Adaptive Fault Tolerant Resource Allocation Scheme for Cloud Computing Environments

Sathiyamoorthi V., Sona College of Technology, Salem, India

Keerthika P., Kongu Engineering College, Erode, India

Suresh P., Kongu Engineering College, Erode, India

Zuopeng (Justin) Zhang, University of North Florida, USA

Adiraju Prasanth Rao, Anurag Group of Institutions, Majarguda, India

 https://orcid.org/0000-0002-5119-3987

Logeswaran K., Kongu Engineering College, Erode, India

ABSTRACT

Cloud computing is an optimistic technology that leverages the computing resources to offer globally better and more efficient services than the collection of individual use of internet resources. Due to the heterogeneous and high dynamic nature of resources, failure during resource allocation is a key risk in cloud. Such resource failures lead to delay in tasks execution and have adverse impacts in achieving quality of service (QoS). This paper proposes an effective and adaptive fault tolerant scheduling approach in an effort to facilitate error free task scheduling. The proposed method considers the most impactful parameters such as failure rate and current workload of the resources for optimal QoS. The suggested approach is validated using the CloudSim toolkit based on the commonly used metrics including the resource utilization, average execution time, makespan, throughput, and success rate. Empirical results prove that the suggested approach is more efficient than the benchmark techniques in terms of load balancing and fault tolerance.

KEYWORDS

Cloud Computing, Fault Tolerance, Load Balancing, Quality of Service, Resource Allocation, Virtual Machine

1. INTRODUCTION

Cloud computing provides cost-effective computing resources usually with more reliable performance by sharing a large amount of resources with many users who consume the resources at different times (R. K. Gupta & Pateriya, 2017). Cloud computing delivers different types of services typically data storage and computing power services over the internet without direct active management of the hardware equipment by the users (Afzal & Kavitha, 2019). It is mainly used for sharing computing resources in order to accomplish coherence and economy of scale (Hicham, Said, Touhafi, & Ezzati,

2018). Cloud computing shows five major characteristics, including resource pooling, self-service on demand, rapid elasticity, wide access to network, and measured service. The usefulness of cloud includes scalability, reliability, low cost, flexibility and its availability, and has also been commonly used in the industry. (Vella, Yang, Anwar, & Jin, 2018). A general architecture of the cloud computing is shown in Figure 1.

The services of the cloud can be grouped into Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) to clients via Internet (Belalem & Limam, 2011). SaaS provides the software applications as services to the clients via Internet on a remote basis. It offers cloud-based software which is hosted online by an organization and is available for payment purpose via Internet. This type of service is easy to use and manage, as it is not required to be downloaded and installed on individual devices. SaaS may have some issues such as security, interoperability and lack of integration if the applications are not developed with open standards (Chou, 2019). PaaS provides the facility to develop and distribute personalized applications in a hosted environment via web to the clients. It provides services to the developers with a framework to build upon custom applications. In PaaS model, data resides in vendor-controlled cloud server poses security risks and concerns. IaaS allows the clients to utilize the hardware and resources remotely on a "pay-as-per-use" model. These services can replace most of the local solutions with improved performance. It facilitates individuals or organizations to build and manage their hardware or software resources as they grow and pay only for the resources that they have consumed. In IaaS environment, the organization or company is not having any control over cloud security and is only responsible for any upgrades and maintenance of software (Jyoti, Shrimali, Tiwari, & Singh, 2020).

Cloud computing is the foundation of many business solutions now and it is more reliable than on-premise hardware implementations (Amoon & Tobely, 2019). Due to the increase in demand for resources lead to an increase in the services, and thus the establishment of large-scale data centers (Abderrahim & Choukair, 2018). Despite the great promise of cloud computing, several categories of defects may arise in cloud which leads to degradation of performance and many failures. The primary categories of faults in cloud include network fault, physical fault, process fault and service execution fault (Kathpal & Garg, 2019). Network faults arise when resources or services are retrieved through a network and a significant cause of resource failures. The causes of this can be multiple, such as packet corruption or loss, network partitions, congestion, node failure or link failure, etc (V. Gupta, Kaur,

Figure 1. General Cloud Architecture
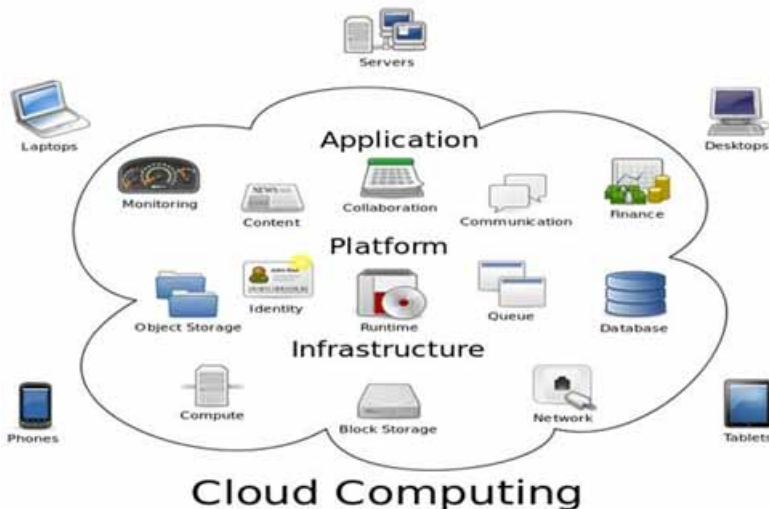
& Jangra, 2019). Physical faults may occur in the hardware resources including failures or faults in processor, memory, I/O devices and storage. Process faults are associated with the processes due to shortage of resources, software bugs, or ineffectual processing capabilities (Garraghan, Townend, & Xu, 2014). Service execution fault occurs if the user or application uses it when the service time of the resources expires. These faults usually result in one of the major failures occurred in cloud environments, including hardware failure, virtual machine failure and application failure (Garraghan et al., 2014). Therefore, fault tolerance measures are vital for cloud computing due to its high dynamicity, huge number of the resources and possibilities of failures in cloud resources.

Fault tolerance requires the development of a blueprint for continuing services even if a few resources in the cloud are down or inaccessible. It prevents network device or computer resources from failures due to any faults in the execution (Parvez, Robel, Rouf, Podder, & Bharati, 2019). Fault tolerance is needed in order to provide assurance for availability and reliability of critical resources as well as task execution. It includes the techniques necessary for robustness, failure recovery and improving overall performance. It also requires for bringing reliability for every resource in cloud environment (Kumari & Kaur, 2018).

Fault tolerance can be implemented using two categories of approaches, based on the policies and procedures such as proactive and reactive. First, proactive fault tolerance approaches predict the deteriorating virtual machines well in advance, in that way the downtime of resources is reduced. This group of methods aims to avoid the time spent for recovering errors, faults and failures and proactively replacing those failed resources with the other active or working resources. Proactive fault tolerance method is implemented with the help of techniques such as preemptive migration, software rejuvenation and workload balancing. Second, reactive fault tolerance approaches focus on reducing the effect of failures on task execution if there is any resource failure. It is implemented using check-points and restart mechanisms (Prathiba & Sowvarnica, 2017).

This paper proposes an Adaptive Fault Tolerant Resource Allocation (AFTRA) approach, as a proactive fault tolerance measure. In particular, this approach is implemented based on an application that is self-adaptive with respect to its current position or state in its state space and the applications or tasks that are submitted. This effectively ensures procedures that are done automatically according to the situation and thus providing better reliability to critical tasks under temporal and resources constraints. As a compromise, less critical tasks are allocated with as much resources as the system can afford without affecting the procedures for the critical tasks, thus gracefully reducing the requirement of the resources.

The paper is organized as follows. Section 2 reviews the relevant works in fault tolerant resource allocation. Section 3 comprehends the suggested adaptive fault tolerant resource allocation approach. Section 4 presents empirical results and Section 5 concludes the proposed method with future work.

2. BACKGROUND

2.1 Proactive Fault Tolerant Resource Allocation Methods

Nowadays many researchers focus on solving the issues in cloud computing, but only few focused on fault tolerance mechanisms for the cloud environment. A Fault Tolerant Scheduling Method (FTSM) is proposed for allotting most appropriate device to services' requests in fog cloud IoT systems. This research is focused on reducing the service latency and service overheads and increases the reliability. It categorizes the requests into three categories based on the kind of service needed which are time-tolerant, time-sensitive and core. The usefulness of this strategy is evaluated using average service time, capacity percentage, operation costs, throughput and success rate (Alarifi, Abdelsamie, & Amoon, 2019).

A fault tolerant VM allocation (FTVMA) strategy is proposed in which resource allocation is done by accounting the failure history, failure rate and execution competence of the virtual machines. It implements a fault aware scheduling policy for discovering a suitable resource by considering the

failure rate and makespan (Keerthika, Suresh, Manjula Devi, Sangeetha & Sagana, 2019). A fault aware job scheduling method is proposed which uses the check-pointing strategy with job migration when resource failure occurs (Latiff & Shafie, 2017).

A hierarchical model for load balancing is developed for cloud computing which considers failure history along with user's satisfaction. It provides the proactive fault tolerant method that achieves better hit rate, reduced communication overhead and better user satisfaction when compared with other benchmark algorithms (P Keerthika & P Suresh, 2015). A smart checkpoint framework is developed using union file system to distinguish read only parts and read write parts in data centre (Goiri, Julia, Guitart, & Torres, 2010). This approach allows read-only sections to be check pointed once, while the remaining checkpoints only need to save the changes in read-write sections, thus minimizing the time necessary to make a checkpoint. A task scheduling technique with load balancing strategy called Minimum Completion Time (MCT) for multi processor environment is proposed in order to schedule the tasks. It considers the ready time and expected completion time of the jobs and aiming to achieve better resource utilization by balancing the load at the resources (Mishra, Sahoo, & Parida, 2020).

2.2 Reactive Fault Tolerant Resource Allocation Methods

A Greedy based methodology for cloud environment is proposed in which the jobs are categorized based on quality of service. Then, appropriate resource is selected based on the tasks categories (Li, Feng, & Fang, 2014). A job scheduling model for multi-processor environment is proposed which considers the state of all resources and chooses the resource with maximum computing power or capacity to run the current job (Manimala & Suresh, 2013). A combined pricing and scheduling model is proposed for delay-tolerant batch services in order to improve the long-term profit of the service providers (Ren & van der Schaar, 2013).

Load Balanced Improved Min-Min (LBIMM) strategy is implemented for decreasing the makespan and to amplify the utilization of the resources (Chen, Wang, Helian, & Akanmu, 2013). A multi-constrained load balancing algorithm with fault tolerance is proposed in order to allocate resources to the tasks with budget constrained and load balancing. It considers the basic requirement factors such as processing time, resource failure, cost, system load and user's deadline while allocating resources to the jobs (Keerthika P & Suresh P, 2015). A load balanced user demand aware scheduling is proposed for grid environment that performs load balancing by considering the load of each resource to minimize the response time and also increases the resource utilization. Since it considers the user deadline of the tasks, the user satisfaction is greatly improved (Suresh & Balasubramanie, 2013).

An improved task scheduling methodology is implemented that mainly works for scientific workflows. It concentrates on minimizing the cost, execution time and improving the utilization of the resources. In this algorithm, Direct Acyclic Graph (DAG) is constructed through which the tasks are categorized into different groups to decrease communication overhead. It focuses on minimizing makespan of workflow and maximizing the utilization of the resources for both computational intensive workflows and data intensive workflows (Geng, Mao, Xiong, & Liu, 2019).

A user deadline aware scheduling algorithm for grid environment is developed which is mainly used to schedule the data intensive tasks. It considers communication time, user deadline of the tasks in order to minimize makespan, communication overhead and improve hit rate (Suresh & Balasubramanie, 2012). A task scheduling strategy for grid computing is suggested and considered user deadline while distributing jobs to various diverse resources from various domains (Suresh et al., 2011). Table 1 shows the state of art of various scheduling parameters considered in different scheduling algorithms.

Even though many research works have been carried out in cloud computing, unfortunately very few focuses are made on combining both fault tolerance and load balancing to develop resource allocation methods. Usually an inefficient task scheduling algorithm which does not take the failure rate and load of the resources will create major impact on QoS parameters like completion time of the jobs and user satisfaction. For example, if a job is allocated to a cloud resource which has more failure

Table 1. Scheduling Parameters considered in different Scheduling Algorithms

Methodology	Type (Proactive/ Reactive)	Execution Time	Response Time	Cost	Makespan	Scalability	Resource Utilization	Load Balancing	Failure Rate	QoS Satisfaction
FTSM (Alarifi et al., 2019)	Proactive	✓	✓	✓	✓	✗	✗	✗	✓	✗
FTVMA (Keerthika et al., 2019)	Proactive	✓	✓	✗	✓	✗	✓	✓	✓	✗
Fault aware job scheduling method (Latiff & Shafie, 2017)	Proactive	✓	✓	✗	✗	✗	✗	✗	✓	✓
Hierarchical model (P Keerthika & P Suresh, 2015)	Proactive	✓	✓	✓	✗	✗	✗	✗	✓	✓
Smart checkpoint framework (Goiri et al., 2010)	Proactive	✓	✓	✗	✓	✗	✗	✗	✗	✗
Minimum Completion Time (Mishra et al., 2020)	Proactive	✓	✓	✗	✓	✓	✓	✓	✗	✗
Greedy based methodology (Li et al., 2014)	Reactive	✓	✗	✗	✓	✗	✗	✗	✗	✓
Job scheduling for multi-processor environment (Manimala & Suresh, 2013)	Reactive	✓	✓	✗	✗	✗	✗	✗	✗	✗
Combined pricing and scheduling model (Ren & van der Schaar, 2013)	Reactive	✓	✓	✓	✓	✗	✗	✗	✗	✓
LBIMM (Chen et al., 2013)	Reactive	✓	✓	✗	✓	✗	✓	✗	✗	✗
Multi-constrained load balancing algorithm (P Keerthika & P Suresh, 2015)	Reactive	✓	✓	✗	✓	✓	✓	✓	✗	✗
Load balanced user demand aware scheduling (Suresh & Balasubramanie, 2013)	Reactive	✓	✓	✗	✓	✗	✓	✓	✗	✓
DAG based Algorithm (Geng et al., 2019)	Reactive	✓	✓	✗	✓	✓	✗	✗	✗	✗
User deadline aware scheduling algorithm (Suresh & Balasubramanie, 2012)	Reactive	✓	✓	✗	✓	✓	✗	✗	✗	✓
Task scheduling strategy with user deadline (Suresh, Balasubramanie, & Keerthika, 2011)	Reactive	✓	✓	✗	✓	✓	✗	✗	✗	✓
Proposed (AFTRA) Algorithm	Proactive	✓	✓	✗	✓	✓	✓	✓	✓	✓

rate and workload, then indisputably it leads to either delay in job completion or job execution failure or job migration which affects the QoS parameters such as less success rate of job completion, more execution time and cost. Based on the survey, an efficient scheduling algorithm should focus on the various parameters such as load balancing, fault tolerance and QoS parameters like execution time, cost and security related metrics. But, considering all the parameters in a single framework may increase the complexity based on the cloud type. So, developing an adaptive method can efficiently improve the reliability by considering resource failure rate and availability by considering the workload of the resources particularly in real-time cloud environment. The proposed work introduces an innovative proactive fault tolerance method which effectively considers the fault rate and VMs workload to successfully reduce its unavailability.

3. PROPOSED FAULT TOLERANT RESOURCE ALLOCATION

3.1 Cloud Center

Cloud comprises of heterogeneous and highly dynamic resources from various data centers that can be applied to solve the problems which are submitted by the clients. In this cloud, task scheduling or allocating resources to the jobs submitted by the users is a NP-hard optimization problem. The arrival rate of the tasks usually assumes that the inter arrival times of the tasks are independent and have an ordinary distribution. In this research, the tasks are assumed to arrive according to a Poisson distribution.

This work is developed based on a cloud center following the multi-server M/M/c queuing model. This model is notated using Kendall's notation that describes a task scheduling system where a queue is maintained for arriving tasks and managed by a Poisson distribution. In M/M/c model, the first letter indicates the inter arrival time, the second signifies the service time and last letter signifies the total virtual machines. In the proposed policy, tasks arrive one by one which are always allowed to enter the cloud; there is always a task queue; and there are no priority rules and tasks are served in order of arrival. The system state is characterized by the number of tasks in the system. If there are n asks in the system, then p_n denotes the equilibrium probability which can derive from the flow diagram as depicted in Figure 2.

3.2 Proposed Method

The proposed cloud architecture is shown in Figure 3. The cloud resource broker will collect the tasks or jobs from the user through the portal. Then, the tasks submitted by the users are stored in the request pool which is a queue of the tasks. The resource pool component collects the information about the resources from the Virtual Machine (VM) information system which has information about the VMs that are managed and hosted by the cloud infrastructure and management framework. A virtual machine is a virtual representation of a physical computer where the users' tasks get executed. Fault detector component is used to detect faults in the cloud resources and maintains fault rate of the resources which may be used while allocating resources to the tasks. The workload predictor

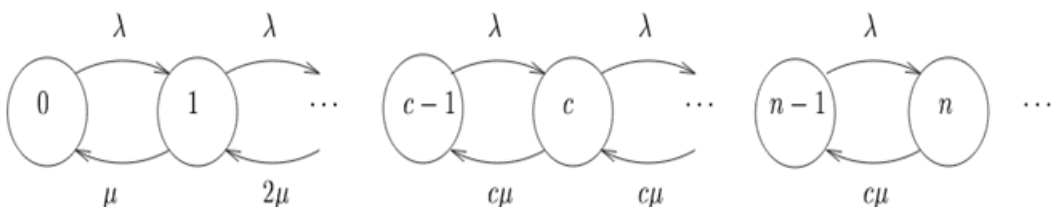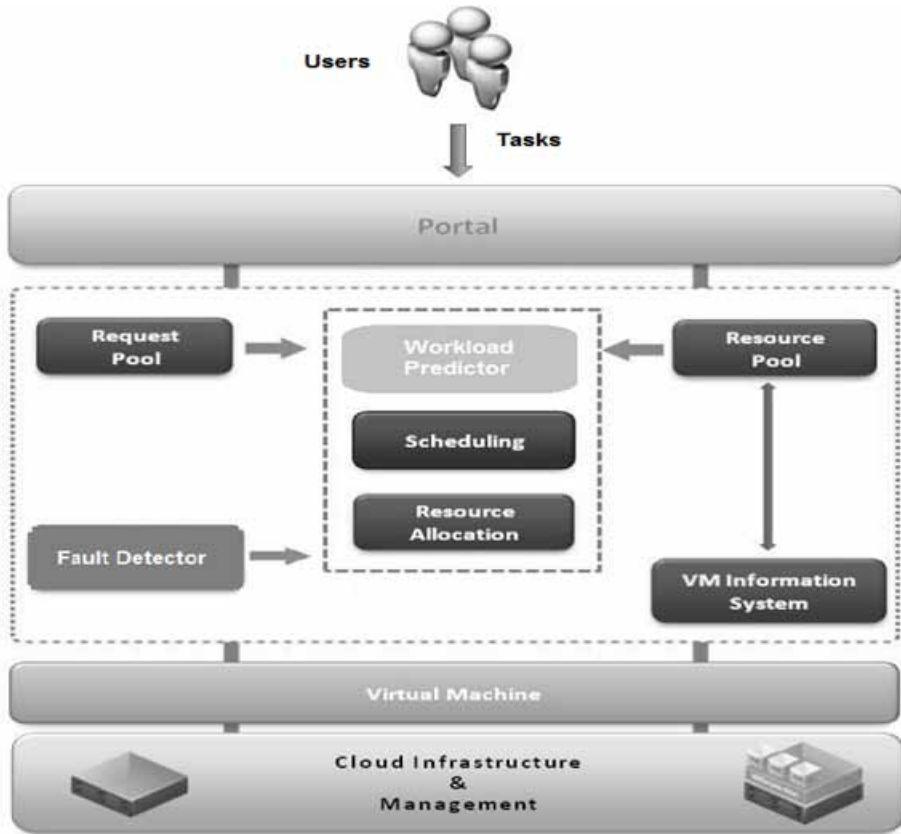Figure 2. Flow diagram of M/M/c model

Figure 3. Proposed Cloud Architecture Model

component identifies and maintains the workload of each virtual machine in order to avoid over busy and idleness of the resources. Finally, scheduler and resource allocator components allocate more suitable resource to the tasks submitted by the users.

The pseudocode of the proposed Adaptive Fault Tolerant Resource Allocation (AFTRA) system is shown in Algorithm 1. In the cloud environment, failure of resources is common due to inevitable hardware or software failures. For developing an effective resource provision policy, the failure rate of the cloud resources needs to be taken into account. The failure directly affects the reliability of the cloud system. Due to the resource failure, the tasks that are allocated to these resources will be delayed or failed. Usually, the failures in cloud environment may happen stochastically, which can be represented using a Poisson probability distribution. The terminologies used in the proposed strategy are listed in the Table 2.

Non-homogeneous Poisson process model is used to represent the number of failures occurred up to time t, denoted as $\left\{ N(t), t \geq 0 \right\}$. The major issue is determining a suitable mean value that denotes the expected failures that may occur during certain period of time. It shows that the number of resource failures will not be same for any two different time intervals. In other words, the failure rate of a resource in the time interval $(t, t + s)$ depends on the present time t and the span of time interval s and does not depend on the failure history of the resource. On the basis of non-homogeneous Poisson probability distribution, the expected failure rate (EFR) of a resource during a certain time interval from 0 to T is calculated as:

Algorithm 1. Pseudocode of the AFTRA algorithm

```
Begin
   While there are tasks in Task Queue
      Select a task i from the Queue
      For all Virtual Machines
          Calculate Expected Failure Rate (EFR) and probability of the failure of the VM
          Calculate the workload of the VM and denote it as WL
          Classify the virtual machine such as overloaded, under loaded and normally loaded as per the
          Pseudocode given in Figure. 3
      End for
   Assign the selected task to the virtual machine which has minimum probability of failure in the under
   loaded list
   End while
   Remove the task i from the queue
End
```

Table 2. AFTRA Terminologies

Terminology	Description
EFR_j	Expected Failure Rate of the Virtual Machine j
WL_j	Virtual Machine j Workload
$\mu_j(t)$	Mean failure rate of virtual machine j
MI_k	Length of task k represented in Million Instructions (MI)
$MIPS_j$	Virtual machine j speed denoted as Million Instructions Per Second (MIPS)
AT_j	Availability time of the virtual machine j
TH	Threshold value of the workloads
$P\big(N\big(T\big) = n\big)$	The probability of the occurrence of n failures for virtual machine
ET_j	Execution time of virtual machine j
D_{ij}	Decision variable of virtual machine j when task i is assigned
T_c	Number of tasks completed within user deadline
T_s	Number of tasks submitted

$$EFR(T) = \int_0^T \mu_j(t)\, dt \qquad (1)$$

The probability of n failures which occur during the time interval $(0, T)$ for the Non-homogeneous Poisson process is given by:

$$P\big(N(T) = n\big) = \frac{EFR(T)^n}{n!} e^{-EFR(T)}, 0 \le P(n) < 1 \qquad (2)$$

In order to maintain a balanced load in cloud, the virtual machine load is calculated as:

$$WL_j = \frac{\sum_{k=0}^n MI_k}{MIPS_j \times AT_j} \qquad (3)$$

where $'n'$ is number of jobs scheduled to the jth VM (VM_j).

For assigning a virtual machine state, the threshold value (TH) is determined by calculating the mean value of the workloads of the virtual machines as follows:

$$TH = \frac{\sum_{j=1}^m WL_j}{m} \qquad (4)$$

where $'TH'$ is workloads threshold and $'m'$ denotes total number of available virtual machines.

The pseudocode of the load classifier is summarized in Algorithm 2, which is used to monitor the available virtual machines and organize them into three classes such as under loaded, heavily

Algorithm 2. Pseudocode of the Load Classifier

```
Begin
    For all virtual machines in the cloud
            Select a VM j from virtual machine pool
             Calculate the workload (WL) of each virtual machine
            Calculate the Threshold value (TH) of the workloads of VMs
             if(WLj< TH)
                    vm.state= "under loaded"
                    Add the VM to Under_loaded list
             else if (WLj> TH)
                    vm.state= "heavily loaded"
                    Add the resource to Heavily_ loaded list
             else
                    vm.state= "normally loaded"
                    Add the resource to Normally_loaded list
            End if
    End for
End
```

loaded and normally loaded according to the workload of the VMs. Whenever the task is selected to schedule, it will be allocated to the virtual machine which has minimum probability of failure in the under loaded list. Finally, the proposed AFTRA method provides a fault-tolerant service for the user's task or job by selecting the more relevant virtual machine for the resource pool.

4. SIMULATION SETUP AND PERFORMANCE EVALUATIONS

The AFTRA strategy is simulated and tested using CloudSim toolkit which is a framework for simulating and modeling the core functionality of cloud such as task queue, creation of cloud entities, processing of events, communication between different entities, accomplishment of resource broker procedures, etc. It is often used for load balancing, job scheduling, optimizing virtual machine allocation and positioning strategies, aggregation or relocation of virtual machines, and optimizing network latency schemes under different scenarios. The proposed AFTRA method is compared in two scenarios with other algorithms such as Fault-Tolerant Scheduling Method (FTSM) (Alarifi et al., 2019), Improved Load Balanced Min-Min algorithm (LBIMM) (Chen et al., 2013), MCT (Mishra et al., 2020) and Fault Tolerant Virtual Machine Allocation (FTVMA) (Keerthika et al., 2019) methods.

4.1 Performance Metrics

The following measures are used to assess the suggested AFTRA method:

- **Makespan:** Makespan is defined as the maximum time taken by any virtual machine to complete all its tasks assigned to it. The minimal makespan helps in an efficient load balancing of all virtual machines. A virtual machine's execution time is dependent on the decision variable D_{ij}:

$$D_{ij} = \begin{cases} 1 & if \ T_i \ is \ allocated \ to \ VM_j \\ 0 & if \ T_i \ is \ not \ allocated \ to \ VM_j \end{cases} \tag{5}$$

Virtual machine execution time is measured as:

$$ET_j = \sum_{i=1}^{n} \frac{MI_i}{MIPS_j} \times D_{ij} \tag{6}$$

where $'n'$ is no. of jobs scheduled to a VM j.

Makespan is calculated as:

$$Makespan = \max\left(ET_j\right), 1 \leq j \leq m \tag{7}$$

where $'m'$ is number of virtual machines.

- **Resource Utilization:** The VM utilization (Resource utilization) of virtual machine j is calculated as:

$$RU_j = \frac{ET_j}{AT_j} \times 100 \tag{8}$$

The average resource or VM utilization is calculated as:

$$ARU = \frac{\sum_{j=1}^{n} RU_j}{n} \tag{9}$$

where $'n'$ is number of virtual machines.

- **Success Rate:** The efficacy of the fault tolerance approach is calculated using the success rate. It is a significant performance metric which is used to assess the system reliability in the cloud. This is the ratio between the amount of tasks completed and the time limit given by users. It is calculated as follows:

$$Sucess\ rate = \frac{T_c}{T_s} \times 100 \tag{10}$$

- **Throughput:** Throughput refers the total tasks executed by a virtual machine in a unit of time:

$$Throughput = \frac{Number\ of\ tasks\ successfully\ executed}{Specified\ period\ of\ time} \tag{11}$$

Throughput indicates the overall system performance or the productivity of a cloud environment i.e. higher throughput indicates higher performance of the system.

4.2 Simulation Environment 1

Under this simulation environment, the total tasks considered to be 500 (Keshk, El-Sisi, & Tawfeek, 2014; W.-J. Wang, Chang, Lo, & Lee, 2013; W. Wang, Zeng, Tang, & Yao, 2012) and total virtual machines varied from 25 to 200 with intervals of 25. The memory size of the virtual machines is assigned randomly which is varied from 512- 2048 KB. The length of the tasks is represented using Million Instructions (MI) which is varied from 1000 to 5000 MI. The virtual machine speed is represented using Million Instructions Per Second (MIPS) which is varied from 500 to 1000 MIPS. The bandwidth of the virtual machines is represented using bits per second (bps) which is varied from 5000 to 10000 bps.

The usefulness of the proposed AFTRA strategy is evaluated in this scenario. Figure 4 illustrates the outcomes based on the makespan. As the number of virtual machine grows, the makespan is decreased for all strategies. The makespan is minimized in the proposed method, as the submitted tasks are executed without execution failure and rescheduling. Figure 5 shows the comparison based on the VM utilization. It obviously shows that the proposed method has higher VM utilization than the other methods due to the balanced load at the virtual machines. Figure 6 shows the comparison based on the success rate. The proposed method has higher success rate because it allows an immense opportunity for tasks to be completed within the deadline. Figure 7 shows the comparative results based on the throughput. Since the proposed method effectively considers the failure rate of the resources, it has a higher throughput.

4.3 Simulation Environment 2

In this scenario, the number of virtual machines is fixed as 100 and the number of tasks is varied from 100 to 1000 with intervals of 100 (Wang et al., 2012; Wang et al., 2013; Keshk et al., 2014). The
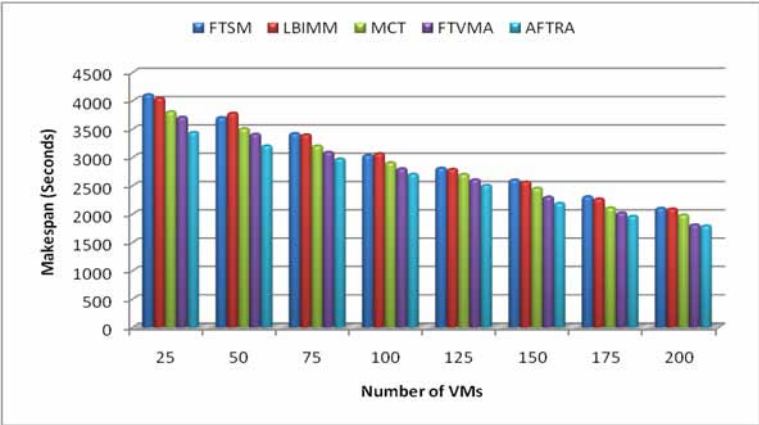
Figure 4. Makespan Comparison under Scenario 1

Figure 5. VM utilization Comparison under Scenario 1
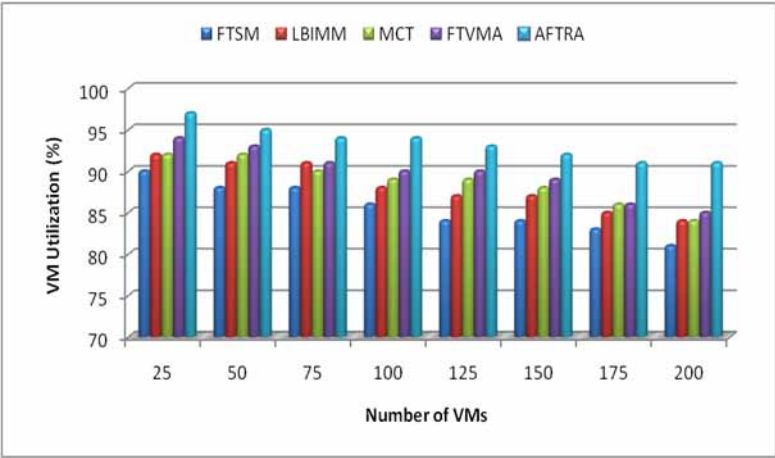

Figure 6. Success Rate Comparison under Scenario 1
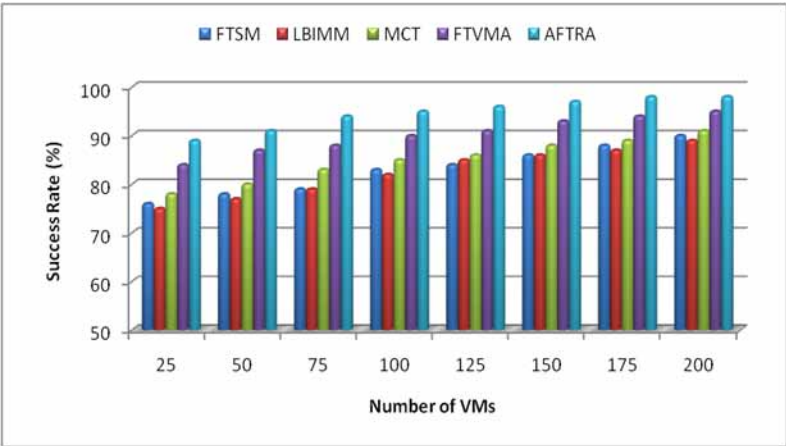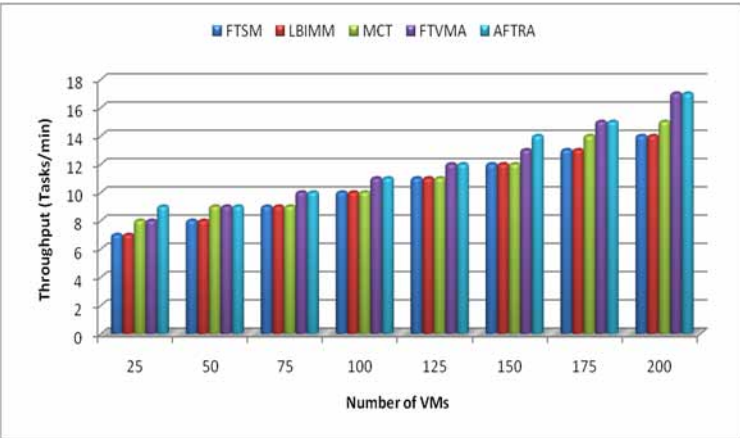
Figure 7. Throughput Comparison under Scenario 1

length of the tasks, speed, memory capacity and bandwidth of the virtual machines are considered as mentioned in experiment 1. Figure 8 shows the Makespan comparison; Figure 9 shows the VM utilization comparison; Figure 10 shows the Success rate comparison; and Figure 11 shows the throughput comparison in which the proposed method efficiently reduces the makespan due to the effective load balancing scheme. Also, it increases the VM utilization, success rate and throughput because it effectively considers the probability of failures of the resources.

5. CONCLUSION

An efficient fault tolerant system is implemented to support cloud computing with high level of fault tolerance, in order to effectively utilize the cloud computing resources. This method classifies the resources based on the load factors into under, heavily and normally loaded and calculates the probability of resources failure, and then allocates the resources according to the need of the tasks. The suggested strategy is analyzed under two different scenarios, and the outcomes are compared with other recent benchmark strategies. The experimental results proved that the proposed strategy provides

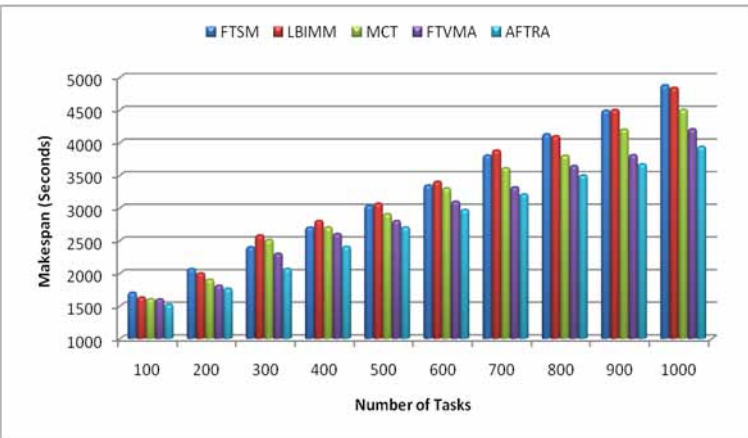**Figure 8. Makespan Comparison under Scenario 2**

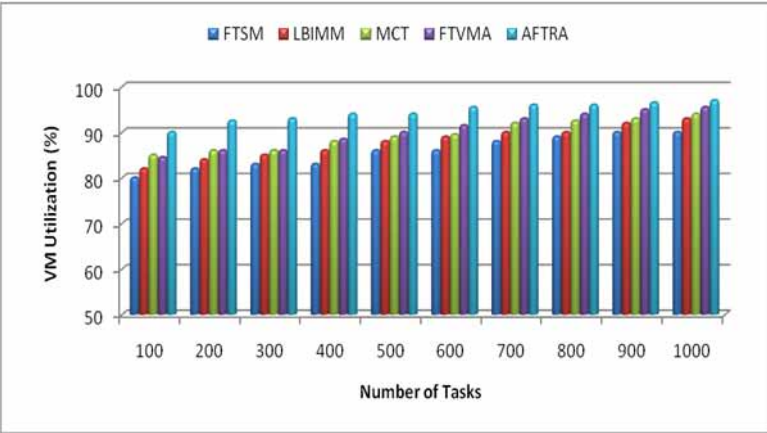**Figure 9. VM Utilization Comparison under Scenario 2**

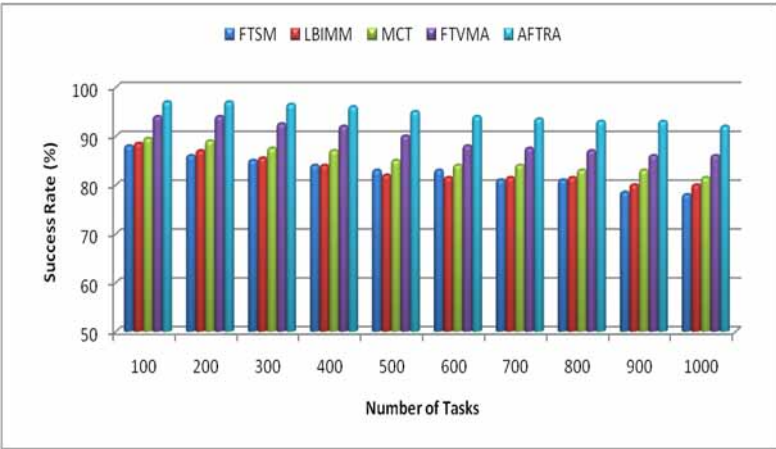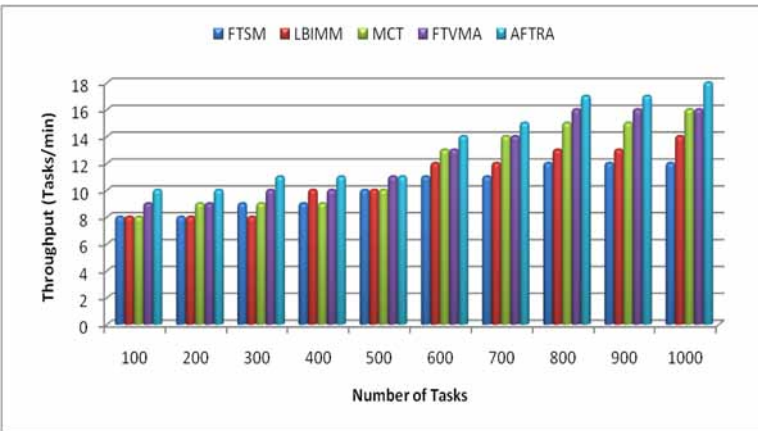**Figure 10. Success Rate Comparison under Scenario 2**

Figure 11. Throughput Comparison under Scenario 2

better performance over the other algorithms based on performance measures makespan, throughput, success rate and VM utilization. The foremost contribution of the suggested method relies on reducing makespan and improving the throughput and the reliability of the cloud environment. Despite promising, this research can be extended by applying optimization algorithms to boost its performance further by considering various parameters such as cost for execution, security, transmission cost and energy consumption. These above considerations can help in further improvement in the performance of the resource allocation in cloud.

REFERENCES

Abderrahim, W., & Choukair, Z. (2018). Brokerage-based dependability integration in cloud computing services. *The Journal of Supercomputing*, *74*(7), 3359–3387. doi:10.1007/s11227-018-2388-4

Afzal, S., & Kavitha, G. (2019). Load balancing in cloud computing–A hierarchical taxonomical classification. *Journal of Cloud Computing*, *8*(1), 22. doi:10.1186/s13677-019-0146-7

Alarifi, A., Abdelsamie, F., & Amoon, M. (2019). A fault-tolerant aware scheduling method for fog-cloud environments. *PLoS One*, *14*(10), e0223902. doi:10.1371/journal.pone.0223902 PMID:31622419

Amoon, M., & Tobely, T. E. E. (2019). A green energy-efficient scheduler for cloud data centers. *J Cluster Computing*, *22*(2), 3247–3259. doi:10.1007/s10586-018-2028-z

Belalem, G., & Limam, S. (2011). Fault tolerant architecture to cloud computing using adaptive checkpoint. *International Journal of Cloud Applications and Computing*, *1*(4), 60–69. doi:10.4018/ijcac.2011100105

Chen, H., Wang, F., Helian, N., & Akanmu, G. (2013). *User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing.* Paper presented at the 2013 national conference on parallel computing technologies (PARCOMPTECH). doi:10.1109/ParCompTech.2013.6621389

Chou, T.-H. (2019). Exploring Relationship Quality of User's Cloud Service: The Case Study of SaaS CRM. *Journal of Organizational and End User Computing*, *31*(3), 17–36. doi:10.4018/JOEUC.2019070102

Garraghan, P., Townend, P., & Xu, J. (2014). *An empirical failure-analysis of a large-scale cloud computing environment.* Paper presented at the 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering. doi:10.1109/HASE.2014.24

Geng, X., Mao, Y., Xiong, M., & Liu, Y. (2019). An improved task scheduling algorithm for scientific workflow in cloud computing environment. *Cluster Computing*, *22*(3), 7539–7548. doi:10.1007/s10586-018-1856-1

Goiri, Í., Julia, F., Guitart, J., & Torres, J. (2010). *Checkpoint-based fault-tolerant infrastructure for virtualized service providers.* Paper presented at the 2010 IEEE Network Operations and Management Symposium-NOMS 2010. doi:10.1109/NOMS.2010.5488493

Gupta, R. K., & Pateriya, R. K. (2017). Balance Resource Utilization (BRU) Approach for the Dynamic Load Balancing in Cloud Environment by using AR Prediction Model. *Journal of Organizational and End User Computing*, *29*(4), 24–50. doi:10.4018/JOEUC.2017100102

Gupta, V., Kaur, B. P., & Jangra, S. (2019). An efficient method for fault tolerance in cloud environment using encryption and classification. *J Soft Computing*, *23*(24), 13591–13602. doi:10.1007/s00500-019-03896-6

Hicham, B., Said, B., Touhafi, A., & Ezzati, A. (2018). *Deadline and Energy Aware Task Scheduling in Cloud Computing.* Paper presented at the 2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech).

Jyoti, A., Shrimali, M., Tiwari, S., & Singh, H. P. (2020). Cloud computing using load balancing and service broker policy for IT service: A taxonomy and survey. *Journal of Ambient Intelligence and Humanized Computing*, *11*(11), 1–30. doi:10.1007/s12652-020-01747-z

Kathpal, C., & Garg, R. (2019). Survey on Fault-Tolerance-Aware Scheduling in Cloud Computing. In *Information and Communication Technology for Competitive Strategies* (pp. 275–283). Springer. doi:10.1007/978-981-13-0586-3_28

Keerthika, P., & Suresh, P. (2015). A Hierarchical Load Balanced Fault tolerant Grid Scheduling Algorithm with User Satisfaction. *WSEAS Transactions on Computers*, *14*, 15–28.

Keerthika, P., & Suresh, P. (2015). A multiconstrained grid scheduling algorithm with load balancing and fault tolerance. *TheScientificWorldJournal*, *2015*, 2015. doi:10.1155/2015/349576 PMID:26161438

Keerthika, P., Suresh, P., Manjula Devi, R., Sangeetha, M., & Sagana, C. (2019). Design of a Fault Tolerant Strategy for Resource Scheduling in Cloud Environment. *International Journal of Engineering and Advanced Technology*, *9*(1), 5121–5128. doi:10.35940/ijeat.A1519.109119

Keshk, A. E., El-Sisi, A. B., & Tawfeek, M. A. (2014). Cloud task scheduling for load balancing based on intelligent strategy. *International Journal of Intelligent Systems and Applications*, *6*(5), 25–36. doi:10.5815/ijisa.2014.05.02

Kumari, P., & Kaur, P. (2018). A survey of fault tolerance in cloud computing. *Journal of King Saud University-Computer Information Sciences*. 10.1016/j.jksuci.2018.09.021

Latiff, A., & Shafie, M. (2017). A checkpointed league championship algorithm-based cloud scheduling scheme with secure fault tolerance responsiveness. *J Applied Soft Computing*, *61*, 670–680. doi:10.1016/j.asoc.2017.08.048

Li, J., Feng, L., & Fang, S. (2014). An greedy-based job scheduling algorithm in cloud computing. *Journal of Software*, *9*(4), 921–925. doi:10.4304/jsw.9.4.921-925

Manimala, R., & Suresh, P. (2013). *Load balanced job scheduling approach for grid environment.* Paper presented at the 2013 International Conference on Information Communication and Embedded Systems (ICICES). doi:10.1109/ICICES.2013.6508305

Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: A big picture. *Journal of King Saud University-Computer Information Sciences*, *32*(2), 149–158. doi:10.1016/j.jksuci.2018.01.003

Parvez, A. S., Robel, M. R. A., Rouf, M. A., Podder, P., & Bharati, S. (2019). *Effect of Fault Tolerance in the Field of Cloud Computing.* Paper presented at the International Conference on Inventive Computation Technologies.

Prathiba, S., & Sowvarnica, S. (2017). *Survey of failures and fault tolerance in cloud.* Paper presented at the 2017 2nd International Conference on Computing and Communications Technologies (ICCCT). doi:10.1109/ICCCT2.2017.7972271

Ren, S., & van der Schaar, M. (2013). Dynamic scheduling and pricing in wireless cloud computing. *IEEE Transactions on Mobile Computing*, *13*(10), 2283–2292.

Suresh, P., & Balasubramanie, P. (2012). User demand aware scheduling algorithm for data intensive tasks in grid environment. *European Journal of Scientific Research*, *74*(4), 609–616.

Suresh, P., & Balasubramanie, P. (2013). User demand aware grid scheduling model with hierarchical load balancing. *Mathematical Problems in Engineering*, *2013*, 1–8. Advance online publication. doi:10.1155/2013/439362

Suresh, P., Balasubramanie, P., & Keerthika, P. (2011). Prioritized user demand approach for scheduling meta tasks on heterogeneous grid environment. *International Journal of Computers and Applications*, *23*(1), 6–12. doi:10.5120/2856-3670

Vella, E., Yang, L., Anwar, N., & Jin, N. (2018). *Adoption of cloud computing in hotel industry as emerging services.* Paper presented at the International Conference on Information. doi:10.1007/978-3-319-78105-1_26

Wang, W., Zeng, G., Tang, D., & Yao, J. (2012). Cloud-DLS: Dynamic trusted scheduling for Cloud computing. *Expert Systems with Applications*, *39*(3), 2321–2329. doi:10.1016/j.eswa.2011.08.048

Wang, W.-J., Chang, Y.-S., Lo, W.-T., & Lee, Y.-K. (2013). Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments. *The Journal of Supercomputing*, *66*(2), 783–811. doi:10.1007/s11227-013-0890-2

V. Sathiyamoorthi (PhD) is currently working as an Associate Professor in Computer Science and Engineering Department at Sona College of Technology, Salem, Tamil Nadu, India. He was born on June 21, 1983, at Omalur in Salem District, Tamil Nadu, India. He received his Bachelor of Engineering degree in Information Technology from Periyar University, Salem with First Class. He obtained his Master of Engineering degree in Computer Science and Engineering from Anna University, Chennai with Distinction and secured 30th University Rank.He received his Ph.D degree from Anna University, Chennai in Web Mining. His areas of specialization include Web Usage Mining, Data Structures, Design and Analysis of Algorithm and Operating System. He has published many papers in International Journals and conferences. He has published many books and book chapters in various renowned international publishers. He has also participated in various National level Workshops and Seminars conducted by various reputed institutions.

P. Keerthika (PhD) is currently working as Senior Assistant Professor in the Department of Computer Science and Engineering, Kongu Engineering College, Perundurai, Erode. She has a work experience of 14 years in teaching. She has completed her Ph.D in Anna University, Chennai. She is contributing towards research in the area of Grid and Cloud Computing for the past 9 years. She has published around 20 research papers in reputed journals indexed by Scopus and SCI and presented 15 papers in National and International Conferences. Her areas of interest are Machine Learning, Cloud Computing and IoT. She is one of the recognized Supervisors in the Faculty of Information and Communication Engineering under Anna University, Chennai. She is currently guiding Ph.D scholars in the areas of Machine Learning and Cloud. She has received grants from various funding agencies like DRDO and DST. She is currently acting as Reviewer and Editorial Board member in reputed journals.

P. Suresh (PhD) is currently working as Senior Assistant Professor in the Department of Information Technology, Kongu Engineering College, Perundurai, Erode. He has a work experience of 14 years in teaching. He has completed his Ph.D in Anna University, Chennai. He is contributing towards research in the area of Grid and Cloud Computing for the past 10 years. He has published 25 research papers in reputed journals indexed by Scopus and SCI and presented 25 papers in National and International Conferences. His areas of interest are Networks, Cloud Computing and Big Data. Currently he is working in the areas of Machine Learning and IoT. He is one of the recognized Supervisors in the Faculty of Information and Communication Engineering under Anna University, Chennai. He is currently guiding Ph.D scholars in the areas of Machine Learning and Cloud. He has received grants from various funding agencies like DRDO, ICMR, and CSIR towards the conduct of seminars and workshops. He has received Best Faculty Award from Kongu Engineering College. He is currently acting as Reviewer and Editorial Board member in reputed journals.

Justin Zhang is a faculty member in the Department of Management at University of North Florida. He received his Ph.D. in Business Administration with a concentration on Management Science and Information Systems from Pennsylvania State University, University Park. His research interests include economics of information systems, knowledge management, electronic business, business process management, information security, and social networking. He has published research articles in various scholarly journals, books, and conference proceedings. He is the editor-in-chief of the Journal of Global Information Management. He also serves as an associate editor and an editorial board member for several other journals.

A. Prasanth Rao (PhD) is currently working as a Professor, in the Dept. of Information Technology, Anurag group of institutions, Hyderabad. Total 15 years of Experience in teaching. Completed Doctorate from JNTU-H, Hyderabad. Research domains are IOT, Cloud Computing and Information Security.

K. Logeswaran received his M.E. degree in computer science and engineering from Anna university, Chennai, India, in 2011 and B.Tech degree in information technology from Anna university, Chennai, India in 2009. He is presently pursuing his Ph.D. in Information and Communication Engineering under Anna University, Chennai. He is currently working as Assistant Professor in the department of Information Technology at Kongu Engineering College, Perundurai, India. His research interests are in the areas of Data mining, Knowledge discovery and Networking. He has authored over 20 research papers in refereed journals and international conferences. He has received grants from various funding agencies like DRDO, SERB, and CSIR towards the conduct of seminars and workshops.