \frac{Z_i - Z_{min}}{Z_{max} - Z_{min}} \tag{1}$$

Here, the dataset values are verified for maximum and minimum values to set the web services over the cloud environment. The failure rate is predicted based on the actual request generated from the user. The error over the system due to failure is expressed as in Eq. (2):

$$error = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - Y_{i_{PT}}\right)^2} \tag{2}$$

Here, $Y_{i_{PT}}$ and Y_i are the predicted and the actual task failure identified over the cloud environment, $'n'$ is several tasks generated from the user.

3.1.1 Resource model for task allocation

In a CC environment, the service provider receives various independent tasks given by the users. The set of tasks are provided as $T = \{T_1, T_2, \dots, T_k\}$. These tasks are connected with various task requirements with a set of parameters required to execute certain Task T_k. It includes incoming tasks, deadlines, and system failure, respectively. The tasks are categorized as faulty when the system fails based on fairness proneness. The failed task is designed as $T = \{T_1, T_2, \dots, T_l\}$ for failure based scheduling process with the fault-tolerance mechanism. The cloud environment is composed of a host with virtual machines $(VM1, VM2, VM3)$ where the cloud model is used for creating virtualized resources to fulfill the end-users' requirements. The VM requirements are provided using $V_j = \{V_j^n\}$ where $'n'$ is the number of resources used for task allocation.

3.1.2 Fault tolerance mechanism

The task failure occurs due to resource unavailability, time exceeding the threshold limit, hardware failure, hardware failure, system that runs out of memory, improper library installation, resource over-utilization, and so on. Some of these faults are considered to be permanent/transient, which are independent of each other. Therefore, modelling of an efficient fault-tolerant mechanism has to fulfil the task deadline (response time) of all the system task that handles fault occurs under a worst-case environment. The three-tier model is extensive for fault tolerance that replicates tasks to multiple copies scheduled to various hosts. Thus, there is a possibility of enormous resource wastage with remarkable energy consumption. In this research work, the tasks are replicated (failure). Initially, the next tasks are considered from task (failure), and the tasks are replicated. The collector model then collects the tasks and maintains it as a bag of incoming tasks to the failed system from the end-users. The collector model is used to map the appropriate hosts, allocate resources, and schedule various hosts' tasks, respectively. The redundant copies of the task sequence are designed as depicted in Fig 1. Thus, the execution should not overlap various hosts to eliminate redundant data execution.

3.1.3 Task scheduling and fairness resource allocation

The process of scheduling is partitioned into two phases. Initially, the task is provided to the failure system, and the second phase is the creation of tasks from replicated data. The tasks from these regions are provided to the collector module. The tasks are collected from various systems and maintained over the bag of incoming features (BoT). The fairness resource allocation is given to BoT, which is maintained over the collector module. The scheduler needs to schedule the Task-based on the available resources with the proper response time. When the task completion period exceeds the given time limit and the response time is prolonged, it is observed that the system encounters an internal fault (any sort) and causes a delay in the task completion process. The system's failure has to be identified efficiently, where the process is explained in the algorithm given below. Algorithm 1 depicts the prediction of fault and performance tolerance based scheduling process.

Algorithm 1: Prediction of fault and tolerance scheduler

1. Process: Fault prediction and tolerance scheduler ()
2. Initialize task, historical dataset, set of VM, set of hosts
3. Failure prediction (task): prediction ()
4. if (prediction status = failed/faulty) then
5. Preserve the Task over BoT
6. Predict fault over system ()
7. else
8. Maintain Task over BoT
9. Non-fault task scheduling ()
10. end if
11. end procedure (Fault prediction and tolerance scheduler)

The failure tasks are placed over the task queue in an earlier deadline way. The consecutive tasks are considered from task queue (failure), and every task is replicated into multiple copies. The requested resources of the entire task from successive tasks are evaluated where the requested values are selected, i.e., resource vector is computed for all the tasks. Then, the task needs to be mapped towards available hosts with appropriate resources using Algorithm 1 and the fault prediction algorithm is given in Algorithm 2.

Algorithm 2: Fault prediction & scheduling

1. Process: fault prediction ()
2. Initialize the weight of BoT
3. for all dataset ($i = 1\ to\ n$)
4. for all $j = 1\ to\ n$;
5. Compute task scheduling with actual output;
6. Evaluate Error (MAE);
7. Use Mean Absolute Error to reduce the error;
8. Adopt it to the layers of collector and redundant failure model;
9. Repeat the process till the error is minimized;
10. end for
11. end for
12. end procedure

//Scheduling the Task

1. Procedure scheduling failed task ()
2. Set of failed task $\{T_1, T_2, \ldots, T_l\}$
3. Select successive tasks from failed task over queue, i.e., T_1, T_2, T_3;
4. perform reconstruction with filter module
5. Check VM list;
5. Sort task based on resource availability;
6. Check available host list;
7. if resources are scheduled to perform task, then
8. check the task scheduling order
9. if $task = not\ executed$, then
10. schedule task to available host
11. else
12. break the process
13. end if
14. else
15. break the process
16. end if
17. end procedure

3.1.4 Redundant filter module

The redundant filter model reduces the bottleneck while handling more filters. The task scheduling speed is reduced when the successive request is generated from when the constraints are encountered when a considerable number of requests appear from the primary source. The advantage of executing the redundant filter module is to provide an opportunity for paralleling running the task even in case of system failure by facilitating the load to be spread among the available resources (fairness allocation) and enhance the throughput, as shown in Fig 1. The redundancy filter provides a pipeline based

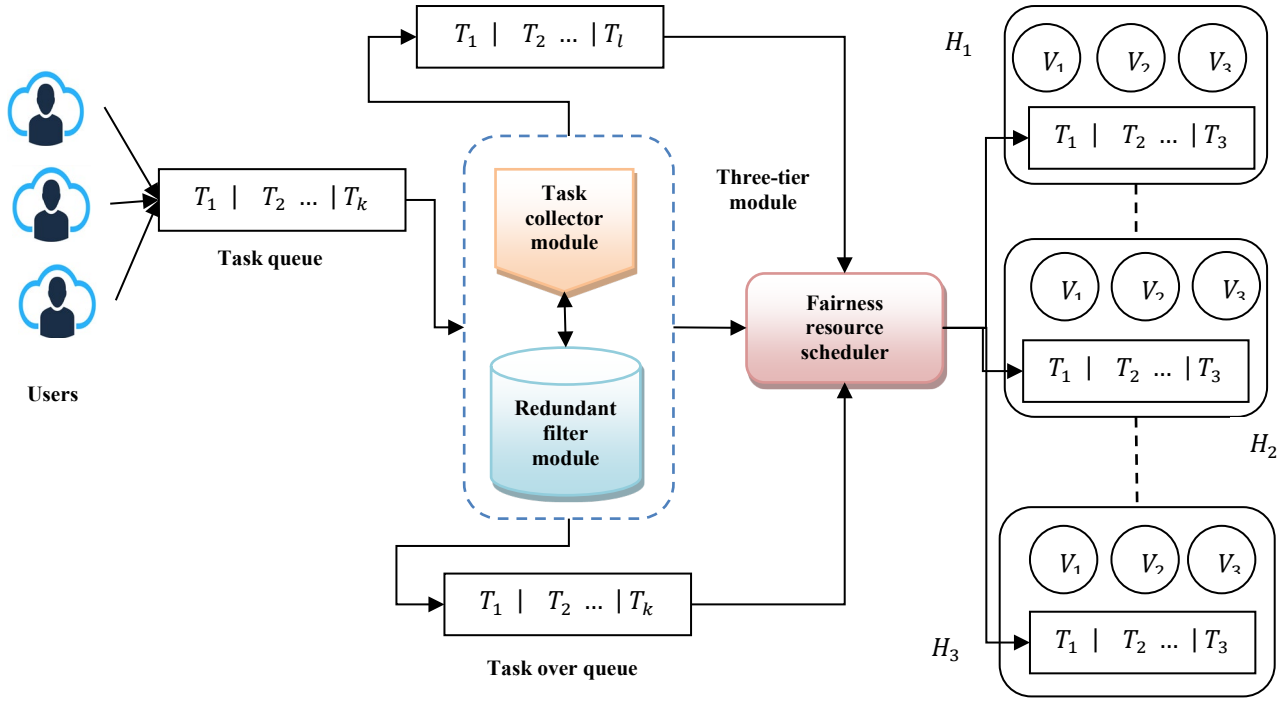

Fig 1 System model of three-tier module

manner without any redundancy of task allocation. The allocated task is given to the scheduler for appropriate scheduling with the delayed response time. The filtering is performed before it filters out the redundant task to complete the process. The primary benefit of this module is its resiliency.

When the filter encounters any failed task over the machine (unavailable), the queue's task has to be re-scheduled. The filter directly interferes with other instances of the components. The failure of the system does not affect the outcome of the failure of the redundancy filter. The process of the filter is more complicated as it is carried out in a distributed environment. The task that flows among these filters re no lost and preserved over the queue. When the filter fails to receive any task, the queue's task needs to be re-scheduled to another filter instance. Based on these analyses, it is observed that there is successful task allocation over the host environment. The filter runs in an isolated manner and provides an appropriate context to carry out the task allocation process.

This proposed three-tier module's primary objective is to attain better performance when the incoming tasks are prone to failure. The task that arrives over the system needs to join the queue without any system failure. The three-tier model's complexity is evaluated for the $'n'$ tasks and expressed in the $O(n^2)$. The relation among the failure system is based on the task and resources with correlation establishment. The correlation determines the association between the task and resources. The task failure is encountered when the scheduler gives minimal resources for processing. The tasks are failed only during the resource allocation is lower, and the request is higher. Based on the experimental observations, the three-tier model uses a fault tolerance mechanism to assign failure tasks to the most appropriate VM and physical host, as in Fig 1. Therefore, the proposed three-tier model works efficiently.

4. Numerical results and discussions

This research work chooses MATLAB 2016 environment, which industrialists and academia extensively utilize to perform experimentations. It is competitive to offer the cloud environment with the entire essential testing interface. Therefore, it is extensively suitable for algorithm verification. The essential parameters are discussed below: processing capabilities of the host are chosen randomly (1000-4000 MIPS (Million Instruction Per Second)), bandwidth (1-5 Gbps), VM processing capabilities (250-1000 Mbps). The performance of the proposed three-tier model is verified with parameters like response time, throughput

(Kbps), number of redundant data, time taken by all the end-users', Mean Absolute Error (MAE), final time value, cosine similarity value (CS), Pearson's correlation coefficient (PCC) and Karl Pearson's correlation coefficient (KPCC).

The proposed three-tier model's performance is tested with the variations in task count to improve the guarantee ratio. The three-tier model searches for a suitable task scheduling process and virtual machine where the existing approaches consume more resources. The neighbourhood scheduling mechanisms' guarantee ratios are lesser where these models consume excess tasks from the host with increased task count. Therefore, the tasks involved are finished after crossing the deadline setup. Some investigators adopt an overlapping mechanism for concurrent task processing. When the task count is higher, then the task scheduling process leads to a conflict where it reduces the guarantee ratio.

Similarly, with the increase in task size, the slack time is reduced drastically. When the primary sources are failed, then the backup copy does not efficiently perform the execution. With the increased task size, the guarantee ratio is also higher. The existing approaches do not adopt a resource utilization method that cannot give a time slot for execution. The process cannot perform the task before the deadline. When the task size increases with the failed system, then the backup copy consumed more resources. It cannot predict the more suitable VM for task scheduling. Thus, a new host is launched with the available VM. Moreover, VM and host's initiation needs a specific time to complete the task before the given deadline. The consequences of host count initialization are performed.

It is observed that the guarantee ratio of the proposed three-tier model is increased with the increase in host count. With the increase in host count, more VM is provided to execute the slack time. The guarantee ratio of the proposed three-tier model is higher than the other approaches. Based on this, the throughput is higher (kbps) with proper response time, elimination of redundant data. The similarity among the redundant data is analysed with cosine similarity, KPCC, and PCC. The error that occurred during the task performance is provided with the error rate analysis done with MAE. The failed system shows higher error compared to the non-failed system. The error rate is inversely proportional to resource allocation, i.e., when the system fails, it leads to error; however, in a non-failed system, the resource has to be allocated concurrently.

When there is an increase in host count, then there is some impact on task completion time. The parameter related to the task does not show any variations; thus, the time needed to complete the specific task is not reduced. The task completion over the three-tier model is higher than the existing approaches. The simulation parameters are given in Table 1.

Table 1. Simulation environment and parameter setup

Web services type	Cores	MIPS	RAM (MB)	Disk storage (GB)
CS1	1	500	1540	6
CS2	1	1000	3850	6
CS3	1	2000	885	6

The response time (seconds) of the proposed three-tier model is higher than other approaches. It is analysed using three different servers known as CS1, CS2, and CS3, respectively. Fig 2 depicts cloud servers' responses in three different colours, i.e., blue, dotted red, and dashed yellow. The X-axis represents the execution of MIPS tasks, while Y-axis depicts the response time (seconds). Here, the response time of CS1 is higher than in another model when there is no failure over the system design. Fig 3 depicts the throughput (kbps) where the x-axis shows real end-users and the y-axis shows throughput computation with kbps. The throughput of CS2 is higher when compared to CS1 and CS3. Here, CS1 shows nominal throughput values while CS3 gives lesser values during proper system functionality. However, when the system fault is injected; then the model shows some fluctuations in the output. Fig 4 depicts the computation of several redundant data. The efficiency of CS1 and CS3 is reduced while handing redundant data; similarly, in CS2, the efficiency is higher. Fig 5 depicts the total response time to the users while requests are generated from the end-users. Here, the response from three different servers is noted (CS1, CS2, and CS3), respectively. The total time taken by CS1 is higher than CS2 and CS3 during the system failure. Here, CS3 takes a lesser response time than CS1 and CS2 during the system failure (for maximum request generated from the users).

The failure in the cloud system leads to error generation to the output parameters. The error is measured here is Mean Absolute Error. The error rate of the response from the CS is analysed for 2000 users. The probability of error occurrence in provided CS (1, 2, and 3) is sensed from this observation. CS1 and CS2 give higher errors than CS3. Thus, the feasibility of CS3 is higher for handling the fault. However, it shows a lesser response time. The comparison of CS1, CS2, and CS3 is shown in Fig 6 – Fig 8. Here, the existing TL-CSP is compared with the proposed three-tier model.

Fig 7 depicts the computation of KRCC. The trust establishment and the success rate (guarantee ratio) of CS1, CS2, and CS3 are measured. The service provisioning rate of the web services is evaluated. The value ranges from 0-1 gives the feasibility of the correlation coefficient. When the value lies within this range, the service is said to be more efficient. When the value exceeds 1, then the service is not so proficient. The cosine similarity is measured and shown in Fig 10. The cosine similarity measures the similarity among the two vectors and determines whether the flow is done over the same direction. Based on this, the similarities among the services provided are analysed. The proposed model gives better similarity mapping during service provisioning to fulfil the QoS requirements. Fig 9 depicts

the computation of PCC. The relationships among the variables are measured with the ratio scale to establish the relationship strength with continuous variables. Based on these similarity measures, the service provided by CS3 is better and more reliable. Thus, Fig 6-8 depicts the selection of the cloud service model. CS3 is chosen to be a feasible model to tolerate the fault. However, the response time is lesser for CS3 while handling a massive request from the users. Fig 12 depicts the total time value taken by the cloud services to respond during the time of faulty condition. The functionality of these three services w.r.t fault tolerance and response time is inversely proportional to one another. The response time of CS3 is lesser than CS1 and CS2, respectively. Table 2 depicts the analysis of allocated resources based on the online available web service dataset.

Table 2. Allocated resource analysis

Average resources (failed tasks)	0.420	0.419	0.421
Average resources (successful task)	0.560	0.585	0.580
Pearson Correlation coefficient (PCC)	0.98	0.97	0.98
Cosine Similarity (CS)	0.97	0.97	0.97
Karl's Pearson coefficient for correlation (KRCC)	0.98	0.98	0.98

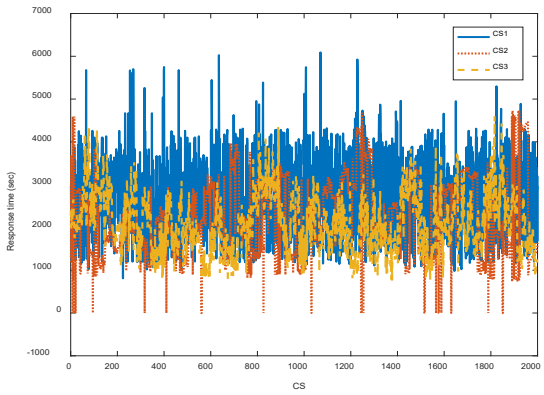

Fig. 2. Response time based on cloud services

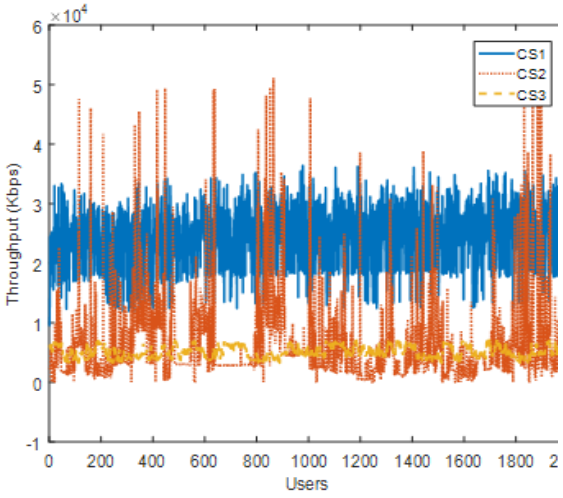

Fig .3. Throughput computation

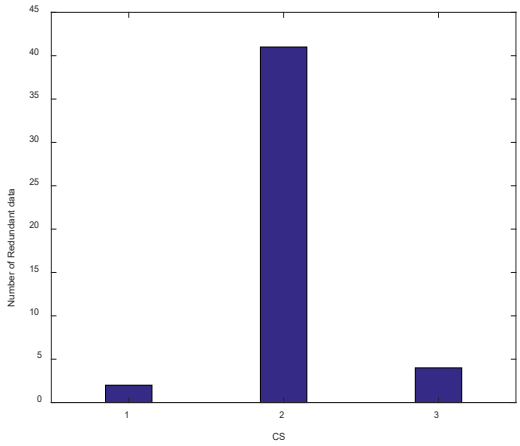

Fig. 4. Number of redundant data from incoming task

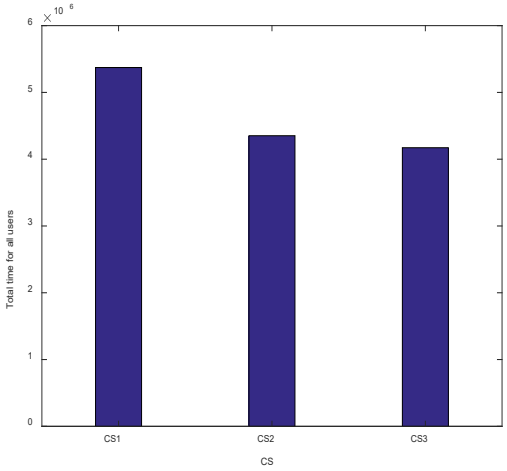

Fig .5. Total time needed for all user's

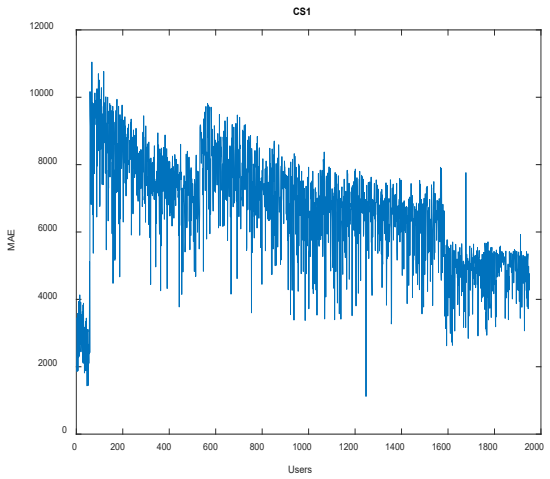

Fig .6. MAE computation of cloud service 1

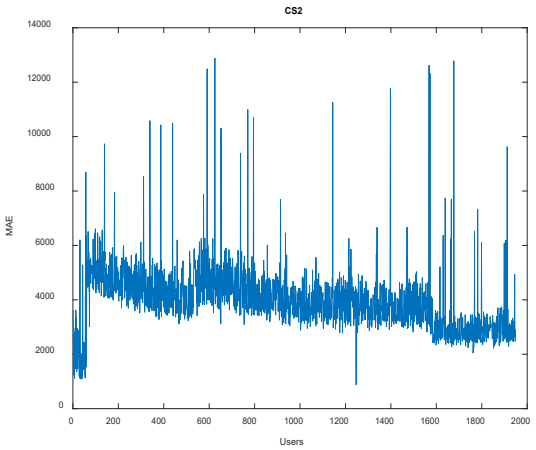

Fig. 7.MAE computation of cloud service 2

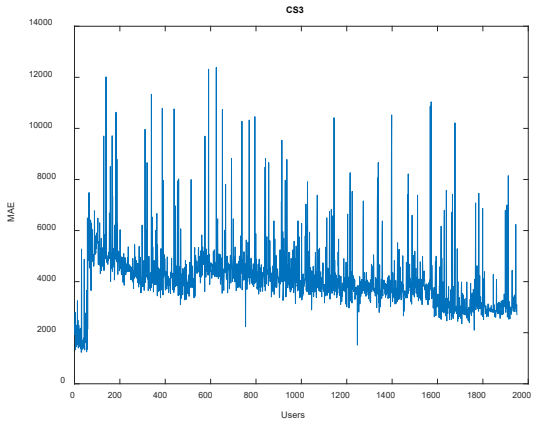

Fig .8. MAE computation of cloud service 3

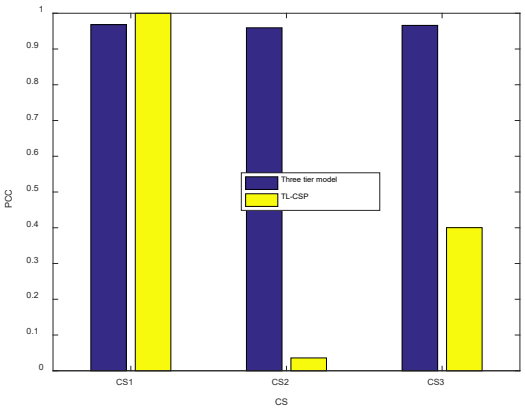

Fig .9. Pearson correlation coefficient computation

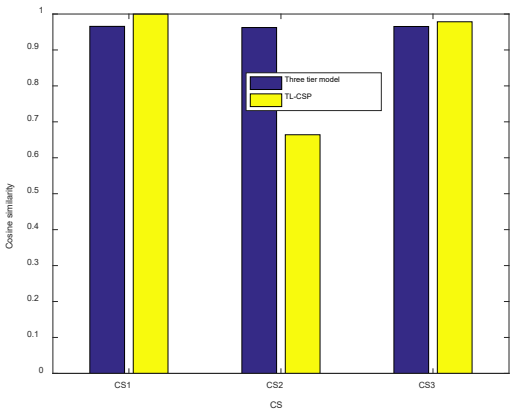

Fig.10. Cosine similarity computation

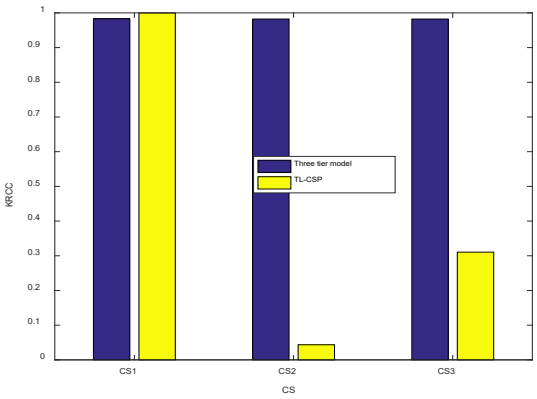

Fig .11. KRCC computation

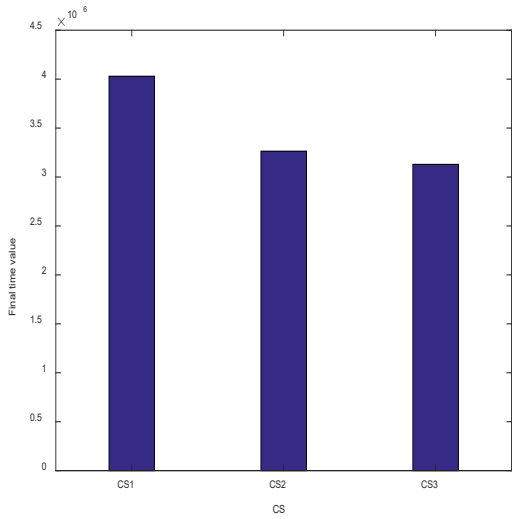

Fig .12. Total time taken by CS 1-3

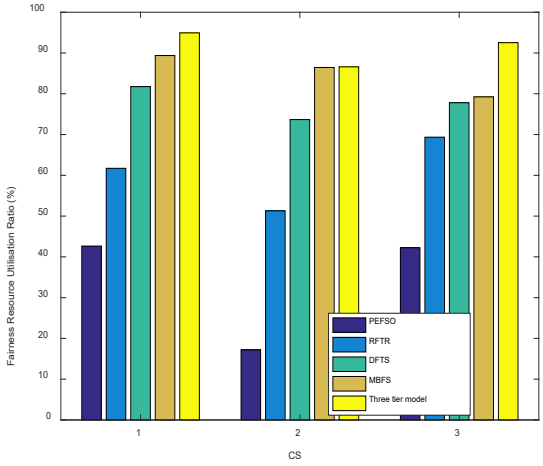

Fig .13. Comparison of Fairness resource utilization rate

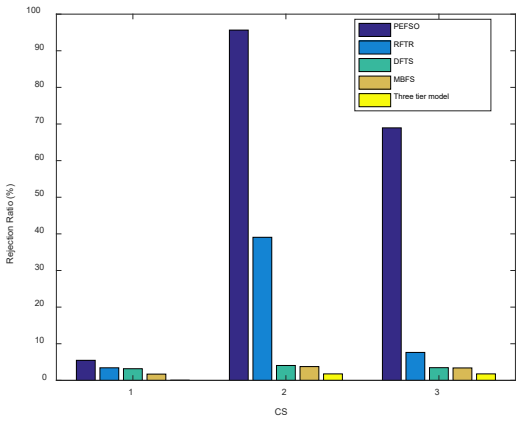

Fig.14. Comparison of rejection ratio (%)

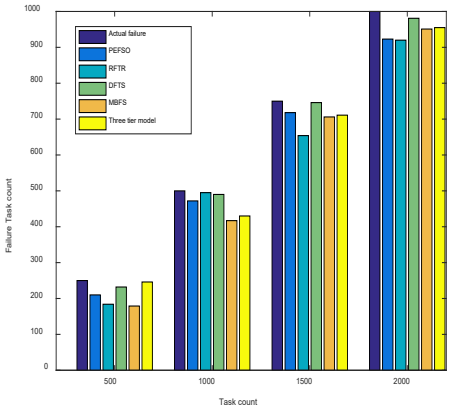

Fig. 15. Comparison of Failure task count

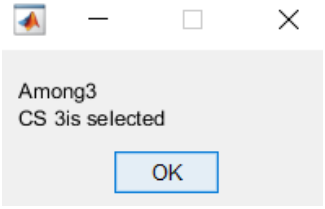

Fig .16. Selection of cloud service

Similarly, Fig 13 depicts the fairness resource utilization rate. It is depicted as the ratio among the time taken to execute specific tasks by total time. Fig 14 depicts the rejection rate of incoming tasks. Fig 15 shows that the task failure rate is analysed based on the tasks being failed due to no proper scheduling. The comparison of fairness resource utilization ratio (%), rejection ratio (%), and failure task count is done among three-tier model, Prediction-based Energy-aware fault tolerance scheduling (PEFS), real-time fault-tolerant scheduling approach with rearrangement (RFTR), dynamical fault-tolerant scheduling (DFTS), and Modified Breadth-First Search (MBFS). Fig 16 depicts the service selection of cloud with web service 3. Table 3 depicts the comparison of fairness resource utilization rate and the evaluation done among the proposed and existing approaches, and table 4 depicts the rejection ratio (%) evaluation. Table 5 depicts the evaluation of failure task count.

Table. 3. Comparison of Fairness resource utilization

Web Services	PEFSO	RFTR	DFTS	MBFS	Three-tier model
CS1	42.6480	61.7164	81.7597	89.3796	94.9354
CS2	17.1910	51.2942	73.6735	86.4453	86.6071
CS3	42.2390	69.3503	77.8024	79.2437	92.5351

Table .4. Comparison of rejection ratio (%)

Actual Failure	250	500	750	1000
PEFSO	210	472	718	923
RFTR	184	495	654	920
DFTS	232	490	746	981
MBFS	179	417	706	951
Three-tier model	246	430	711	955

From the above Tables, it is observed that the anticipated three-tier model outperforms existing PEFSO, RFTR, DFTS, and MBFS model. The fault tolerance ability of the three-tier model is higher than the other models. Simultaneously, the response time to the incoming task request is nominally lower for the three-tier model. However, it does not produce any conflicts over the fairness of resource allocation. Thus, the three-tier model pretends to fulfil the research objective efficiently.

6. Conclusion and Future Work

Cloud applications generally rely on a largescale environment composed of a vast number of distributed nodes. The complex and dynamical nature of the cloud environment loads to various conflicts in node failure very often. Some existing approaches do not concentrate on real time fault-tolerance, and some investigators concentrate on schedule to attain the optimization objective to enhance reliability and resource utilization efficiency. This research work concentrates on providing a hierarchical model with three tiers known as the collector model, redundant data filter model, and fairness resource allocation model. It examines the dynamical fault tolerance ability with appropriate VM deployment with task and response time based on a deadline. Also, it provides the essential components of cloud applications like users' requests, tasks, VM, and scheduling processes. Then, analysis and validation are performed with the dynamic characteristics of constructed three-tier model. The algorithm provided by this model realizes the tolerating ability of the system with the response time. Finally, the experimentation is carried out in a MATLAB simulation environment to show various applications' reliability.

The anticipated fault tolerance method and the existing approaches like PEFSO, RFTR, DFTS, and MBFS are evaluated to fulfil the system's reliability. The well-realized fault tolerance approach can efficiently demonstrate the structural model and fault tolerance process over the computing environment, therefore enhancing the cloud extendibility. Moreover, the computing environment's failure is hugely complex, and failure has to be handled efficiently. The process of fulfilling the QoS requirements

Table. 5. Comparison of Failure Task count

Web Services	PEFSO	RFTR	DFTS	MBFS	Three-tier model
CS1	5.4741	3.4345	3.1845	1.7077	0.0126
CS2	95.6448	39.0667	4.0640	3.7637	1.7687
CS3	68.9504	7.6284	3.4695	3.3955	1.7705

is exceptionally challenging. In the future, the anticipated three-tier model is extended to demonstrate the multi-objective constraint, which deals with the stability of transferring the fault tolerance to fault prevention in a stable manner. Similarly, the three-tier model's implementation is compared with the evaluation of various online available benchmark datasets based on QoS metrics like execution time, reliability, and resource utilization.

References

[1] Xia, J. Zhang, T. Q. S. Quek, S. Jin, and H. Zhu, “Energy-efficient task scheduling and resource allocation in downlink C-RAN,” in 2018 IEEE Wirel. Commun. Netw. Conf., Apr 2018, pp. 1–6.

[2] Liu, T. Han, N. Ansari, and G. Wu, “On designing energy-efficient heterogeneous cloud radio access networks,” IEEE Trans. Green Commun. Netw., pp. 1–13, 2018.

[3] Al-Dhabi, F. Paraiso, N. Djarallah, and P. Merle, “Elasticity in cloud computing: State of the art and research challenges,” IEEE Trans. Serv. Comput., vol. 11, no. 2, pp. 430–447, Mar 2018.

[4] Botta, W. de Donato, V. Persico, and A. Pescapé, “Integration of cloud computing and internet of things: A survey,” Futur. Gener. Comput. Syst., vol. 56, pp. 684–700, Mar 2016.

[5] Sharafeddine, K. Jahed, O. Farhat, and Z. Day, “Failure recovery in wireless content distribution networks with device-to-device cooperation,” Comput. Networks, vol. 128, pp. 108–122, Dec 2017.

[6] Su, Q. Xu, J. Luo, H. Pu, Y. Peng, and R. Lu, “A secure content caching scheme for disaster backup in fog computing enabled mobile social networks,” IEEE Trans. Ind. Informatics, pp. 1–11, 2018.

[7] Chintala, R.R., Narasinga Rao, M.R., Venkateswarlu, S. “Performance metrics and energy evaluation of a lightweight block cipher in human sensor networks”, International Journal of Advanced Trends in Computer Science and Engineering, vol. 8 , no. 4 ,pp. 1487-1490, 2019.

[8] Sambasiva Rao, K., Kameswara Rao, M. "A lightweight digital signature generation mechanism for authentication of IoT devices ", International Journal of Recent Technology and Engineering, vol. 7, no. 6, pp. 1862-1866, 2019.

[9] Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, ``Fuzzy keyword search over encrypted data in cloud computing," in Proc. IEEE INFO- COM, San Diego, CA, USA, Mar. 2010, pp. 1_5.

[10] Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos, "Security and privacy for storage and computation in cloud computing," Information Sciences, vol. 258, pp. 371–386, 2014.

[11] Kadham, N.R., Sreenivasa Ravi, K. "A lightweight One Time Password (OTP) based smart learning in Internet of Things", 2018 International Journal of Engineering and Technology(UAE), vol. 7 pp.480-483, 2018.

[12] Alsaghier, H.M., Shakeel Ahamad, S., Udgata, S.K., Reddy, L.S.S. "A secure and lightweight protocol for mobile DRM based on DRM community cloud (DCC)", Advances in Intelligent Systems and Computing, vol. 515, pp.475-483, 2018.

[13] Chen W, Lee Y C, Fekete A, et al. Adaptive multiple-workflow scheduling with task rearrangement [J]. The Journal of Supercomputing, 2015, 71(4): 1297-1317.

[14] Jing W, Liu Y. Multiple DAGs reliability model and fault-tolerant scheduling algorithm in cloud computing system[J]. Computer Modeling and New Technologies, 2014, 18(8): 22-30

[15] Patra PK, Singh H, Singh R, et al. Replication and Re-submission Based Adaptive Decision for Fault Tolerance in Real-Time Cloud Computing: A New Approach[J]. International Journal of Service Science, Management, Engineering, and Technology, 2016, 7(2): 46-60.

[16] Swathi, D.R., Srikanth, V. "Hybrid trust management in cloud computing", International Journal of Advanced Science and Technology, vol. 29, no. 3, pp. 8394-8401,2020

[17] Potluri, S., Rao, K.S. "Simulation of QoS-Based Task Scheduling Policy for Dependent and Independent Tasks in a Cloud Environment", Smart Innovation, Systems and Technologies, vol. 159, pp.515-525,2020

[18] Sirisha, N., Kiran, K.V.D., "An efficient and lightweight security scheme for big data", International Journal on Emerging Technologies vol. 11 , no. 1, pp. 414-420, 2020.

[19] Soniya, J. Angela, J. Sujana, and T. Revathi, "Dynamic fault-tolerant scheduling mechanism for real time tasks in cloud computing," International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016.

[20] Gali, S., Nidumolu, V. "Multi-context trust aware routing for internet of things", International Journal of Intelligent Engineering and Systems, vol. 12, no. 1, pp. 189-200,2019.

[21] Duan, C. Chen, G. Min, and Y. Wu, "Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems," Future Generation Computer Systems, vol. 74, pp. 142–150, 2017.

[22] Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient VM scheduling for cloud data centers: Exact allocation and migration algorithms," 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, 2013.

[23] Nazir, K. Qureshi, and P. Manuel, "Adaptive check pointing strategy to tolerate faults in economy based grid," Journal of Supercomputing, vol. 50, no. 1, pp. 1–18, 2009.

[24] P. Sherubha, "Semi-supervised Learning approach for detecting abnormalities in cloud computing," "International Virtual Conference on Smart Advanced Material Science & Engineering Applications," 2020.

[25] P. Sherubha, "Graph Based Event Measurement for Analyzing Distributed Anomalies in Sensor Networks," Sådhanå (2020) 45:212, https://doi.org/10.1007/s12046-020-01451-w

[26] Dhote, B.L., Krishna Mohan, G. "Trust and Security to Shared Data in Cloud Computing: Open Issues, Advances in Intelligent Systems and Computing", vol. 870, pp. 117-126, 2019

[27] Patil Abhijit, J., Syam Prasad, G.,"Trust based security model for IoT and fog based applications, International Journal of Engineering and Technology(UAE)", vol. 7, pp. 691-695,2018

[28] Nashnosh, M.T., Vijaya Babu, B.," LESIPT: Lightweight enhanced secure incentive protocol with trusted value for multi-hop wireless networks" ,International Journal of Applied Engineering Research vol. 9, no. 19, pp. 5373- 5384, 2014

[29] Kameswara Rao, M., Usha Switha, T., Naveen, S. "A novel graphical password authentication mechanism for cloud services", Advances in Intelligent Systems and Computing, vol. 433, pp. 447-453,2016.

[30] https://sourceforge.net/projects/qosmonitoring/files/webservice1.txt/download