*I.J. Intelligent Systems and Applications,* **2022, 3, 18-26**
Published Online June 2022 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2022.03.02

# A Fuzzy Approach to Fault Tolerant in Cloud using the Checkpoint Migration Technique

**Noshin Hagshenas**
Department of Computer Engineering, Liyan Institute of Education, Bushehr, Iran
E-mail: nooshinhsh@gmail.com

**Musa Mojarad***
Department of Computer Engineering, Firoozabad Branch, Islamic Azad University, Firoozabad, Iran
E-mail: m.mojarad@iauf.ac.ir

**Hassan Arfaeinia**
Department of Computer Engineering, Liyan Institute of Education, Bushehr, Iran
E-mail: harfaeinia@gmail.com

Received: 10 October 2021; Accepted: 24 December 2021; Published: 08 June 2022

**Abstract:** Fault tolerance is one of the most important issues in cloud computing to provide reliable services. It is difficult to implement due to dynamic service infrastructures, complex configurations and different dependencies. Extensive research efforts have been made to implement fault tolerance in the cloud environment. Many studies focus only on fault detection and do not consider fault tolerance. For this reason, in this paper, in addition to recognizing the nature of the fault, a fuzzy logic-based approach is proposed to provide an appropriate response and increase the fault tolerance in the cloud environment. Checkpoint-based migration technique is used to increase fault tolerance. Using a checkpoint during migration can reduce time and processing costs and balance the load between virtual machines in the event of a fault. The simulation is performed according to the data center of Vietnam Telecommunications Company (VDC). The results of the proposed method in a period of 60 minutes show 98.03% fault detection accuracy, which is 4.5% and 4.1% superior to FLPT and PLBFT algorithms, respectively.

**Index Terms:** Cloud environment, fuzzy approach, fault tolerance, fault detection, migration technique, checkpoint.

## 1. Introduction

Today's world is the age of cloud computing. The purpose of cloud computing is to empower data centers as a network of virtual services, such as hardware, database, interface or user interface, etc. [1]. In real-time cloud applications, computing is performed on remote nodes. Cloud computing sometimes results in a chance of fault due to poor control over the computing nodes. This system must be very reliable, so the demand for fault tolerance increases to achieve reliability for real-time computing on cloud infrastructure. Extensive use of cloud computing resources has led to the emergence of cloud environment reliability as an important issue [2].

Fault tolerance in cloud computing is about designing a way to keep tasks and programs running whenever multiple components are down or unavailable. These algorithms balance the entire system load. This helps companies assess their infrastructure needs and provide services when related devices are not available. This does not mean that fault tolerance methods can provide 100% full service, but this concept keeps the system usable and, most importantly, at a good level of performance [3].

In general, fault tolerance refers to the ability of a system (computer, network, cloud cluster, etc.) to continue working without interruption in the event of a failure of one or more components. The purpose of creating a fault tolerant system is to prevent disturbances caused by a breakdown point, to ensure the availability and continuity of important applications or systems [4]. Fault-tolerant systems use backup components that automatically replace faulty components and increase system reliability.

The ability of the operating system to recover and tolerate errors flawlessly can be provided using hardware, software or a combination solution that ensures load balance. Some computer systems use multiple duplicate fault tolerant systems for easy error handling. Fault tolerant methods can play a role in the failure recovery strategy. For example, fault-tolerant systems with backup components in the cloud can recover important systems quickly. Tolerance is an important issue in cloud computing, as it can guarantee features such as performance and reliability.

Tolerance is an important issue in cloud computing, as it can guarantee features such as performance and reliability.

 

Types of faults are divided into three categories of virtual machines, request and execution time according to different situations. Many studies focus only on fault detection and do not consider improving fault tolerance. Therefore, in this paper, in addition to recognizing the nature of the fault, a fuzzy logic-based approach is proposed to provide an appropriate response and increase the fault tolerance.

In this method, in order to increase the fault tolerance and load balance in the event of a fault, the migration technique through the checkpoint has been used. The migration technique overlaps over time, so using a checkpoint can largely avoid re-runs and scheduling. Due to the possibility of fault during the operation of the fault tolerance system, in each time period, fault detection is applied simultaneously with the fault tolerance.

The continuation of the paper is as follows. Section 2 reviews related work. The proposed model for fault detection and tolerance is described in Section 3. In Section 4, the simulation results are presented and finally in Section 5, conclusions and future suggestions are presented.

## 2. Literature Review

So far, a variety of methods have been proposed to detect and tolerate faults in cloud computing. Here are some of the latest researches.

In [5], an approach to increase the fault tolerance in cloud computing using fuzzy logic is proposed. The results of this study show that this algorithm provides better performance in terms of accuracy and faster detection for supercritical systems. The results of the experiments indicate that the accuracy criterion is 6.49% superior to the ABFT method and 2.27% superior to the FFD method.

In [6], improved tolerability and fault detection in cloud computing based on two fuzzy systems were proposed. In this method, to increase the fault tolerance and load balance in the event of a fault, the techniques of re-requesting the task and migrating through the checkpoint are used. The results of this method in a period of 60 minutes show 98% fault detection accuracy, which is 4% superior to the FLPT algorithm on average.

In [7], a fuzzy clustering method was proposed to identify the fault location and reduce energy consumption in cloud networks. This paper investigates the fault in these environments using fuzzy logic and ppaper swarm optimization (PSO). Using fuzzy logic, first a clustering was applied on the network nodes and the nodes were divided into different clusters. Then, with PSO, the clusters were optimized for more efficiency.

In [8], a new method is proposed to increase the fault tolerance in cloud computing using fuzzy logic. The authors propose a detailed analysis of the nature of the fault and its diagnosis, as well as a fuzzy-based method for preparing an appropriate response to the fault tolerance. Here, in order to increase the fault tolerance and load balance in the event of a fault, re-execution and migration request techniques are used.

In [9], fault tolerance is performed using a load balancing schedule on an application of IoT. The main idea of this method is to classify different modules along with calculating energy consumption and finding the minimum energy consumption. To distribute the modules among IoT devices, this paper presents an efficient productivity checkpoint and load balancing technique based on the Bayesian classification called ECLB.

In [10], a multi-cloud fault tolerant architecture for managing transient servers in cloud computing called MULTS is proposed. This architecture can provide a flexible environment by using an optimal scenario-based checkpoint in a design to ensure running processes while reducing user costs. The simulation shows a high level of accuracy for MULTS.

In [11], the FFD algorithm was proposed which is a method for fault tolerance in cloud computing based on fuzzy logic. In this system, there are two types of computational nodes, including physical servers and computing servers, which are managed through fault detection and monitoring parameters for fault detection. The monitoring component uses several network parameters such as response time, throughput, bandwidth and response rate, and the fault detection component detects the fault through these parameters and using a fuzzy method.

In [12], the FLPT algorithm is proposed which, based on a detailed analysis of the nature of the fault, proposes a technique for the rapid detection of faults in the IaaS cloud system. By combining fuzzy logic and prediction techniques, this method can perform better in terms of accuracy and speed in fault detection and thus increase system reliability. The faults simulated in this method include four categories of resource usage, throughput, response time and bandwidth.

In [13], a reactive approach with a flexible inspection distance for fault tolerance in cloud computing systems is proposed. Checkpoint is one of the most popular reaction fault tolerance techniques used in distributive calculations. However, this can create significant overhead costs that apply to the checkpoint distance, and these high costs reduce cloud performance. In this paper, the reactive fault tolerance approach in the field of inspection is proposed and evaluated with the aim of achieving better performance. This method depends on using a flexible distance from the checkpoint to reduce additional costs.

## 3. Proposed Method

The proposed approach uses checkpoint-based migration to increase fault tolerance through fuzzy logic. Fig. 1 shows the architecture of the proposed method. In the proposed system, the *R* request (task) is entered by the users into

 

the cloud data center. Tasks are first assigned to physical servers by the cloud scheduling system. The existing $N$ physical servers then perform tasks on virtual machines based on their internal scheduling management and return the results to users. In this model, only faults due to virtual machines and runtime are considered. In this paper, faults are entered into the system randomly and at specific time periods. In the proposed architecture, the monitoring component is used to extract the required parameters for fuzzy systems in any given time period, i.e., $(t_k - t_{k-1})$. This component first extracts the input parameters of the fuzzy fault detection system and then the fuzzy fault detection system predicts the presence or absence of fault based on these parameters. In the event of a fault, the monitoring component again extracts the parameters required by the fuzzy fault tolerance system. Finally, the fault tolerance system, based on the built-in fault tolerance techniques, sends the appropriate response to the load data center to load balancing.

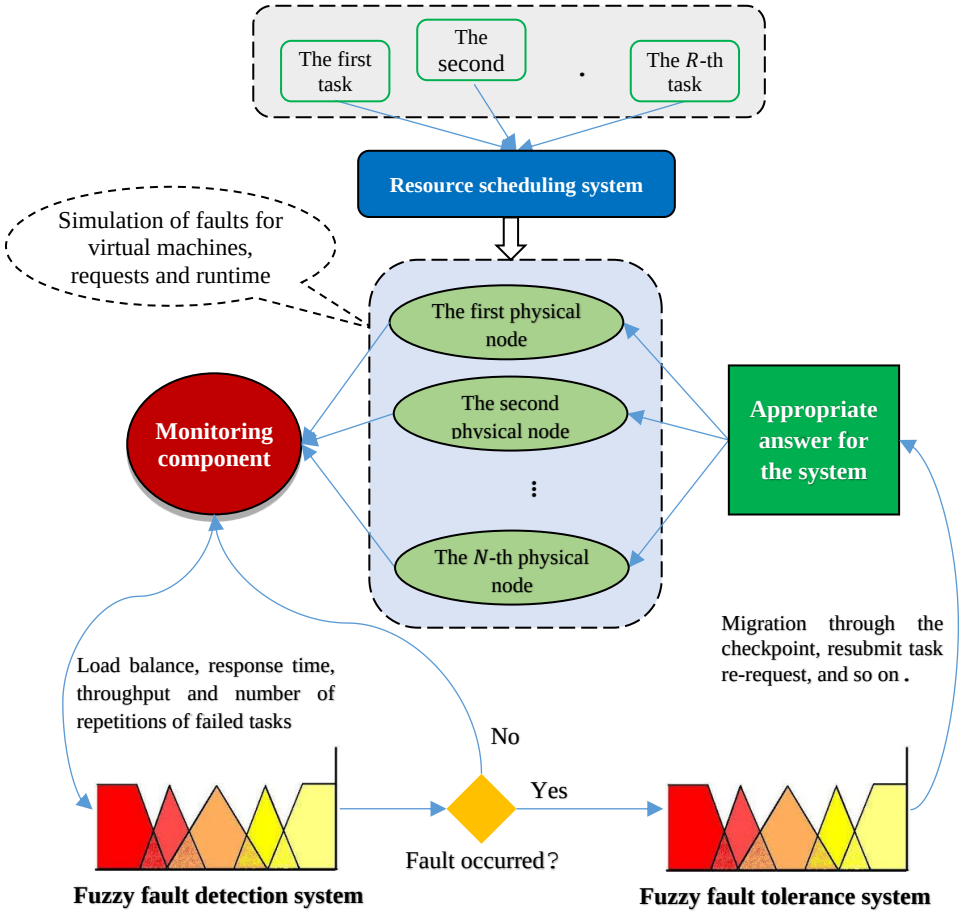


Fig.1. Proposed method architecture

The proposed model uses two fuzzy systems for fault detection and tolerance, respectively. A fuzzy-based system is used to accurately identify the source of the fault (failure or malfunction), and a fuzzy-based system is used to provide an appropriate response to the predicted fault (fault tolerance system). The fuzzy fault detection system uses the parameters of response time, load density, and throughput to detect, and the fuzzy system uses fault tolerance by the parameters of system status, task waiting time, number of failed repetitions, and server throughput. Reduces the effects of select fault. All parameters are expressed in fuzzy and the system output is determined based on the fuzzy rules database.

The output of the fault detection system is known as the system status parameter and the output of the fault tolerance system is the appropriate response to the cloud system. In this research, the techniques of requesting re-submission of duty and migration through the checkpoint have been used as a response. In recent years, the migration technique in cloud computing has become one of the major challenges in increasing fault tolerance. Migration overlaps with time, so using the checkpoint avoids re-running and re-starting tasks as much as possible. Due to the computational overhead of this technique, it is important to minimize the migration rate along with the load balancing. Fig. 2 shows the checkpoint architecture to increase fault tolerance in cloud computing.

### *3.1. Fault tolerance techniques*

There are two challenges to using fault tolerance techniques and the fault correction process to use fault tolerance techniques. Based on these two challenges, fault tolerance techniques can be described as follows, where in this paper,

 

"Request a re-submission of task" and "Migration through Checkpoint" are used for fault tolerance. In general, fault tolerance techniques are as follows:

- Repeat task: This technique executes the task on one or more physical servers.
- Repeat Task: This technique re-executes the task on one or more virtual machines in the current physical node.
- Request to resubmit a task: In case of a fault, this technique sends a request to the user and asks him to check the correctness of his task and resubmit it.
- Execution of the task through the checkpoint: In case of fault, this technique executes the task from the point of fault.
- Checkpoint migration: In the event of a fault, this technique transfers task-related data to be executed from the fault point to another physical server.
- Virtual machine migration: In the event of a fault, this technique transfers one or more virtual machines from one physical server to another.

Faults enter the system randomly and possibly at any given time. In this paper, three categories of faults are examined. We use the probable parameter $FR_{VM}$ for the virtual machine fault, $FR_{Request}$ for the task fault, and $FR_{Task}$ for the runtime fault.

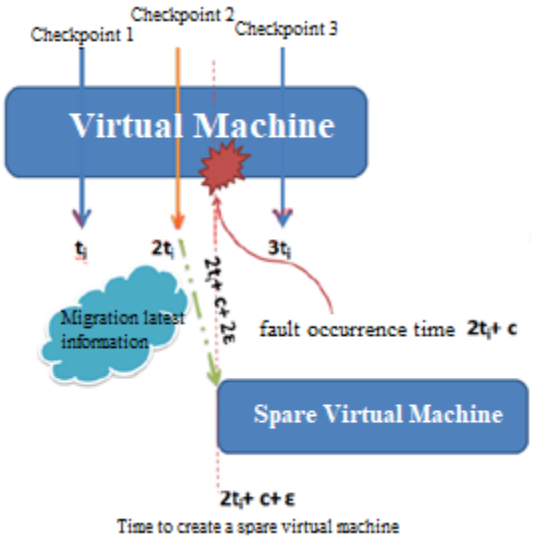


Fig.2. Checkpoint architecture to increase fault tolerance in cloud computing

### *3.2. Proposed fuzzy system*

The proposed fuzzy system structure is similar in fault detection systems as well as fault tolerance. In general, the proposed fuzzy system according to Fig. 3 consists of four main parts: 1) Fuzzification, 2) Knowledge-Base, 3) Inference Engine and 4) Defuzzification.

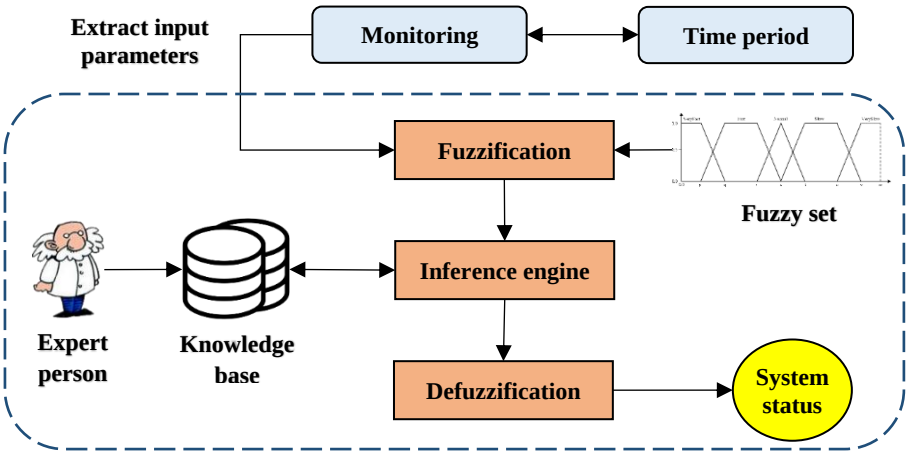


Fig.3. Proposed fuzzy system architecture

### *3.3. Details of fuzzy fault detection system*

The monitoring component collects the required parameters of the fuzzy fault detection system from each physical server. This increases the computational load; however, it can lead to the detection of the source of the fault. Therefore, the monitoring component makes the data related to physical servers available to the fuzzy system in each time period. Fuzzy fault detection system parameters include response time, load balance, throughput and MakeSpan. In this paper, the rules in the knowledge base are created by an expert [14]. Table 1 shows some examples of fuzzy rules used for fuzzy fault detection systems.

Table 1. Some examples of rules related to fuzzy fault detection system

| Rule | | Response time | | Load balance | | Throughput | | MakeSpan | | Output |
|---|---|---|---|---|---|---|---|---|---|---|
| $R_1$ | *If* | *verylow v low* | *and* | *verylow* | *and* | - | *and* | - | *Then* | *verylow* |
| $R_2$ | *If* | *middle* | *and* | *low v middle* | *and* | *veryhigh v high* | *and* | - | *Then* | *low* |
| $R_3$ | *If* | *middle* | *and* | *low* | *and* | *high* | *and* | *middle* | *Then* | *low* |
| $R_4$ | *If* | *high* | *and* | *low* | *and* | *middle v high* | *and* | - | *Then* | *low* |
| $R_5$ | *If* | *middle v high* | *and* | *verylow* | *and* | *middle v high* | *and* | *low v middle* | *Then* | *middle* |
| $R_6$ | *If* | *middle* | *and* | *high* | *and* | *middle* | *and* | *middle* | *Then* | *middle* |
| $R_7$ | *If* | *verylow* | *and* | *high v veryhigh* | *and* | *low v middle* | *and* | *low v middle* | *Then* | *middle* |
| $R_8$ | *If* | *middle* | *and* | *middle v high* | *and* | *high* | *and* | *high* | *Then* | *high* |
| $R_9$ | *If* | *high* | *and* | *middle* | *and* | *veryhigh* | *and* | *middle v high* | *Then* | *high* |
| $R_{10}$ | *If* | *high v veryhigh* | *and* | - | *and* | *veryhigh v high* | *and* | - | *Then* | *veryhigh* |

The output of the fault detection system is veryhigh, high, middle, low and verylow. These terms indicate the status of a physical server to predict fault in it. The output of this system is configured as a system status parameter in three modes "None", "Risk" and "Danger". Hence, the verylow and low terms are mapped to None, the middle term to Risk, and the high and veryhigh terms to Danger. The system status parameter is ultimately used as an input parameter for the fault tolerance system.

### *3.4. Details of fuzzy system fault tolerance*

Based on the output of the fuzzy system, fault detection can identify the source of the fault. This is done by the fault tolerance system. The monitoring component collects the required parameters of the fault tolerance system from the malicious physical server. These parameters include system status (fuzzy fault detection system output), standby time, number of failed iterations, and throughput. In the event of a fault, the monitoring component collects the parameters of the fuzzy system fault tolerance by examining the system status parameter. Depending on the criticality of the system, various fault tolerance techniques are applied to the cloud environment. In this research, "request to resubmit a task" and "Migration through checkpoint" techniques have been used. These techniques are the output of the fuzzy fault tolerance system [14]. Table 2 shows some examples of fuzzy rules used for the fuzzy fault tolerance system.

Table 2. Some examples of rules related to fuzzy fault tolerance system

| Rule | | Node state | | Response time | | Number of unsuccessful repetitions | | Throughput rate | | Output |
|---|---|---|---|---|---|---|---|---|---|---|
| $R_1$ | *If* | *none* | *and* | - | *and* | - | *and* | - | *Then* | *none* |
| $R_2$ | *If* | *risk* | *and* | *verylow v low* | *and* | *low* | *and* | *middle v high* | *Then* | *none* |
| $R_3$ | *If* | *risk* | *and* | *low* | *and* | *high v very high* | *and* | *middle v high* | *Then* | *resubssion* |
| $R_4$ | *If* | *risk* | *and* | *middle v high* | *and* | *low v middle* | *and* | *high* | *Then* | *migration* |
| $R_5$ | *If* | *risk* | *and* | *middle* | *and* | *high* | *and* | *high* | *Then* | *resubssion* |
| $R_6$ | *If* | *danger* | *and* | *middle* | *and* | *middle v high* | *and* | *middle* | *Then* | *resubssion* |
| $R_7$ | *If* | *danger* | *and* | *middle* | *and* | *verylow* | *and* | *veryhigh* | *Then* | *migration* |
| $R_8$ | *If* | *danger* | *and* | *middle* | *and* | *middle* | *and* | *high* | *Then* | *migration v resubssion* |
| $R_9$ | *If* | *danger* | *and* | - | *and* | *veryhigh* | *and* | *middle v high* | *Then* | *resubssion* |
| $R_{10}$ | *If* | *danger* | *and* | *high v veryhigh* | *and* | *veryhigh v high* | *and* | - | *Then* | *migration v resubssion* |

In this system, the output of the task resubmit request is sent to the user via a control message. The user can restart the execution process by modifying the request and resubmitting it. The output of migration is through the checkpoint to perform the tasks that have failed. In this case, the execution buffer information related to the task on the current virtual machine is transferred to another virtual machine and the new virtual machine executes the task from the checkpoint according to the information it receives from the buffer.

 

## 4. Simulation Results

The proposed algorithm is simulated and compared according to the data center of Vietnam Telecommunications Company (VDC). According to VDC data, 75 to 85 virtual machines are available at any one time. There are also 50 physical servers and 50 requests in this data center. Each physical server consists of 1 to 5 virtual machines. Also, the power of virtual machines in the range of 100 to 1000 MIPS is assumed. On the other hand, the size of requests is in the range of 500 to 2000 MIPS. In addition, each request contains 1 to 10 separate tasks. In this paper, the efficiency of the proposed method against FLB [15], FLPT [12] and PLBFT [16] algorithms are compared.

The simulated faults include "virtual machine fault", "request fault" and "runtime fault", which enter the cloud system randomly with a probability of 0.3 [17]. The cloud system is evaluated by evaluation criteria in each period of 5 minutes and the total simulation time is 60 minutes. MATLAB version 2019a software was used for simulations and comparisons [18].

First, the simulation results are compared with other methods to measure the accuracy of fault detection. Fig. 4 shows the results of this comparison. The results of this comparison show the significant superiority of the proposed method in fault detection accuracy over other algorithms. This advantage is greater than the FLB algorithm and is reported to be 11.3% on average. The superiority of the proposed method over FLPT and PLBFT algorithms has been reported to be about 4.5% and 4.1% in different periods, respectively. The best accuracy reported by the proposed method after the simulation period is 98.03%.

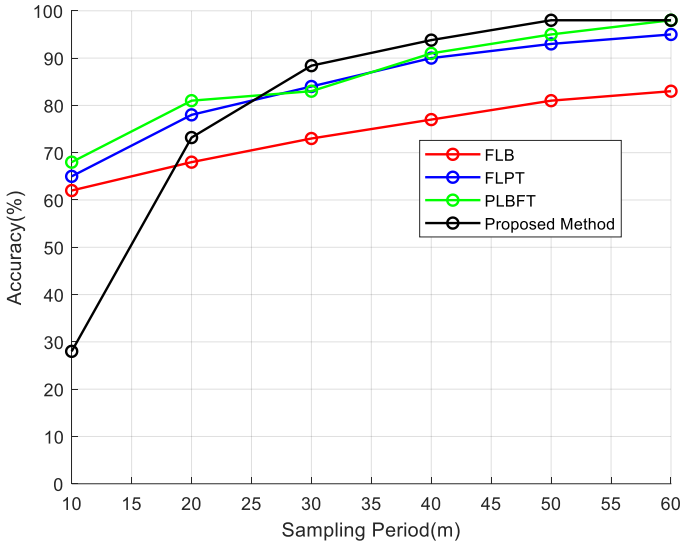


Fig.4. Comparison based on fault detection accuracy criteria

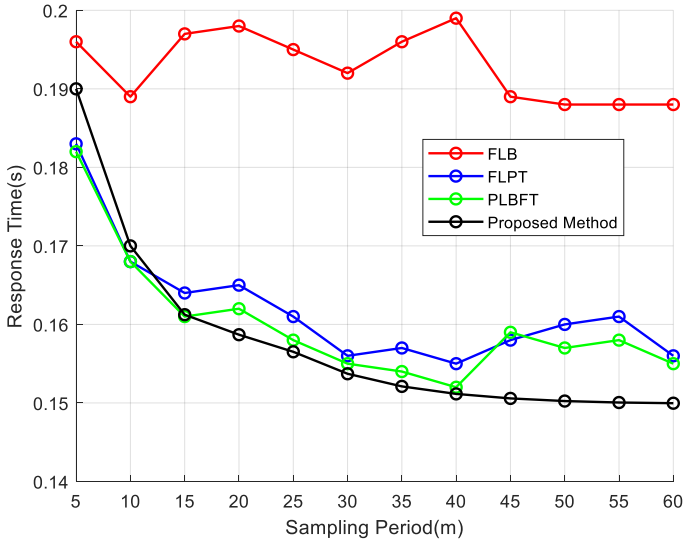


Fig.5. Comparison based on response time criteria

In another experiment, different algorithms were compared based on the response time (response rate) criterion in fault detection. The results of this comparison are shown in Fig. 5. The results of this simulation show that the proposed method is much superior to the FLB algorithm and on average this advantage is 12.0%. Also, the superiority of the proposed method over FLPT and PLBFT algorithms is about 3.5% and 3.1%, respectively. However, the results show that the response time rate fluctuates slightly.

In the last experiment, the algorithms were compared according to the MakeSpan criterion. Fig. 6 shows the results of this comparison for the number of different requests. The results of this comparison also show the better performance of the proposed method. On average, with the number of different requests, the proposed method provides a better MakeSpan criterion and has an 18.3% advantage over the FLB algorithm. This advantage is about 5.6% and 4.2% for FLPT and PLBFT algorithms.

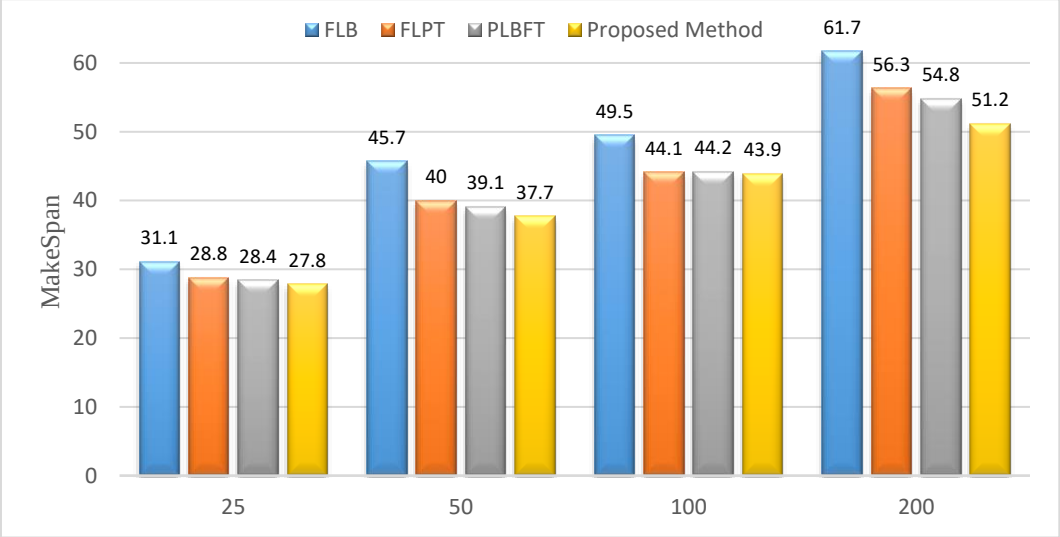


Fig.6. Comparison of results based on MakeSpan criteria

In the event of a virtual machine failure, the application of migration techniques can lead to better system load balancing. This process ultimately improves load balance by shifting tasks between virtual machines. Fig. 7 reports the load status of virtual machines over total time periods. In general, the load of each virtual machine is calculated based on the volume of requests allocated to it in relation to the total volume of requests. The results clearly show small fluctuations in the graph, where the load of all virtual machines varies in the range of 0.01 to 0.03 after the simulation cycle is completed. These results are positive on the proper performance of the proposed method in system load balancing.

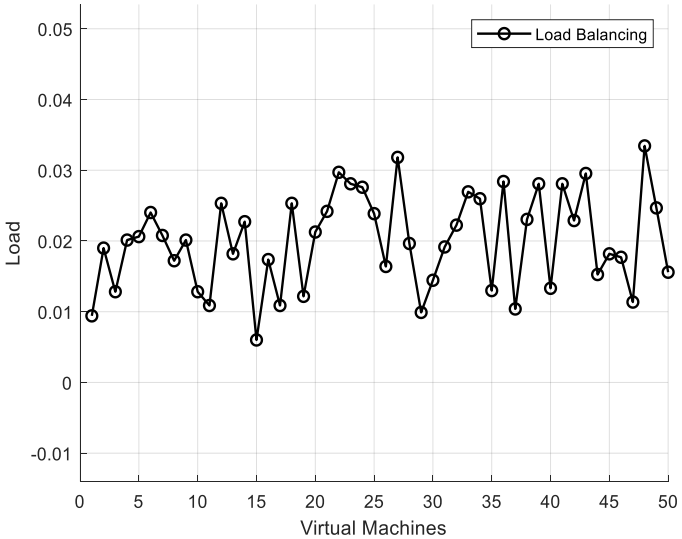


Fig.7. Investigation of load balance of virtual machines after the simulation period

## 5. Conclusion

Cloud computing has revolutionized the IT delivery model from a product to a service. Fault tolerance is difficult due to the dynamic service infrastructure, complex configurations, and different dependencies that exist in the cloud. Fault tolerance refers to tasks that must be performed even when an fault occurs. As the demand for services increases, so does the likelihood of fault. To reduce the impact of fault, many fault tolerance techniques have been designed and proposed. In this paper, a fuzzy fault tolerance system is proposed for fault tolerance in real-time cloud environment with higher reliability and availability. In the proposed model, in addition to recognizing the nature of the fault, a fuzzy logic-based approach is proposed to provide an appropriate response and increase the fault tolerance in the cloud environment. The proposed model uses two fuzzy systems for fault detection and tolerance, respectively. Using this model, fault detection can be performed and thus the reliability of the system is significantly improved. In addition, due to the speed of reaction, the cost of paying for this model is quite acceptable. In the fuzzy system, various parameters such as response time, load balance, throughput and MakeSpan are used to detect the fault. However, system performance can be measured by other parameters such as available resources, latency, bandwidth and productivity.

## References


[1] Emesowum, H., Paraskelidis, A., & Adda, M. (2017). Fault tolerance improvement for cloud data center. *Journal of Communications*, 12(7), 412-418.

[2] Li, Z., Liu, L., & Tong, Z. (2017). Study on Fault Tolerance Method in Cloud Platform based on Workload Consolidation Model of Virtual Machine. *Journal of Engineering Science & Technology Review*, 10(5).

[3] Prathamesh Churi, N. T. Rao, " Teaching Cyber Security Course in the Classrooms of NMIMS University ", International Journal of Modern Education and Computer Science(IJMECS), Vol.13, No.4, pp. 1-15, 2021.DOI: 10.5815/ijmecs.2021.04.01

[4] Karthikeyan, L., Vijayakumaran, C., Chitra, S., & Arumugam, S. (2021). Saldeft: Self-adaptive learning differential evolution based optimal physical machine selection for fault tolerance problem in cloud. *Wireless Personal Communications*, 1-28.

[5] Nazari Cheraghlou, M., Khadem-Zadeh, A., & Haghparast, M. (2018). A framework for optimal fault tolerance protocol selection using fuzzy logic on IoT sensor layer. *International Journal of Information & Communication Technology Research*, 10(2), 19-32.

[6] Moghtadaeipour, A., & Tavoli, R. (2015, November). A new approach to improve load balancing for increasing fault tolerance and decreasing energy consumption in cloud computing. In *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI*) (pp. 982-987). IEEE.

[7] Velde, V., & Rama, B. (2017, June). An advanced algorithm for load balancing in cloud computing using fuzzy technique. In *2017 International Conference on Intelligent Computing and Control Systems* (ICICCS) (pp. 1042-1047). IEEE.

[8] Rezaeipanah, A., Mojarad, M., & Fakhari, A. (2020). Providing a new approach to increase fault tolerance in cloud computing using fuzzy logic. *International Journal of Computers and Applications*, 1-9.

[9] Sharif, A., Nickray, M., & Shahidinejad, A. (2020). Fault-tolerant with load balancing scheduling in a fog-based IoT application. *IET Communications*, 14(16), 2646-2657.

[10] Neto, J. P. A., Pianto, D. M., & Ralha, C. G. (2019). MULTS: A multi-cloud fault-tolerant architecture to manage transient servers in cloud computing. *Journal of Systems Architecture*, 101, 101651.

[11] Bui, D. M., & Lee, S. (2016). Fuzzy Fault Detection in IaaS Cloud Computing. In *International Conference on Ubiquitous Information Management and Communication*. p. 65. ACM.

[12] Bui, D. M., & Lee, S. (2018). Early fault detection in IaaS cloud computing based on fuzzy logic and prediction technique. *The Journal of Supercomput*ing, 74(11), 5730-5745.

[13] Amoon, M., El-Bahnasawy, N., Sadi, S., & Wagdi, M. (2019). On the design of reactive approach with flexible checkpoint interval to tolerate faults in cloud computing systems. *Journal of Ambient Intelligence and Humanized Computing*, 10(11), 4567-4577.

[14] Cheraghlou, M. N., Khademzadeh, A., & Haghparast, M. (2019). New Fuzzy-Based Fault Tolerance Evaluation Framework for Cloud Computing. *Journal of Network and Systems Management*, 1-19.

[15] Arabnejad, H., Pahl, C., Estrada, G., Samir, A., & Fowley, F. (2017, September). A fuzzy load balancer for adaptive fault tolerance management in cloud platforms. In *European Conference on Service-Oriented and Cloud Computing* (pp. 109-124). Springer, Cham.

[16] Attallah, S. M., Fayek, M. B., Nassar, S. M., & Hemayed, E. E. (2021). Proactive load balancing fault tolerance algorithm in cloud computing. *Concurrency and Computation: Practice and Experience*, 33(10), e6172.

[17] Tran Son Hai, Le Hoang Thai, Nguyen Thanh Thuy,"Facial Expression Classification Using Artificial Neural Network and K-Nearest Neighbor", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.3, pp.27-32, 2015. DOI: 10.5815/ijitcs.2015.03.04

[18] Saptarsi Goswami, Amlan Chakrabarti,"Feature Selection: A Practitioner View", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.11, pp.66-77, 2014. DOI: 10.5815/ijitcs.2014.11.10


 

## Authors' Profiles

**Noshin Hagshenas** received his B.E. degree in computer software engineering from Lian Institute, Bushehr, Iran, and Department of Computer Engineering in 2018, and has received his M.Sc. degree in computer software engineering from of Lian Bushehr Institute, Iran, in 2021. Her hobbies are Optical Trapping, Optical Tweezers, Biophysics, Experimental Physics, Edge Computing, Edge Security, and Cloud Computing.

**Mousa Mojarad** received his PhD in Computer-Software Engineering in 2020. He is currently a lecturer and faculty member of the Islamic Azad University, Firoozabad Branch. His hobbies are Big Data, Face Recognition, Machine Learning, Pattern Recognition and Feature Extraction.

**Hassan Arfaeinia** obtained her B.E in Computer Science from Rafsanjan University, Kerman, Iran in 2009. HE received her M.S in Computer Engineering from the Amirkabir University of Technology, Tehran, Iran in 2011 and PhD in Computer Engineering from Islamic Azad University, North Tehran Branch, Iran in 2016. His main paper interests consist of Software Development, Machine Learning, Statistical Modeling and Computer Programming.

**How to cite this paper:** Noshin Hagshenas, Musa Mojarad, Hassan Arfaeinia, "A Fuzzy Approach to Fault Tolerant in Cloud using the Checkpoint Migration Technique", International Journal of Intelligent Systems and Applications(IJISA), Vol.14, No.3, pp.18-26, 2022. DOI: 10.5815/ijisa.2022.03.02

 