Multilevel Fault-Tolerance Aware Scheduling Technique in Cloud Environment

Devi K.[1], Paulraj D.[2]
[1] Department of CSE, SRM Valliammai Engineering College, India
[2] Department of CSE, RMD Engineering College, India
devii.jeya@gmail.com, kingrajpaul@gmail.com

Abstract

In cloud computing, the resources are delivered to the users on demand at a considerable cost. Due to low maintenance and high scalability services, enterprises wish to deploy their newly developed application towards the computing environment. For large scale applications, fault tolerance is an essential task that guarantees the reliability and availability of computing services. In this paper, a multi-level fault tolerance scheduling mechanism is proposed that overcomes the real-time failure in the system. In the first phase, non-functional testing and decision making algorithm is used to find the reliability of virtual machines. Here, the reliability criterion is achieved by Reliable Decision K-Nearest Neighbor (RDK-NN) algorithm that considers only the best reliable virtual machine. In the second phase, high availability is achieved using a scheduling algorithm. For this purpose, a Teaching-Learning Based Optimization (TLBO) scheduling is proposed that provides a better-scheduled set of tasks for the corresponding users. The evaluation of the proposed approach is carried out under Cloudsim platform. The performance is determined in terms of makespan time, failure ratio, performance improvement rate, response time and rejection ratio to estimate the scheduling task. The result shows that the system achieves high reliability and availability of data with a multi-level format in the cloud environment.

Keywords: Cloud computing, Makespan, Task assignment, Task scheduling, Fault tolerance

1 Introduction

In the modern computing technology, cloud computing offers accessing, manipulating, and configuring the resources to the customers via the internet with pay-as-you-go basis. It is an on-demand system that provides users to utilize the computing resources through service providers such as Amazon, Google, Microsoft, and Apple on a pay–per use basis. Based on the services delivered, the cloud environment is classified as follows: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) & Software as a Service (SaaS) [1]. The IaaS infrastructure deals with resources such as operating system, networking equipment, storage, and processors. It provides the user with the cloud provider. The SaaS is an on-demand software application, more popular among Corresponding consumers which provides the user with the access of software and application such as Google docs, Email cloud, etc. over the internet. The PaaS in the cloud-like Google App Engine manages the software application over the web that allows the user to create applications. This service enables the user to avoid the complexity of managing and buying the licenses, network, development tools, and other resources [2]. The concept of virtualization technology in the cloud enables the user to utilize the computing services with the help of virtual machines (VM) and lease it to the enterprises or an individual user [3]. As many VM are employed, the task scheduling strategy is adapted to allocate the task to the resources in the cloud [4]. Even though Cloud Computing is a general trend in all industries, some failures need to be addressed. Some of the main issues are to ensure robustness, reliability and availability of important services in the cloud system. On the other hand, failures in the cloud degrade the performances, which should be managed using the fault-tolerance technique [5].

Fault-tolerance deals with the ability of a cloud scheduler to protect the delivery of tasks and to operate continuously, even in the case of failure [6-7]. The problems should be identified by cloud fault-tolerance components and should be resolved within the shortest period [8]. The fault may arise due to hardware failures, virtual machine malfunctioning, network congestion, and application failure [9]. Fault Management in a cloud computing environment depends on two major parameters. (i) Recovery point objective: It defines the volume of data lost during a fault. (ii) Recovery Time Objective: It defines the amount of time that takes to repair the fault when it occurs. Cloud resources are known to experience inconsistency in their performance delivery [10]. Fault tolerance is one of the important issues that detect and locate the faulty nodes

*Corresponding Author: Devi K.; E-mail: devii.jeya@gmail.com
DOI: 10.3966/160792642021012201011

based on the faulty diagnosis protocol [11]. Fault detection in the virtualized system uses the I/O architecture that provides a greater benefit for the reliability and reusability features. Based on the I/O architecture model, the faults on the VM is detected and recovers by switching to another one and continue to operate without data loss [12]. The resources are dynamically provisioned and delivered to users in a transparent manner automatically on-demand. As more number of Virtual Machines are employed in the cloud, the process of assigning tasks to the cloud resources becomes a difficult task. Thus, a task scheduling strategy is applied to map the tasks and dispatch them to the resources in the variant environment efficiently. Due to this, finding an optimal solution in scheduling the task is considered as an NP-complete problem. In the cloud system, several heuristic algorithms such as Min-max, Max-min, and Heterogeneous Earliest Finish Time (HEFT) are available [13]. Some of the heuristic algorithms are also used to find the optimal solutions to the complex problems in cloud systems that include the Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), League Championship Algorithm (LCA) and Deep learning Algorithm [14-17]. These algorithms minimize the search space and the execution is carried out at a certain time limit. The scheduling task is based on several strategies such as time, cost, fault tolerance, and quality of service (QoS) [18-19]. The scheduling task in the cloud environment should meet the QoS requirements including the deadline and the makespan [20-21]. The fault tolerance based scheduling is an efficient approach for the real-time task scheduling on the computing instances. The Primary-Backup (PB) model uses multiple copies namely primary and backup to execute the task in two different processing units for fault tolerance.

Due to the cloud's dynamicity and large scale characteristics, it has been deployed in various fields. Cloud reliability and availability are considered as a major problem [22-23]. In this research work, a Teaching-Learning-Based Optimization (TLBO) algorithm is proposed for task scheduling in cloud computing among the user and the cloud service provider. The proposed scheduling technique for fault tolerance awareness addresses the cloud task execution which would reflect on the currently available resources and reduce the early failure of autonomous tasks. It indicates that the technique is very appropriate for the task execution to ensure high reliability and availability in the cloud computing environment.

The contributions of the work are as follows.

- Fault aware scheduling ensures reliability, robustness, and availability for essential services as well as running of applications in the cloud computing system.
- The RDK-NN decision-making algorithm helps to obtain a reliable VM which utilizes the available cloudlets, resources and also reduces the premature failure of the system.
- The proposed TLBO fault aware scheduling ensures the scheduling of tasks based on fitness level within the minimum execution time.
- Reduces fault tolerance overhead and allocate virtual resources effectively that meet the requirements of both the users and cloud providers.

The rest of the paper is modeled as follows: Section 2 discusses the related works of the fault aware scheduling techniques. The detailed explanation of the proposed multi-level fault aware scheduling model is provided in section 3. In section 4, the performance evaluation and comparison of existing and proposed scheduling methods are discussed. Finally, in section 5, the paper is concluded based on the analyzed results.

2 Related Works

Abdulhamid et al. [24] proposed a Dynamic Clustering League Championship Algorithm (DCLCA) that addresses the fault tolerance scheduling issues in the cloud computing environment. In League Championship Algorithm (LCA) fault detection is carried out to detect the failure at the operating system, virtual machine (VM), and application level. Based on this, the job is reassigned from the insufficient resources to the ideal queue resources. Here, task clustering is performed to categorize the most exceptional task cluster and to select the best VM by the cloud information system (CIS). Finally, scheduling is carried out based on the current CIS information that partitions the task according to the available resources.

Abdulhamid and Latiff [25] proposed a fault aware scheduling scheme named Check Pointed League Championship Algorithm (CPLCA) to handle the unexpected task execution failure. In this task, the failed job is transferred to the available VM, and the execution is carried from the last current state using the checkpointing strategy. It reduces the redundant execution breakdown during the task event and the VM failure.

Zhou et al. [26] discussed the makespan-aware optimum scheduling scheme with a two-stage heuristic method that determines the assignment and replication task and then estimates the schedule of the assigned task. In the first stage, clustering is performed for makespan minimization that enables the replication of assigned tasks to satisfy the reliability requirements.

Almezeini and Hafez [27] proposed a task scheduling algorithm in cloud based on the Lion Optimization Algorithm (LOA). It is a nature-inspired metaheuristic algorithm that mimics the hunting behavior of the ant lions to decrease the execution time of the task. The lions in the residents which are considered as male remain in position whereas the

female has their strategies to search the optimal solution. This algorithm determines the best solution based on their fitness value (makespan) obtained from each lion.

Xu et al. [28] proposed Min-min based time and cost tradeoff (MTCT) to minimize the overall completion time (makespan) and execution cost in the cloud computing environment. It determines the workflow scheduling issues in cloud and adopted a fault recovery technique to improve the reliability and fault tolerance techniques. In this, the given application is composed of a set of tasks scheduled to the proper resources with the consideration of fault recovery in the cloud.

Zhang and Zhou. [29] describe the two-stage strategy to enhance the quality of service, makespan, and scheduling performance in the clouds. In the first stage, the task is classified, and VM is created based on the historical task scheduling data. At the second stage, matching is performed between the task and the suitable VM with different resource attributes. It saves the waiting time for scheduling the task to utilize the VM by the user.

Marahatta et al. [30] proposed energy-aware fault-tolerant dynamic scheduling scheme (EFDTS) for task classification. It was developed to divide the immediate tasks into distinct classes and then allocate them to the most suitable virtual machines based on their classes. This will lead to the reduction in response time while considering energy consumption. Replication was used for the fault tolerance to minimize the task rejection ratio caused by machine failure and delay. Furthermore, a migration policy was developed that can simultaneously improve energy efficiency, but no complementary features were considered for resource utilization.

Hsieh et al. [31] proposed a Feature-oriented Fault Diagnosis Agreement (FFDA) protocol with the exchange of three round of message that detect the fault processors in failure mode. This model consists of three stages which include message exchange, fault diagnosis, and decision making phase. Initially, at the message exchange phase, the messages are collected to determine and eliminate the faulty processors for the next stage. Then apply dormant diagnosis rule for each round of message exchange and malicious faulty processor rule to find the faulty processors. Finally, the decision value is obtained in the decision making phase. However, the time consumption is high in the data exchange process.

From the above analysis of the fault tolerant scheduling schemes, the existing schemes perform the execution of task in the presence of faults. Fault tolerance is executed in many earlier works on the basis of primary backup technique of scheduling the task on two different processors. Thus the task are able to schedule before its deadline but it suffers from high processing time. Also, at the time of processing, node failures may occur very frequently and thus the replication technique is utilized to achieve fault tolerance which increases the energy consumption to complete the same set of tasks. Further, the proactive fault tolerance mechanism migrates a task from unhealthy node to the healthy node without stopping of node migration. However, the overall overhead and the sudden failure prolong the execution time due to the failure in the system. The faulty recovery schemes mostly rely on the reactive scheme of checkpoint mechanism where the possible failures are predicted without considering scheduling issues. The task execution failure is no longer accidental but it is a common characteristic of the cloud computing environment. However, these affect the availability and reliability in the cloud environment. Hence, an active fault aware scheduling technique should be adapted to utilize the cloud resources efficiently. To overcome these limitations, a multilevel fault aware scheduling mechanism is proposed to schedule the task to ensure the high reliability and availability in the cloud system.

3 Proposed Multi-Level Fault Tolerance Mechanism

The proposed fault tolerance aware multi-level scheduling technique is processed in two phases. In the first phase, the allocation of a task to the virtual machine is carried out, and non-functional testing is performed to determine the efficient VM. In the second phase, fault tolerance aware scheduling is achieved through the TLBO technique execute the group of an independent task to a suitable VM.

Figure 1 shows the architecture of the proposed multi-level fault aware scheduling algorithm. Initially, the task is assigned from the N- number of users to the service provider. Then, the fault detection is performed with five levels of testing, and the RDK-NN decision mechanism is used to obtain reliable VMs. The TLBO scheduling schedules the task based on their fitness measure to the required host.

3.1 VM Fault Detection

In the cloud task scheduling, the task is assigned to the independent virtual machine that undergoes non-functional testing and decision phase to determine the best VM. It helps to obtain reliable virtual machines in the cloud environment and access the process of the client request. Initially, *n* number of tasks obtained from the cloud users are fed into multiple virtual machines for processing. After processing, the non-functional testing is carried out to obtain a reliable VM. The testing modules are as follows.

3.1.1 Non-Functional Testing

Cloud Testing refers to the validation and verification of infrastructure, environment, and application which

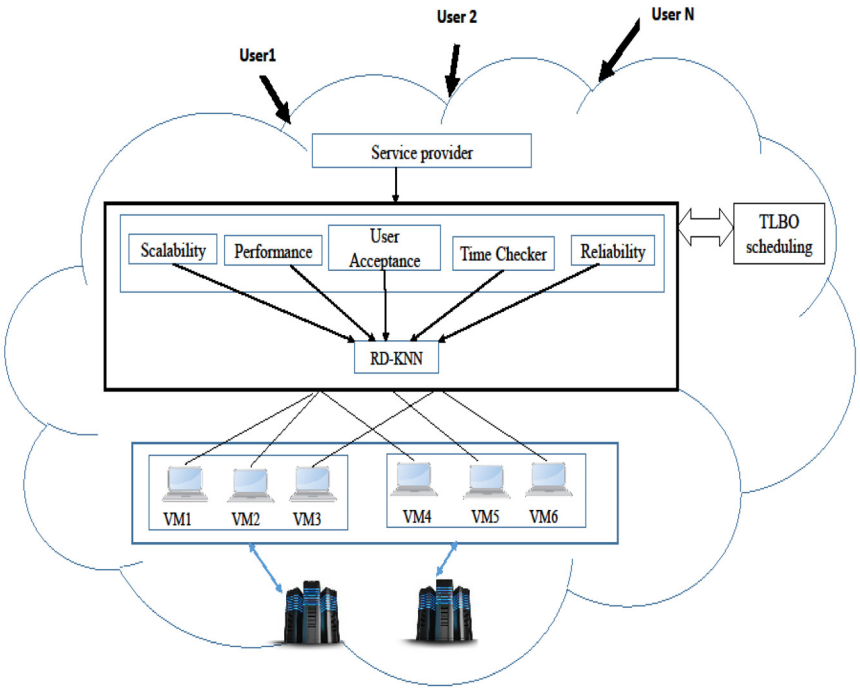

Figure 1. The Architecture of Multi-level Fault Tolerance Using Scheduling

are available on-demand by conforming them to the business model expectation of cloud computing. The testing modules are as follows.

Performance Testing: The performance testing module checks the responding ability of the VM executed under a diverse workload. This type of testing helps to obtain a reliable VM for satisfying the business requirements. The formula calculates the testing:

$$performance\ testing = \frac{Time\ spent\ for\ the\ task}{total\ available\ time} * 100 \quad \textbf{(1)}$$

From equation (1), performance testing is estimated based on the ratio between time spend for the task to the total time available for testing. Here, the total available time refers to the time utilized by the system to test and monitor the VM at the predefined conditions in a varying workloads.

Scalability Testing: The scalability module determines the ability of the VM to expand or increase the processing capacity on–demand. This testing ensures that the given task can handle the user requirements on the varying demand and the on-growing capabilities of the system. It is calculated by,

$$Scalability = \frac{Average\ allocation\ of\ VM\ speed}{Execution\ time} \quad \textbf{(2)}$$

Time Checker: It defines the response time of each VM and is estimated in milliseconds. In this, we set the time limit as 5000ms. The virtual machine which respond at this specified time limit is regarded as a reliable VM.

Acceptance Test (AT): This module checks whether the cloud resource produces a logical result or not even during the failure. The failed VM is not considered for the next process, but the corrupted and the succeeded VM will be considered for further processing. It is estimated as follows.

$$AT = \frac{remaining\ task - task\ completed}{total\ number\ of\ task} \quad \textbf{(3)}$$

The acceptance test determines whether the system satisfies the acceptance criteria based on the score obtained from equation (3). In this, if a total of 100 tasks is taken, then the completed task and the task remaining to complete are estimated and their score value is predicted to determine whether the entity accepts the system.

Reliability Testing: It checks the reliability of each virtual machine. Initially, all the virtual machines reliability is set as 100%. If the processing factor gives its result on a particular time limit, its reliability either increases or decreases. After assessing all the virtual machine reliability, the result is passed to the decision making. The reliability is calculated as follows:

$$Reliability = \frac{successful\ responses}{Total\ requests} * 100 \quad \textbf{(4)}$$

3.1.2 RDK-NN for Decision Making

This module helps to determine the best reliable node using Reliable Decision K-Nearest Neighbor (RDK-NN) algorithm. In this algorithm, the reliability level is fixed for each virtual machine. The maximum reliability factor VM is considered as the RDK-NN algorithm for decision making are explained as follows.

3.1.3 Pseudo code for Reliability Level Estimation

Step 1: Initially, load the data; fix the reliability factor as 0.3 to estimate the reliability of VM

Step 2: The similarity measure is calculated based on the Euclidean distance between query instances (resource constraints) and training samples from the non-functional testing to estimate the reliable VM. Euclidean distance is the commonly used distance when the data is in continuous form.

Step 3: The above non-functional testing and system constraint level are considered to calculate the K-nearest neighbor. A harmonic mean distance is calculated based on the sum of the harmonic average of the Euclidean distance metrics from one data point to another. The harmonic mean estimation of the RDK-NN algorithm is calculated by,

$$H_M = \frac{\sum_{fit}^{n} \sqrt{\sum_{i=1}^{n} (S_i - P_i)^2}}{N_i} \quad \textbf{(5)}$$

where, $f_{ti} \rightarrow$ non- functional testing with five testing factors (i defines the testing types)

S_i Result obtained from particular testing

$P_i \rightarrow$ Resource constraint level

$N_i \rightarrow$ number of testing involves detection

Step 4: The distance is sorted and the nearest neighbors are determined based on the K-th minimum distance. The node that achieves the highest reliability factor is considered as the best reliable VM.

Step 5: During the selection, if two nodes have the same highest reliability level, the node with a smaller IP address is selected as the computing cycle output.

From the above steps, the reliable VM is selected among all the individual Virtual machines. Then the result is scheduled using fault tolerance mechanism to determine the high-reliability factor-based VM.

3.2 Scheduling Mechanism

The result achieved from the first level is then send to the second level of the fault awareness scheduling mechanism. In this, the task migration and fault detector approach are carried out to detect the fault at the initial stage. Next, task migration is performed for reassigning faulty jobs to the other available resources. After assessing the best reliable virtual machine, the results are scheduled using the TLBO method shown in Table.1. It is used to determine the optimal resource allocation for the task within the minimization of task execution time in the cloud system.

3.2.1 Teaching Learning Based Optimization (TLBO) Scheduling in Cloud Environment

TLBO is a new meta-heuristic nature-based algorithm in obtaining the optimal solution for allocating the task to the resources which reduces the makespan and the cost in the entire cloud system. It is a population-based method that determines the global search in optimizing the task and obtains the best VM at scheduling. The TLBO method is divided into two phases: They are learner phase and the teacher phase. In the learner phase, the learning is obtained between the learners, and the teaching phase is carried out between the learner and the teacher. In this algorithm, the population which consists of different variables is considered as a group of learners or class of learners. The different variables define the various subjects that are offered as the students. Here the candidate solution consists of the objective function which defines the knowledge of the students and the solution with the best fitness function is considered as the teacher.

The steps in TLBO optimization algorithm are as follows.

Step 1: Initialization

Initialize the number of optimization parameters such as the number of tasks (cloudlets) and the termination criteria.

$$cloudlet_N = \{T_1, T_2, T_3, ..., T_n\}$$

Step 2: Estimation of Population

A random population is generated according to the size and number of VMs. In TLBO, some tasks indicate the learners and VM defines the teachers. The population is expressed as follows:

$$\text{Population} = \begin{bmatrix} P_{1,1} & P_{1,2} .... & P_{1,v} \\ P_{2,1} & P_{2,2} ..... & P_{2,v} \\ ... & .... & .... \\ P_{T_n,1} & P_{T_n,2} & .....P_{T_n,v} \end{bmatrix} \quad \textbf{(6)}$$

where,

$$T_n \rightarrow number \ \ of \ tasks$$
$$v \rightarrow VM$$

Step 3: The computational ratio (VM_{cr}) is calculated and the computational share (VM_S) of the VM in the cloud are calculated by,

$$VM_{cr(f)} = \frac{VM_f MIPS}{\sum_{i=1}^{n} VM_i \ MIPS}, \forall_f = 1,2,3....,n \quad \textbf{(7)}$$

$$VM_{S(f)} = (\sum_{i=1}^{n} cloudlet_l .MI) \times VM_{cr(f)} \ , \ \forall_f = 1,2,3....,n \quad \textbf{(8)}$$

where MIPS denotes Millions of Instructions Per Second (VM computing power) and MI is the Million Instructions (cloudlet size).

From the above results, determine the VM with the high computational ratio in the system.

Step 4: Teacher phase

The provider derives the fitness function by reducing the completion time, while the client aims to

reduce the price of accessing cloud resources by reducing the makespan time. Therefore, the fitness value of the can be computed as

$$f(x) = \min\left\{\bigcup_{i=1}^{m} CT_{ik}\right\} \quad \textbf{(9)}$$

Where $CT_{ik} \rightarrow$ completion of tasks.

The task completion criteria (CT_{ik}) defines the task to be completed on the VM is calculated as,

$$CT_{ik} = \sum_{i=1}^{n} \frac{cloudlet_i.MI}{VM_k MIPS} \quad \textbf{(10)}$$

From equation (9), consider the minimum value $f(x)$ and the corresponding task is determined as the best virtual machine (K-best).

Step 5: Determine the new teacher set (mean function) by modifying the solution based on the k-best solution. It is obtained by the difference (F_D) between the result of VMs and the mean result of a task for the new set of iteration in each user.

$$F_D = r(X_{j,kbest,i} - T_f M_{j,i}) \quad \textbf{(11)}$$

Here, the r_i random value lies between 0 & 1.

$X_{j,\,kbest\,i} \rightarrow$ K-best's relevant task

$T_f \rightarrow$ Teaching factor where 0 and 1 define the teaching quality which termed failed and the reliable VM

$M_{j,i} \rightarrow$ Average task of each user in iteration From (11), the result is added to the current solution to update the values. It is given by

$$X_{NEW} = X_{OLD} + F_D \quad \textbf{(12)}$$

where $X_{NEW} \rightarrow$ Updated fitness function

$X_{OLD} \rightarrow$ The task within the design variable

Based on the updated functions (best fitness function), the new values become input to the learner phase.

Step 6: Learner Phase

In this, consider the result from the teacher (X_{NEW}) and their interaction between (X_{NEW})" themselves. A random interaction is determined between the learners to obtain new knowledge within the learners.

Consider two random tasks p and q, send access cloud resources with formations $X_{NEW,\,p}$ and $X'_{NEW,\,q}$ and update the tasks based on the fitness comparison with a probability of task to access VM resources at time t.

Randomly select two learners X_p and X_q, $p \neq q$

If $f(X_p) \prec f(X_q)$

$X''_{NEW,\,p} = X_{NEW,\,p} + r(X_{NEW,\,p} - X'_{NEW,\,q})$

ELSE

$X''_{NEW,\,p} = X_{NEW,\,p} + r(X'_{NEW,\,q} - X_{NEW,\,p})$

End if

Accept, X_{NEW} if it gets the best fitness function

Step 7: Termination

Check if the termination criteria (task processing to each VM) is satisfied. If yes, the optimal schedule set is achieved; otherwise, iteration is performed by repeating the step from 3 to 6.

Table 1. Algorithm for the proposed Multi-Level Fault Tolerance Mechanism

Input:- A set of tasks, resources
Output:-To obtain the reliable and best VM (resources) for the corresponding users
1. Initialize *n* number of tasks to SP
2. Perform five types of non-functional testing //VM fault detection
3. Determine best VM using RD-KNN decision making algorithm
Set the reliability factor as 0.3
Estimate reliable VM based on Euclidean distance
Calculate H_M based on the average result of Euclidean distance measure
Sort the distance and finds the k-nearest neighbor
Select node with high reliability level as reliable VM
4. Initialize the algorithm parameters of cloudlets, and termination criteria // TLBO algorithm
Generate the random population as in eqn (6)
Calculate VM *cr* and VM_S of VM //teachers phase
Determine CT_{ik}
Compute fitness function based on equation (9) and d etermine as K-best (best VM)
Update the new set of fitness function based on equation (12) and sent to learners phase
Compute the interaction between X_{NEW} and X_{NEW}'' // Learners phase
5. ***Accept*** the task of X_{NEW} with best fitness function
6. ***Terminate***
If processing criteria is achieved
Else
Repeat learners and teachers phase
Schedule the task based on the minimum fitness function measure to cloud
7. End

Figure 2. shows the flowchart of the TLBO scheduling technique. At first, initialization and population estimation are performed to determine the total population (task) in the cloud systems. Then, the solution update is performed based on the best value calculated by comparing the new solution with the existing solution. Then, this updated solution set is send to the learner phase that schedule the task based on the fitness function.

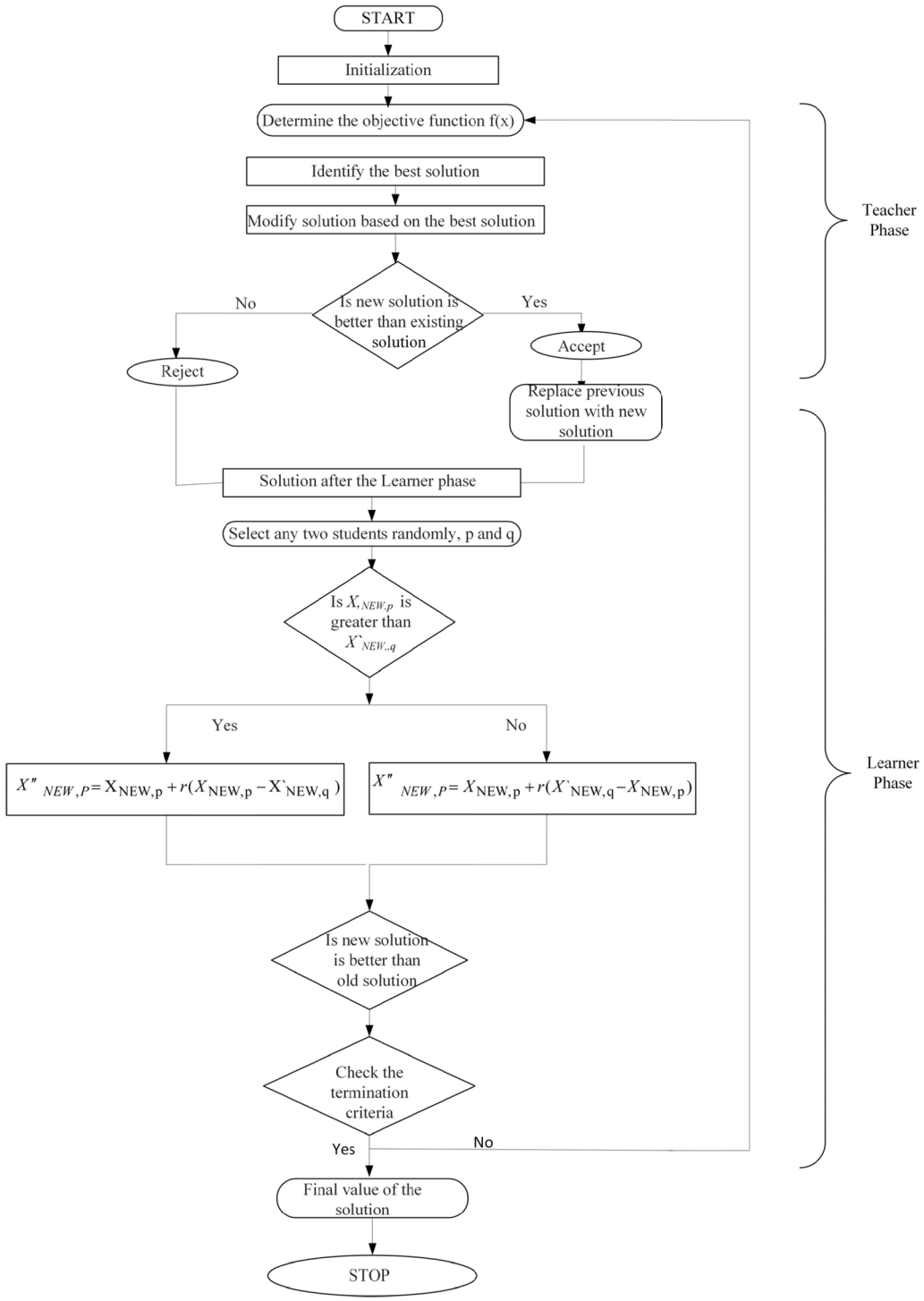

Figure 2. Flow Chart of the TLBO scheduling mechanism

After performing the above operations, the task is scheduled based on their fitness measure and send to the particular VM and direct to the specified host. The remaining task which is not scheduled is resend to the fitness calculation. Hence an iteration procedure is carried to schedule the VM with minimum fitness to the cloud resource.

During scheduling the task into VM, the fault occurs due to the resource or system break down. Hence restoration is performed to obtain the status of the task. Thus, when the fault occurs, the failed task or VM is traced and then schedules the succession task with a

TLBO scheduling algorithm. The fault usually occurs at the application, VM level, and host level. The application-level faults are recovered by the Checkpoint/Restart (CR) method that stores or upgrades the status of the job. The VM level faults are solved by recreating the resources from another host. For this, it uses the snapshot technique that rollback the file system which contains the application state, output data as well as the system configuration. At the host level, the faults are resolved by scheduling the VM to another host. By restoring the status, the VM mechanism aims to increase the reliability of the cloud providers as well as the users. Some of the advantages are (a) avoid saving the whole state VM and renting additional VM during node redundancy, (b) the checkpoint point file helps in recreation as well as the system rollback. Initially, local check pointers save the checkpoint state to the disk, and then a temporary checkpoint file is taken, and finally, the files are transferred asynchronously.

After status restoration, the process again restarts by reassigning and rescheduling the job in the queue of insufficient resources to another accessible VM. It reduces the load balancing problem and provides fault tolerance effectively by utilizing the saved state of the application. In this, the failed task migrates to the other available or the underloaded computing nodes (VM) for execution. Also, the suspended task that occurs due to overloading is instantly scheduled to the other alternative nodes. The task migration is helpful in the following scenarios (a) load balancing problems for VM overloaded systems (b) fault tolerance awareness and (c) migration which can be done based on the resource request in the cloud. In comparison with the traditional checkpoint mechanism, the proposed checkpoint approach does not require any redundant VM on standby nodes and also eliminates the run time processing in the recovery stage. Also, the checkpoint is processed in two different streams where the users have the flexibility to choose an appropriate implementation for the process of checkpoint mechanism. This seems that the backup overhead is reduced to the maximum extent and can be applied for the large usage of cloud**.**

4 Experimental Results and Discussion

This section provides the Metrics, dataset details, and the results of our proposed approach. The proposed multi-level fault aware scheduling mechanism is evaluated under the Cloudsim simulation platform [32].

4.1 Dataset Description

This method is evaluated with the Google Cloud Job (GoCJ) dataset [33]. The dataset comprised of 19 files containing the data that describes the size of the job expressed in Millions Instruction (MI). The data is acquired from the Google cluster traces and is created using the Monte Carlo method. The data file is expressed in "GoCJ_Dataset_XXX.txt, where XXX defines the number of jobs (e.g. "GoCJ_Dataset_ 800.txt"). The size of the job ranges from 15000-900000 MI and is classified as small, medium, large, extra-large, and huge. The cloudlet sizes for the dataset lies between the ranges: small (15000-55000 MI) of about 20%, medium (59000-99000 MI) of 40%, large (101000-135000 MI) of 30%, extra-large (150000-337500 MI) of 4% and Huge (525000-900000 MI) of 6%. This method is evaluated with 500 tasks from the GoCJ dataset in the ratio of small (19%), medium (38%), large (31%), X-large (5%), and huge (7%).

4.2 Evaluation metrics

The proposed method is evaluated under the three metrics which include makespan time, failure ratio, and failure slowdown. The following describes the fault tolerance parameter metrics [34].

4.2.1 Makespan Time

Makespan is defined as the maximum completion time that a resource taken to complete the latest task. The lesser of makespan denotes the better service quality. The makespan corresponding to the optimization criteria in scheduling should be minimized. Equation (9) defines the makespan evaluation formula in the cloud systems. In cloud systems, the completion time should be low thereby reducing the cost of the cloud resources.

4.2.2 Failure Slowdown (FSD)

It is defined as the ratio between the interruption or time delay occurred by the failure-to-failure free job execution time and the average over the total number of jobs. The FSD of the multilevel fault aware scheduling method should be smaller than other techniques and is expressed as follows:

$$FSD = \frac{time\ delay\ occured\ by\ failure\ to\ failure\ free\ job\ execution\ time}{average\ over\ total\ jobs} \quad \textbf{(13)}$$

4.2.3 Failure Ratio (FR)

It is defined as the ratio between the total failures of task happens in the proposed technique to the total failure in the other scheduling algorithms. The proposed fault aware scheduling method FD ratio should be better if the result comes out of less than 1. It is expressed as follows:

$$FR = \frac{time\ of\ failures(proposed\ method)}{total\ number\ of\ failures(other\ methods)} \quad \textbf{(14)}$$

4.2.4 Experimental Setup

This method was evaluated with 500 tasks from the GoCJ dataset in the ratio of small (19%), medium (38%), large (31%), X-large (5%), and huge (7%). The Cloudsim parameter configuration is listed in Table 2.

Table 2. Cloudsim parameters

Size of the job(15000-900000 MI)	Range
CPU	500-2500 MIPS
Memory	500-4096 MB
BW	500-1000 bit
Number of VM	50
MIPS of Processing Element (PE)	250-23000
VM memory RAM	256-4500 MB
No. of PEs per VM	1-8
Datacenter no	10,15,20,30
Host no	2-500
Type of manager	Time Shared
Cloudlet	500
System architecture	X86
Virtual Machine Manager	Xen
OS	Windows

4.3 Observed Result

The performance of the proposed multilevel fault aware scheduling technique is evaluated under makespan, failure ratio and failure slow down.

Figure 3 shows the result of the makespan metrics calculated with 500 tasks and the comparison is carried out under the MTCT, DCLCA, and CPLCA techniques. The result indicates, when increasing the number of cloudlets, makespan (execution time) keeps on increasing. The existing method shows a high makespan time compared to the proposed technique. This outcome shows that the proposed multi-level fault aware scheduling method utilizes less execution time than the existing approaches.

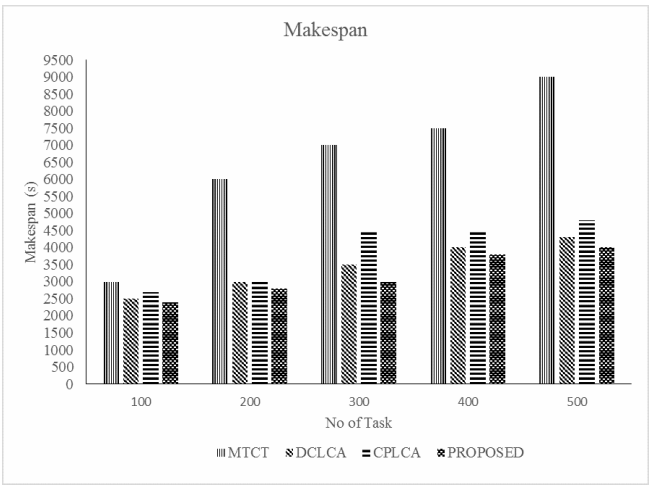

Figure 3. Makespan time results

Figure 4 shows the failure ratio comparison of the proposed scheduling method with the other algorithms. The FR varies from 0.66-0.35 as the task gets increasing in the MTCT technique while DCLCA varies from 0.5-0.12 and the CPLCA method varies from 0.49-0.375. However, the proposed method performs well and obtains the lowest range in the failure ratio.

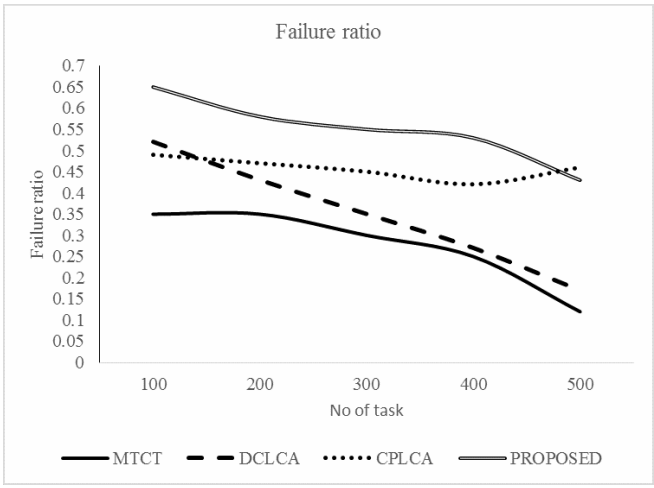

Figure 4. Failure Ratio (FR) of Multilevel Fault aware scheduling algorithm

It clearly shows that the proposed method performs better than the existing techniques and the range lies within 1.

Figure 5. represents the failure slow down ratio compared with the existing algorithms. The FSD value increases when the number of tasks increases. The existing MTCT, DCLCA, and CPLCA methods achieve the increasing range of FSD ratio, but the proposed technique performs lesser ranges even at the increasing number of task levels.

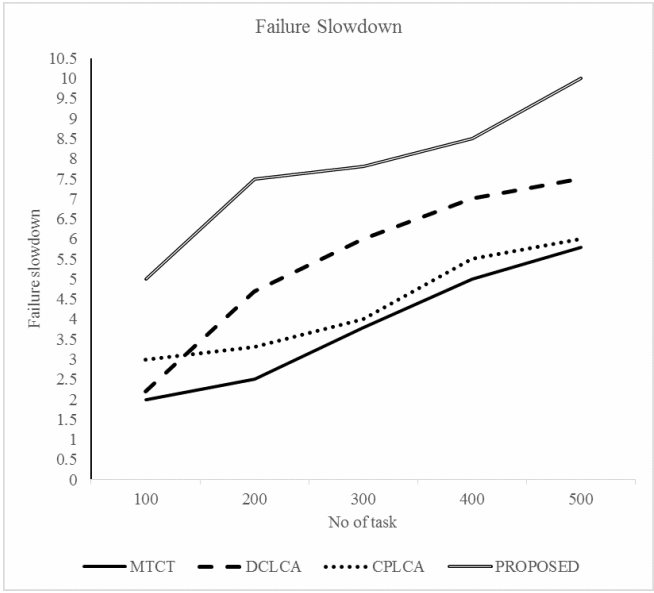

Figure 5. Failure Slowdown (FSD) of Multilevel Fault aware scheduling algorithm

5 Conclusion

In this paper, the multi-level fault aware scheduling mechanism in real-time cloud computing environment is addressed. In this, fault detection is carried out with the testing phase to determine the best reliable VM performed by using the RDKNN decision mechanism. Then the cloudlet is scheduled using the Teaching Learning Based Optimization Algorithm (TLBO with the best makespan (execution time). This algorithm schedules the task to the resources based on their fitness estimation. Due to this strategy, the VM resources are utilized effectively improving the reliability and availability in the cloud environment. The fault -aware technique was evaluated with the CloudSim toolkit. The experimental results show that the proposed scheduling mechanism for fault aware scheduling detection and scheduling provides better scheduling results than the other intelligent techniques.

References

[1] Q. Zhang, L. Cheng, and R. Boutaba, Cloud computing: state-of-the-art and research challenges, *Journal of internet services and applications*, Vol. 1, pp. 7-18, May, 2010.

[2] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, Survey Cloud monitoring: A survey, *Computer Networks*, Vol. 57, No. 9, pp. 2093-2115, June, 2013.

[3] F. Chang, J. Ren, and R. Viswanathan, Optimal resource allocation in clouds, *2010 IEEE 3rd International Conference on Cloud Computing*, Miami, FL, USA, 2010, pp. 418-425.

[4] Q.-Y. Huang and T.-L. Huang, An optimistic job scheduling strategy based on QoS for Cloud Computing, *2010 International Conference on Intelligent Computing and Integrated Systems*, Guilin, China, 2010, pp. 673-675.

[5] K. Lu, R. Yahyapour, P. Wieder, E. Yaqub, M. Abdullah, B. Schloer, and C. Kotsokalis, Fault-tolerant service level agreement lifecycle management in clouds using actor system, *Future Generation Computer Systems*, Vol. 54, pp. 247-259, January, 2016.

[6] A. Ganesh, M. Sandhya, and S. Shankar, A study on fault tolerance methods in cloud computing, *2014 IEEE International Advance Computing Conference (IACC)*, Gurgaon, India, 2014, pp. 844-849.

[7] J. He, M. Dong, K. Ota, M. Fan, and G. Wang, NetSecCC: A scalable and fault-tolerant architecture for cloud computing security, *Peer-to-Peer Networking and Applications*, Vol. 9, No. 1, pp. 67-81, January, 2016.

[8] M. N. Cheraghlou, A. Khadem-Zadeh, and M. Haghparast, A survey of fault tolerance architecture in cloud computing, *Journal of Network and Computer Applications*, Vol. 61, pp. 81-92, February, 2016.

[9] M. A. Mukwevho, and T. Celik, Toward a Smart Cloud: A Review of Fault-tolerance Methods in Cloud Systems, *IEEE Transactions on Services Computing*, pp. 1-1, March, 2018. DOI: 10.1109/TSC.2018.2816644

[10] C. N. Höfer and G. Karagiannis, Cloud computing services: taxonomy and comparison, *Journal of Internet Services and Applications*, Vol. 2, pp. 81-94, September, 2011.

[11] S. C. Wang, M. L. Chiang, K. Q. Yan, and Y. T. Tsai, Fault-diagnosis and Decision Making Algorithm for Determining Faulty Nodes in Malicious and Dormant Wireless Sensor Networks, *Journal of Internet Technology*, Vol. 19, No. 7, pp. 2135-2145, December, 2018.

[12] H. Jo, H. Kim, J. W. Jang, J. Lee, and S. Maeng, Transparent fault tolerance of device drivers for virtual machines, *IEEE Transactions on Computers*, Vol. 59, No. 11, pp. 1466-1479, November, 2010.

[13] S. H. Madni, M. S. Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment, *PloS one*, Vol. 12, No. 5, e0176321, May, 2017.

[14] M. Kalra, and S. Singh, A review of metaheuristic scheduling techniques in cloud computing, *Egyptian informatics journal*, Vol. 16, No. 3, pp. 275-295, November, 2015.

[15] S. H. Madni, M. S. Latiff, Y. Coulibaly and S. M. Abdulhamid, An appraisal of meta-heuristic resource allocation techniques for IaaS cloud, *Indian Journal of Science and Technology*, Vol. 9, No. 4, pp. 1-14, January, 2016.

[16] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, Cloud task scheduling based on ant colony optimization, *The International Arab Journal of Information Technology*, Vol. 12, No. 2, pp. 129-137, March, 2015.

[17] K. Devi D. Paulraj, and B. Muthusenthil, Deep Learning Based Security Model for Cloud based Task Scheduling, *KSII Transactions on Internet and Information Systems*, Vol. 14, No. 9, pp. 3663-3679, September, 2020.

[18] Z. Wang, L. Gao, Y. Gu, Y. Bao, and G. Yu, A fault-tolerant framework for asynchronous iterative computations in cloud environments, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 29, No. 8, pp. 1678-1692, August, 2018.

[19] N. Almezeini, and A. Hafez, An Enhanced Workflow Scheduling Algorithm in Cloud Computing, *6th International Conference on Cloud Computing and Services Science*, Rome, Italy, 2016, pp. 67-73.

[20] N. Jain, I. Menache, J. S. Naor, and J. Yaniv, Near-optimal scheduling mechanisms for deadline-sensitive jobs in large computing clusters, *ACM Transactions on Parallel Computing*, Vol. 2, No. 1, pp. 1-29, May, 2015.

[21] M. A. Rodriguez and R. Buyya, Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds, *IEEE transactions on Cloud Computing*, Vol. 2, No. 2, pp. 222-235, April-June, 2014.

[22] J. Wang, W. Bao, X. Zhu, L. T. Yang, and Y. Xiang, FESTAL: fault-tolerant elastic scheduling algorithm for real-time tasks in virtualized clouds, *IEEE Transactions on Computers*, Vol. 64, No. 9, pp. 2545-2558, September, 2015.

[23] C. Y. Chen, Task scheduling for maximizing performance and reliability considering fault recovery in heterogeneous distributed systems, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, No. 2, pp. 521-532, February, 2016.

[24] S. M. Abdulhamid, M. S. Latiff, S. H. Madni, and M. Abdullahi, Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm, *Neural Computing and Applications*, Vol. 29, No. 1, pp. 279-293, January, 2018.

[25] S. M. Abdulhamid and M. S. Latiff, A checkpointed league championship algorithm-based cloud scheduling scheme with secure fault tolerance responsiveness, *Applied Soft Computing*, Vol. 61, pp. 670-680, December, 2017.

[26] J. Zhou, K. Cao, P. Cong, T. Wei, M. Chen, G. Zhang, J. Yan, and Y. Ma, Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms, *Journal of Systems and Software*, Vol. 133, pp. 1-16, November, 2017.

[27] N. Almezeini and A. Hafez, Task Scheduling in Cloud Computing using Lion Optimization Algorithm, *International Journal of Advanced Computer Science and Applications (IJACSA)*, Vol. 8, No. 11, pp. 77-83, 2017.

[28] H. Xu, B. Yang, W. Qi, and E. Ahene, A multi-objective optimization approach to workflow scheduling in clouds considering fault recovery, *Transactions on Internet and Information Systems (TIIS)*, Vol. 10, No. 3, pp. 976-995, March, 2016.

[29] P. Zhang and M. Zhou, Dynamic cloud task scheduling based on a two-stage strategy, *IEEE Transactions on Automation Science and Engineering*, Vol. 15, No. 2, pp. 772-783, April, 2018.

[30] A. Marahatta, Y. Wang, F. Zhang, A. K. Sangaiah, S. K. Tyagi, and Z. Liu, Energy-Aware Fault-Tolerant Dynamic Task Scheduling Scheme for Virtualized Cloud Data Centers, *Mobile Networks and Applications*, Vol. 24, No. 3, pp. 1063–1077, June, 2019.

[31] H. C. Hsieh, M. L. Chiang, W. C. Tsai, and Y. C. Chen, A Feature-Oriented Fault Diagnosis Agreement Protocol in Distributed Systems, *Journal of Internet Technology*, Vol. 20, No. 5, pp. 1401-1413, September, 2019.

[32] R. N. Calheiros, R. Ranjan and A. Beloglazov, C. A. De Rose, and R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Software: Practice and experience*, Vol. 41, No. 1, pp. 23-50, January, 2011.

[33] A. Hussain and M. Aleem, GoCJ: Google Cloud Jobs Dataset for Distributed and Cloud Computing Infrastructures, *Data*, Vol. 3, No. 4, Article No. 38, December, 2018.

[34] R. Garg and A. K. Singh, Fault tolerant task scheduling on computational grid using checkpointing under transient faults, *Arabian Journal for Science and Engineering*, Vol. 39, No. 12, pp. 8775-8791, December, 2014.

Biographies

Devi K. received her B.E degree in Computer Science and Engineering from the Manonmaniam Sundharanar University, in 2003, the M.E degree from the Anna University, Chennai, in 2005.She is currently working as an Assistant Professor in the Department of Computer Science Engineering at SRM Valliammai Engineering College, Chennai. Her current research interests are Cloud fault tolerance, Scheduling, Deep learning approaches in Cloud and Network security. She is member of the ISTE, CSI and Indian Science Congress.

Paulraj D. received his PhD degree in computer science and engineering from Anna University.He has 20 years of experience including 9 years industrial experience. He has received his B.E (CSE) degree first class from Bangalore University in 1993, M.E (CSE) and Ph.D in Service Oriented Architecture from Anna University respectively in the year 2004 and 2012. During his Ph.D he has proved that Semantic Web Services can be composed using Process Model Ontology instead of Service Profile Ontology. He has published several research publications in refereed International Journals with high impact factor and International Conferences as well. He is the life member of ISTE and a member of IET. He had received IET Men Engineer Award in the year 2012. His research interests include Machine Learning, Data Science, Service Oriented Architecture (SOA), Semantic Web Services, Computer Network and Interface, Cloud and Grid Computing, Big Data Analytics, Block Chain Technologies, Augmented Reality, Virtual Reality.