Asian Journal of Research in Computer Science

11(4): 19-34, 2021; Article no.AJRCOS.73754
ISSN: 2581-8260

State of Art Survey for Fault Tolerance Feasibility in Distributed Systems

Arshad A. Hussein[1*], Adel AL-zebari[1], Naaman Omar[1], Karwan Jameel Merceedi[1], Abdulraheem Jamil Ahmed[1], Nareen O. M. Salim[1], Sheren Sadiq Hasan[1], Shakir Fattah Kak[1], Ibrahim Mahmood Ibrahim[1], Hajar Maseeh Yasin[1] and Azar Abid Salih[1]

[1]Duhok Polytechnic University, Duhok, Kurdistan Region, Iraq.

Authors' contributions

This work was carried out in collaboration among all authors. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/AJRCOS/2021/v11i430268
Editor(s):
(1) Francisco Welington de Sousa Lima, Universidade Federal do Piauí, Brazil.
Reviewers:
(1) Sanjeev Kumar Dwivedi, Dr. SPM International Institute of Information Technology, India.
(2) Ahmed Adeeb Jalal, Al-Iraqia University, Iraq.
Complete Peer review History: https://www.sdiarticle4.com/review-history/73754

Review Article

Received 27 June 2021
Accepted 07 September 2021
Published 08 September 2021

ABSTRACT

The use of technology has grown dramatically, and computer systems are now interconnected via various communication mediums. The use of distributed systems (DS) in our daily activities has only gotten better with data distributions. This is due to the fact that distributed systems allow nodes to arrange and share their resources across linked systems or devices, allowing humans to be integrated with geographically spread computer capacity. Due to multiple system failures at multiple failure points, distributed systems may result in a lack of service availability. to avoid multiple system failures at multiple failure points by using fault tolerance (FT) techniques in distributed systems to ensure replication, high redundancy, and high availability of distributed services. In this paper shows ease fault tolerance systems, its requirements, and explain about distributed system. Also, discuss distributed system architecture; furthermore, explain used techniques of fault tolerance, in additional that review some recent literature on fault tolerance in distributed systems and finally, discuss and compare the fault tolerance literature.

**Corresponding author: E-mail: arshad.akre@yahoo.com;*

Keywords: Cluster; grid; P2P; distributed system; cloud; fault tolerance.

1. INTRODUCTION

A distributed system (DS) is a collection of independent computers that seem to users as a single, unified entity [1]. DSs are also collections of interconnected nodes that all work toward the same goal. The assignment has been split down into smaller tasks that have been handed to various nodes in order to respond to a specific challenge [2]. Each node finishes its portion of the job and sends the results back to the submission node [3]. Furthermore, since DSs may be homogeneous (cluster) or heterogeneous (Grid, Cloud, and P2P), they are vulnerable to difficulties like Quality of Service (QoS), Resource Selection, Load Balancing, and Fault Tolerance, among others [4].

Unlike uniprocessors, fault tolerance determines how a system reacts to abrupt hardware or software failure; breakdowns in a DS are harder to detect [5]. Fault tolerance is made up of two key components: failure detection and recovery [6]. Maintaining the system's functionality in the event of a breakdown or if any of its components get disconnected or malfunction is a major difficulty in DSs [7]. Retry, replication, check pointing, and message logging are just a few of the FT techniques available in the distributed paradigm [8].

A malfunctioning system causes human/economic harm, affects air traffic control, and interrupts telecommunications, among other things [9]. The necessity for reliable fault tolerance solutions reduces these threats to a minimum [10]. Faults are limited or partial in DSs [11]. Because the whole system isn't knocked down or unavailable, a section of a DS failure isn't as severe [12]. For example, if a system contains more than one processing core (CPU), if one core fails, the system will continue to operate as if it only had one physical core [13]. As a consequence, the remaining cores would regularly execute and analyze data. However, in a non-DS, if one of its components fails, the whole system or program fails and all related actions stop [14].

A system fault is a fault that results to an error when enabled. Fault tolerance includes many techniques in a distributed system architecture [15]. The failure tolerance is a dynamic technique to combine connected systems and to ensure that dispersed systems are reliable and accessible [16]. Hardware and redundancy solutions in distributed systems are famed as tolerance solutions (DS) [17]. Fault tolerance solutions for hardware include CPU's, communication links, memory, and I/O devices, whereas solutions for faulty software demand certain programs to deal with difficulties [18]. Solutions for failure tolerance that assist discover and recover problems, if practicable [19]. The failure tolerance of the software includes checkpoints and rollback retrieval [20]. Checkpoints are like a secure state or a functional snapshot of the whole system [21]. The fault tolerance validation approach is used to increase dependability and decrease construction mistakes [22]. There are several ways of doing this, one is to improve a formal language system model and utilize a validation programmer [23]. In contrast, error correction is the approach used to improve the dependability [24]. Failure to identify errors is reduced by redundancy tolerance. Before latent error processing is successful, latent errors are detected and repaired [25]. In fault tolerance systems, which involve hardware redundancy, software redundancy, redundancy of information and time redundancy, are also employed in many forms of redundancy. Hardware fault tolerance is the optimum in most applications [26].

In this paper we show ease fault tolerance systems, its requirements, and explained what a distributed system is. Furthermore, discuss distributed system architecture, and then explain used techniques of fault tolerance. Also, review some recent literature on fault tolerance in distributed systems and finally, we discuss and compare the fault tolerance literature.

2. BACKGROUND THEORY

2.1 Ease Fault Tolerance Systems

The fault tolerance system is an essential topic in distributed computing since it keeps the system functioning in a failure. Most importantly, keeping the system operating even if one or more of its components fails or malfunctions [27].

To be fault resistant, a system is related to "Dependable Systems." As demonstrated in Fig. 1, reliability meets many essential criteria in the fault tolerance system. The following criteria are required: availability, dependability, safety, maintenance [28].

- Availability: The point at which a system is ready to start delivering its functionality to its consumers. At any given moment, highly available systems are operational [29].
- Reliability: This refers to a computer system's capacity to function constantly without interruption [30]. In contrast to availability, dependability is defined as a time period rather than a moment in time. For a lengthy length of time, a highly dependable system functions constantly and without interruption [31].
- Safety: happens when a system fails to properly perform its related operations and its actions are erroneous, but no catastrophic failure occurs [32].
- Maintainability: Also, a highly sustainable system may suggest a high degree of accessibility, particularly if the accompanying problems can be mechanically diagnosed and repaired [33].

2.2 Distributed System

Distributed systems (DS) are ones that don't share memory and a common clock. Distributed nodes exchange data through a communication channel to connect and transmit information [34]. Each computer in a DS has its own memory and operating system, and the node using it retains local resources [35]. Remote resources, on the other hand, are those accessible via a network or communication channel [36]. Divided systems, especially for highly demanding and complicated control systems, have developed considerably in the past many years in terms of capability, scalability and openness [37]. This presents significant problems for the design of a functioning and dependable system, mainly because the hardware architecture has been loosely interconnected without a common physical memory [38]. For instance, control spreader synchronization process usually utilizes common variables on a single computer but must be related to message transmission on a distributed system [39]. The additional time lag associated with the transmission of messages across a network increases process asynchrony and requires the usage of specific protocols to coordinate its activities [40]. Like the distributed tolerance unit, the distributed system software and hardware monitors the problem and diagnoses it before any defects arise [41].

DS has developed throughout time, but the most prevalent implementations now are designed to work mainly on the Internet and in particular on the cloud [42]. A distributed system begins with a task to render a video to generate a ready-to-release completed product [43]. This tasking, such as a video editor on a client computer, is being handled through the web application or distributed apps. Scalability, competitors, available fault tolerances, transparency, heterogeneity and replication are the key properties of DS [44]. The scalability that it may expand as the workload size increases is an essential property of distributed systems by adding extra processing units or network nodes, if necessary. The competition feature is one component running in DS at the same time [45]. The absence of a "ground clock" is also characterized by when tasks take place at various rates and out of sequence [46]. Fault tolerance is the heart of the system work, since if a node fails, the other nodes may function without creating the whole calculation problem [47]. Transparency is an external programmer or end-user, rather than the underlying pieces, obtains a distributed system as a single computing unit [48]. The range of systems components called heterogeneity is typically asynchronous in most distributed systems, with various hardware, middleware, software and operating systems. The distributed systems may therefore be expanded by adding additional components [49]. The fourth characteristic of DS is replication that enables the exchange of information and messages, guaranteeing consistency amongst redundant resources, such as software, hardware and other components [50]. The function of distributed systems suits business demands, which do not have the complexity of a complete network of telecommunications [51]. Scalability and increased performance can be achieved in ways that monolithic systems cannot and since distributions can rely on the capabilities of other computer devices and processes, they can offer features that would be hard or impossible to build on a single machine [52].

In a DS, a set of rules is used to synchronize the operations of several or distinct processes through a communication network, resulting in a distinct collection of related activities [53]. Fig. 2 shows the communication network connecting systems in dispersed environments.

The autonomous system or computers in a DS communication environment access resources remotely or locally [54]. These elements are merged to form a single, comprehensible system

[55]. The user is unaware of the many interconnected procedures that ensure the work is completed effectively in a distributed environment [56]. In a DS no one system is required to fulfill a task or to shoulder the whole system's load [57].

2.3 Distributed System Architecture

The DS's design is based on current operating systems and network software [58]. A DS is made up of a group of self-contained computers connected via a computer network and distribution middleware [59]. In a DS, the distribution middleware allows the corresponding computers to manage and share the resources of the corresponding system, giving the computer users the impression that the system is a single unified computing infrastructure [60]. Middleware is the glue that holds dispersed applications together, regardless of their location, computer hardware, network protocols, operating systems, or programming languages [61]. Standard services like as naming, concurrency management, event dissemination, security, and permission are provided by the middleware [62]. Fig. 3 depicts the DS architecture, with the middleware providing services to the distributed environment's linked systems [63]. The structure of a DS might be completely linked or partly linked networks [64].

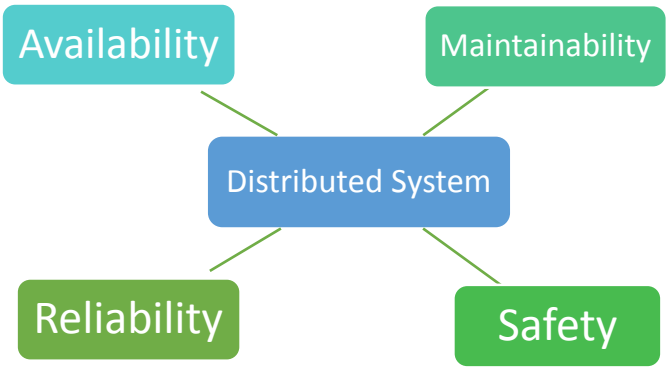

Fig. 1. Trustworthy DS

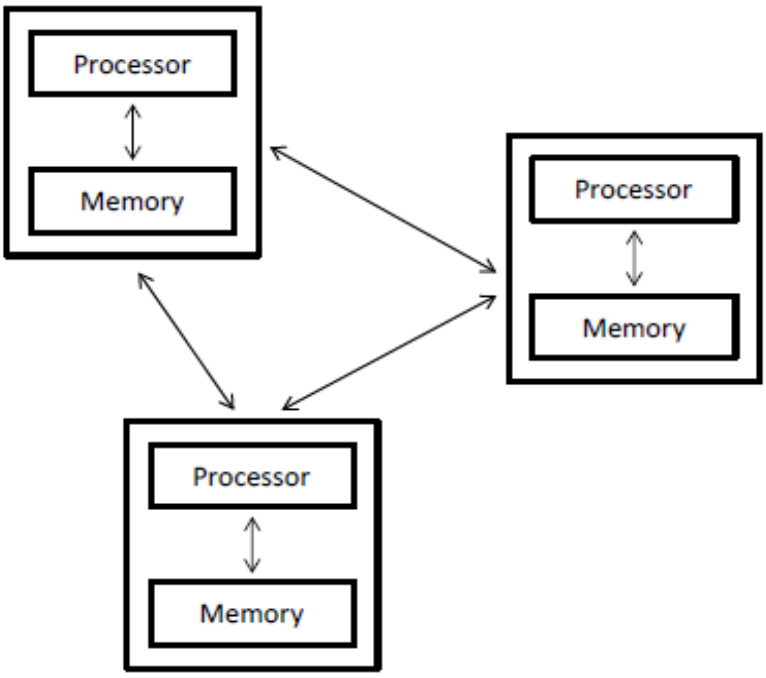

Fig. 2. Distributed system

A full-connected network, as shown in Fig. 3, is one in which each node is connected to the rest of the network. The disadvantage of this network is that when a new computer is installed, the number of nodes linked to nodes increases physically because the network connects nodes [65]. As the number of nodes has increased, so has the number of file descriptors and the complexity of each node's interaction. File descriptors are conceptual indicators for accessing a file or other input/output resource, such as a pipe or network connection [66]. Thus, the capability of the connected node to open file descriptors and manage new connections constrained the ability of networked systems to function correctly [67]. Because messages transmitted from one node to the next are routed over a single connection, fully linked network systems can continue interact with others, even if the node and the link fail [68]. Components and connections are linked together with distributed system designs. The components might be single nodes or significant architectural components, whereas the connections link each of them [69]. Distributed systems connect to a network that shares all computers, software and hardware components together to transmit messages for communication. It may be connected to that network with an IP [70]. The messages exchanged between machines contain data formats, which systems like databases, objects and files wish to share [71].

Client server architecture offers data and services integration and allows customers inherent complexity such as communication protocols to be removed from them [72]. Clients can request to the right server because of the simplicity of their client-server design. These applications are made as transactions [73]. SQL or PL/SQL transactions and features which access single databases and services are frequently customer transactions [74]. A common resource server such as a data basket, printer and Web server consists of the distributed system architecture [75]. It had several clients and users deciding when to use, how to utilize and display the shared resource, changing data and returning it to the server [76]. Examples for the distributed real-time systems, parallel processing and distributed database systems are networking, the telecommunications network [77].

2.4 Fault Tolerance Techniques

As seen in Fig. 4, several fault tolerance strategies are available in classic distributed paradigms and may be applied at the task or process level [78].

2.4.1 Reactive FT

When a system fails, reactive FT methods are employed to mitigate the effect of the failure on the system. Retry, replication, check-pointing, and message logging [79].

1) ***Retry****: the most common failure recovery method, with the assumption that the cause of the failures would not be revisited in future retries [80].*
2) ***Replication****: method is to create several task clones for each running job and send them to different hosts so that none of the duplicated jobs hang (due to a host crash, the host disconnected from the network). Client, etc.), the job would be completed successfully [81].*
3) ***Check-pointing****: The most basic FT approach employed in DS is check-pointing; the basic idea is that the system saves its state on a regular basis on trustworthy and stable storage [82]. The system was restarted from the most recent checkpoint rather than from the beginning after a crash. In your pick, underline all author and affiliation lines [83].*
4) ***Message Logging****: Because check-pointing is an expensive technique, several methods have been devised to reduce the number of checkpoints while still enabling recovery [84]. This concept is based on the idea that if messages can be replayed in a predetermined order, the system can always achieve a consistent state without having to restore it from stable storage [85]. Instead, a check-pointed state is utilized as a starting point, and all messages sent since then are simply retransmitted and processed the same way they were before [83].*

2.4.2 Proactive FT

To avoid recovery from faults and errors, proactive FT predicts breakdowns and replaces healthy (functional) components for problematic components [86]. Preemptive migration, software rejuvenation, load balancing, and other techniques follow this paradigm [87].

1) Software Rejuvenation: A programmed change in the status is made with each system reboot [88].

2) Self-healing: The core concept is to automate the failure of a virtual machine-based application instance [89].

3) Preemptive Migration: method for continuously examining and evaluating an application. Fault tolerance classification [90].

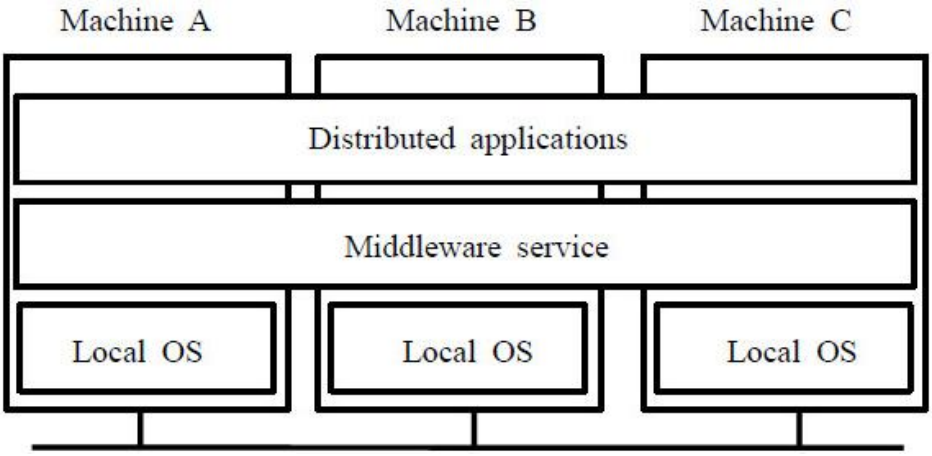

Fig. 3. A distributed system's basic architecture

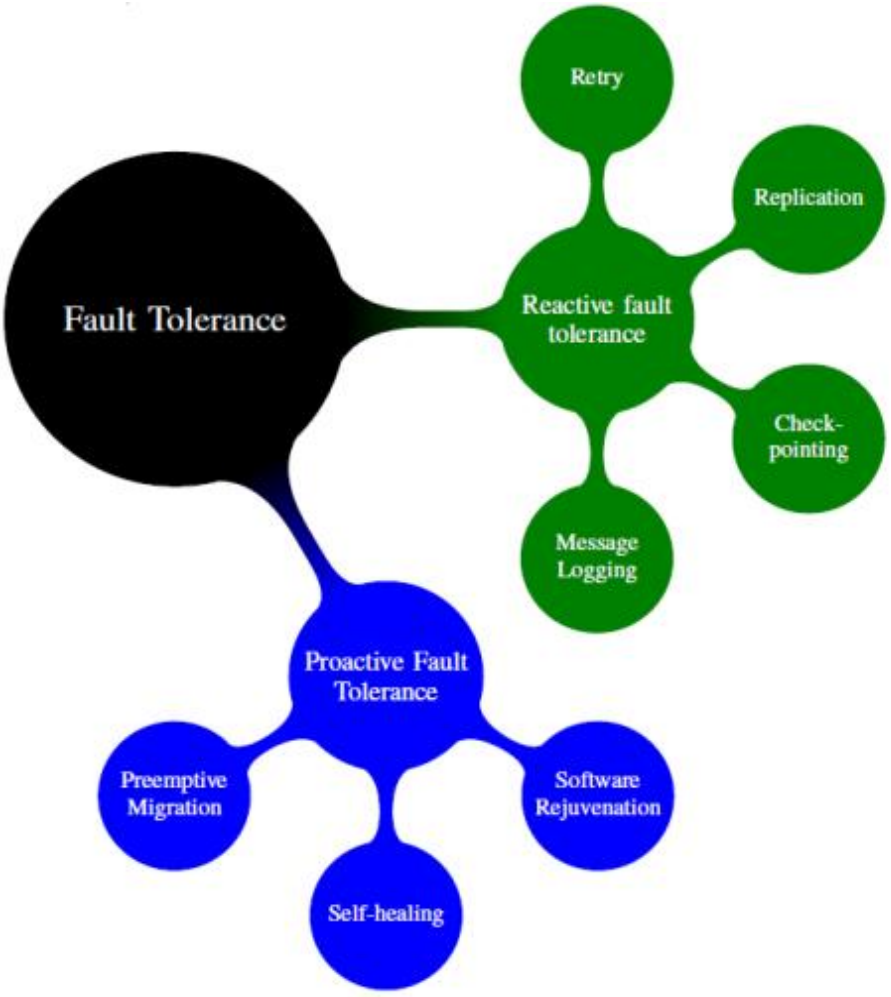

Fig. 4. Fault tolerance taxonomy [59]

3. LITERATURE REVIEW

Fochi, Caimi et al. [91] offered a technique for recovering when an MPE fails and offered a technique for properly migrating management software to a new PE The protocol uses task migration to free a processor if there is no processor available to accept the kernel that was previously running in a problematic processor. The idea is transparent to the many-core applications, with just a little execution cost detected during administration and job transfer.

Vladimirova and Fayyaz [66] Proposed FT distributed satellite architecture. Following system functionality and application actions, quantifiable performance measurements are developed. Then, as an FT distributed computing system, a revolutionary multi-processor chip (MPSoC) AOCS computer is built. XILINX Zynq FPGAs then prototype distributed MPSoC AOCS. Using a MATLAB AOCS model and a fault injection approach, the prototype was thoroughly tested as loop hardware. The testing findings reveal that in terms of processor fault resilience and computational performance, the FT distributed computing architecture outperforms standard on-board solutions.

Arafa, Barai et al. [83], examined at two distinct distributed file systems' FT approaches, HDFS and Ceph. Erasure coding is provided in the current HDFS and Ceph versions, in addition to standard replication. Using common benchmarks and fault injection, both systems analyze the performance and storage cost of replication and erasure coding implementations statistically. Results highlight the trade-offs of replication and erasure coding algorithms and serve as a basis for constructing high-availability, high-performance storage systems.

Bravo, Rodrigues et al. [80] Explored a unique way including describing not just system configuration, but also fault handling behavior and how the system responds to changes in workload in a policy language handled outside the controlled system. This methodology shows how to employ a single streamlined, controlled system codebase to easily fulfill a broad variety of reliability criteria.

Hussain, Cui et al. [57] Color is a message-based recovery method that prevents failures. He recommended implementing MPI utilizing rescuer processes that share resources with original processes. He established that Color surpasses standard checkpoint/restart (C/R) and pure replication throughout a range of core counts at accurate accuracy using model-based analysis and real-system trials.

Yusuf and Junaidu [92] Proposed an FT strategy paradigm that would enable self-detection and tolerate temporary node-dependent failures. The model is compared to preceding techniques in terms of detection capability, internode interdependence, and execution cost.

Zhao, Shen et al. [27] Using RDP codes instead of Cocytus RS codes. To make DS more efficient, Shortens RDP code recovery time utilizing two techniques: RDOR and CRR architectural decoding, dramatically boosting user experience and system stability. Finally, this study assesses both optimization approaches' performance as well as the different application situations they may be applied in.

Boem, Gallo et al. [78] Using Active Fault Isolation, a scalable distributed FTC technique was presented to monitor coupled subsystems. Following identification of faults, the recommended methodology ensures that the issue may be precisely isolated in a minimum number of steps and local controllers securely reconfigured, or that disconnection of the faulty subsystem is preferred to reduce the spread of the effects of the issue.

Deng, Che et al. [93] A novel distributed control strategy was given to solve the fault-tolerant problem of output regulation for linear MASs with actuator faults. New distributed estimation methodology including online learning Techniques for detecting and calculating the system matrix of the ecosystem were developed for each subsystem. To offset actuator difficulties, a novel distributed FT controller based on the developed estimator was suggested.

Ghosh, Eisele et al. [94] discussed the fault management component of the RIAPS architecture, demonstrating its implementation in a transactive energy application. The fault management subsystem was created utilizing a methodological approach wherein the different frame failure scenarios were uncovered by assessing interaction patterns across RIAPS architectural tiers. The value of various services and their communication protocols in improving the resilience characteristics of the system was accurately analyzed and implemented.

Khalili, Zhang et al. [53] FT distributed leader following tracking approach for high order nonlinear unpredictable multi agent systems even with multiple simultaneous processes and actuator failures in distributed agents, adaptive learning techniques based on neural networks are meant to learn unknown defect functions, ensuring system stability and cooperative tracking. Over direct connections, the time-varying command of the leader is sent to just a subset of follower agents, and each follower agent shares local measurement information with its neighbors through a bidirectional but asymmetric topology.

Knasmüller, Hochreiner et al. [60] Proposed pathfinder architecture that tackles this constraint by allowing functional redundancy at stream processing paths. Pathfinder reacts to operator faults during system execution, relocating to a fault-free path with similar functionality. Pathfinder employ circuit breaker pattern to restore the main route while recovering a failed operator.

Li, Hua et al. [63] A distributed FT consensus control problem was investigated for a sort of uncertain, nonlinear multi agent systems with actuator and process flaws. Most present studies on tolerant control of multi-agent systems concentrate mostly on actuator failures. Unlike works, it is prone to both actuator and process failures. We present a less conservative criteria for the unknown nonlinear term. The recommended controllers, according to Lyapunov's stability theory, induce followers to agree with the leader.

Loutskii, Volokyta et al. [95] provides a strategy for improving the hyper de Bruijn topology's FT. It is suggested that extra de Bruijn be used. Routing techniques are investigated, and fast and FT routing strategies are provided. These approaches are based on quasi quantum relationships and are the outcome of the utilization of surplus code. The topology that resulted was synthesized. Its qualities are subjected to scaling computations. A comparison with various topologies was carried out.

Mahjoubi, Zeynalpour et al. [96] in distributed controllers, he demonstrated a load balancing and fault tolerant (LBFT) technique. The technique, which facilitates group switch migration, has been developed as a module on top of each controller. He examined the effects of load balancing and FT on RTT, packet loss, and throughput.

Pareek, Sharma et al. [97] Using RAID-5 architectural principles, he could safely recover data in the case of a single site crash. An revised model, mathematically justified, was presented to allow FT for a simultaneous double disk failure. To verify the model theory, Python programming language simulations were undertaken. Based on these simulations, the recommended improvements were tolerant to a double disk failure.

Shi [98] the cooperative FT formation control problem is investigated for a type of nonlinear leader follower multi-agent systems with actuator flaws (MASs). For each subsequent agent, a kind of decentralized observer is built using the local output information. A unique adaptive fault estimator is then used to estimate the real time fault signal. Using the obtained status and fault information, a novel fault-tolerant control method for stabilizing the closed loop system is devised.

Han, Jang et al. [36] suggested switch-centric byzantine FT (SC-BFT) Techniques for rapid consensus in distributed SDN enabled by programmable switches. In terms of communication overhead and reaction time, the evaluation findings show that SC-BFT outperforms traditional BFT Techniques (e.g., PBFT). SC-BFT is also shown to be less sensitive to the location of the consensus processing node.

Zhang, Xue et al. [99] He suggested a distributed adaptive (FT) control method, which is supported by the Lyapunov stability theory, and simulation studies validate the technique's convergence and resilience. Under the "leader-follower" concept, the multi-agent formation system can not only swiftly restore the system's normal operation but also guarantee that the agents move in accordance with the pre-set formation. In order to build the pre-set configuration, however, only the fault-tolerant control from various beginning locations at varied speeds was explored.

Perez, Goodloe et al. [100] He has proposed methodically evaluating the failure model in a distributed satellite swarm, introducing FT techniques, and verifying their appropriateness. He demonstrated how seeing defects through the eyes of the receiver simplifies the fault model and aids in the analysis of the FT measures necessary. Their computational abstraction represents satellites that communicate sensor data. Using linear temporal logic, he modeled

and validated the features of consensus and interactive agreement. He utilized this simulation to inject faults and random testing to alter the likelihood of various types of defects in the system and to ensure that the FT approaches could rectify these problems.

4. DISCUSSION AND COMPARISON

Problem tolerance is an important component of a distributed system because it maintains the system's continuation and operation in the event of a fault or failure. Using the comparison details illustrated in Table 1, noted the number of techniques used in twenty of the fault tolerance literature, and discusses the problems that face distributed systems, as well as the results of solving these problems using fault tolerance techniques. Active fault Isolation, LBFT, RIAPS, AOCS, hierarchically organized MPEs, and RAID-5 are the most commonly used techniques. FT is a dynamic strategy for keeping interconnected systems together, ensuring dependability and availability in distributed systems. Efficient FT approaches aid in the detection of flaws and, if feasible, their recovery.

Table 1. Comparison fault detection and tolerance techniques used in distributed systems

Ref.	year	methods topologies/techniques	problems Detected/fault	Significant result
[98]	2019	Cooperative Formation Control	multi-agent systems (MASs)	The numerical demonstration validated the theoretical findings provided
[92]	2018	Parity Checking	Transient defect in parallel query processing	Detection capability, inter-node interdependence, and execution cost
[93]	2019	A novel distributed estimation	the FT cooperative output regulation challenge for linear MASs with actuator failures	The suggested strategy can address the cooperative FT output control issue
[57]	2018	Co-Located Rescuers	a message logging-based technique that allows for fail-safe recovery	an MPI implementation that employs rescuer processes that share resources with the original processes
[53]	2019	Adaptive learning	a DFTC for a family of high order nonlinear uncertain(MASs)	full-state assessment and just a limited amount of output measurement
[97]	2019	RAID-5	reduces the cases of complete system failure	Improving reliability with RAID
[100]	2019	swarm of satellites	assesses FT Techniques in a satellite swarm	the suggested fault model's applicability
[27]	2018	RDP codes - RS codes	RDP code recovery time is greatly reduced	distributed system more efficient
[95]	2019	Hyper de Bruijn	improve the HDB by using excess de Bruijn	enhance the HDB topology's FT
[36]	2020	(SC-BFT) Techniques	Consensus latency and traffic load are increased since all messages must be validated and multicast among controllers	SC-BFT gives an 80 percent decrease in reaction time as well as a considerable decrease in communication overhead
[83]	2018	HDFS and Ceph	Two typical distributed file systems' FT techniques	show the trade-offs of replication and erasure coding
[78]	2018	Active Fault Isolation.	Large-Scale Systems	It has been described a scalable distributed FTC technique for monitoring coupled subsystems

Ref.	year	methods topologies/techniques	problems Detected/fault	Significant result
[96]	2019	(LBFT)	LBFT on RTT	a single point of failure are all important considerations
[94]	2019	(RIAPS)	real time, embedded systems, fault detection.	cyber-physical system.
[63]	2019	less conservative Lipschitz condition	Uncertain nonlinear multi agent systems(MAS) with actuator failures and process faults	verify the effectiveness of the theoretical
[60]	2019	Pathfinder framework	the length of the failure by a large period of time	The circuit breaker pattern is used by Pathfinder
[66]	2018	(AOCS)	a fault-tolerant distributed design that is novel	show that the FT distributed computing system outperforms
[99]	2020	DAFTC	Multi-agent with Actuator Fault	The adaptive control strategy provides a high level of resilience
[80]	2018	PBA	Byzantine FT Distributed Graph Database	A single simplified managed system codebase may be utilized
[91]	2018	MPEs hierarchically organized	When an MPE fails, there is a recovery technique	safely migrate the management software to a new processing element

5. CONCLUSION

Fault tolerance is a key element in DS design and is one of the main subjects in distributed systems. FT is defined as the ability of a system to operate in the event of failure. Problem tolerance is an important component of a distributed system because it maintains the system's continuation and operation in the event of a fault or failure. The most methods for fault tolerance detection in distributed system adopted by the researchers which are Active fault Isolation, LBFT, RIAPS, AOCS, hierarchically organized MPEs, and RAID-5. In this paper shows ease fault tolerance systems, its requirements, and explained what a distributed system is. Then discusses distributed system architecture, and then we explain used techniques of fault tolerance then, we review some recent literature on fault tolerance in distributed systems, and finally, we discuss and compare the fault tolerance literature.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Abdullah RM, Ameen SY, Ahmed DM, Kak SF, Yasin HM, Ibrahim IM, et al. Paralinguistic speech processing: An overview, Asian Journal of Research in Computer Science. 2021;34-46.
2. Ibrahim IM, Ameen SY, Yasin HM, Omar N, Kak SF, Rashid ZN, et al. Web server performance improvement using dynamic load balancing techniques: A Review, Asian Journal of Research in Computer Science. 2021;47-62.
3. Zeebaree S, Ameen S, Sadeeq M. Social media networks security threats, risks and recommendation: A case study in the kurdistan region, International Journal of Innovation, Creativity and Change. 2020;13:349-36,.
4. Ahmed DM, Ameen SY, Omar N, Kak SF, Rashid ZN, Yasin HM, et al. A state of art for survey of combined iris and fingerprint recognition systems, Asian Journal of Research in Computer Science. 2021;18-33.
5. Maulud DH, Ameen SY, Omar N, Kak SF, Rashid ZN, Yasin HM, et al. Review on natural language processing based on different techniques, Asian Journal of Research in Computer Science. 2021;1-17.
6. Salih AA, Ameen SY, Zeebaree SR, Sadeeq MA, Kak SF, Omar N, et al. Deep learning approaches for intrusion detection, Asian Journal of Research in Computer Science. 2021;50-64.
7. Hassan RJ, Zeebaree SR, Ameen SY, Kak SF, Sadeeq MA, Ageed ZS, et al. State of art survey for iot effects on smart city technology: Challenges, opportunities, and solutions, Asian Journal of Research in Computer Science. 2021;32-48.
8. Yahia HS, Zeebaree SR, Sadeeq MA, Salim NO, Kak SF, Adel AZ, et al. Comprehensive survey for cloud computing based nature-inspired algorithms optimization scheduling, Asian Journal of Research in Computer Science. 2021;1-16.
9. Ageed ZS, Zeebaree SR, Sadeeq MM, Kak SF, Rashid ZN, Salih AA, et al. A survey of data mining implementation in smart city applications, Qubahan Academic Journal. 2021;1:91-99.
10. Sulaiman MA, Sadeeq M, Abdulraheem AS, Abdulla AI. Analyzation study for gamification examination fields, Technol. Rep. Kansai Univ. 2020;62:2319-2328.
11. Ageed ZS, Zeebaree SR, Sadeeq MA, Abdulrazzaq MB, Salim BW, Salih AA, et al. A state of art survey for intelligent energy monitoring systems, Asian Journal of Research in Computer Science. 2021;46-61.
12. Sadeeq M, Abdulla AI, Abdulraheem AS, Ageed ZS. Impact of electronic commerce on enterprise business, Technol. Rep. Kansai Univ. 2020;62:2365-2378.
13. Alzakholi O, Shukur H, Zebari R, Abas S, Sadeeq M. Comparison among cloud technologies and cloud performance, Journal of Applied Science and Technology Trends. 2020;1:40-47.
14. Salih A, Zeebaree ST, Ameen S, Alkhyyat A, Shukur HM. A Survey on the Role of Artificial Intelligence, Machine Learning and Deep Learning for Cybersecurity Attack Detection, in 2021 7th International Engineering Conference Research & Innovation amid Global Pandemic (IEC). 2021;61-66.
15. Abdullah DM, Ameen SY, Omar N, Salih AA, Ahmed DM, Kak SF, et al. Secure data transfer over internet using image steganography, Asian Journal of Research in Computer Science. 2021;33-52.

16. Kareem FQ, Ameen SY, Salih AA, Ahmed DM, Kak SF, Yasin HM, et al. SQL injection attacks prevention system technology, Asian Journal of Research in Computer Science. 2021;13-32.
17. Ageed Z, Mahmood MR, Sadeeq M, Abdulrazzaq MB, Dino H. Cloud computing resources impacts on heavy-load parallel processing approaches, IOSR Journal of Computer Engineering (IOSR-JCE). 2020;22:30-41.
18. Ismael HR, Ameen SY, Kak SF, Yasin HM, Ibrahim IM, Ahmed AM, et al. Reliable communications for vehicular networks, Asian Journal of Research in Computer Science. 2021;33-49.
19. Sallow A, Zeebaree S, Zebari R, Mahmood M, Abdulrazzaq M, Sadeeq M. Vaccine tracker, SMS reminder system: Design and Implementation; 2020.
20. Abdulla AI, Abdulraheem AS, Salih AA, Sadeeq M, Ahmed AJ, Ferzor BM, et al. Internet of things and smart home security, Technol. Rep. Kansai Univ. 2020;62:2465-2476.
21. Sadeeq MA, Zeebaree SR, Qashi R, Ahmed SH, Jacksi K. Internet of Things security: A survey, in 2018 International Conference on Advanced Science and Engineering (ICOASE). 2018;162-166.
22. Abdulraheem AS, Salih AA, Abdulla AI, Sadeeq M, Salim N, Abdullah H, et al. Home automation system based on IoT; 2020.
23. Salih AA, Zeebaree S, Abdulraheem AS, Zebari RR, Sadeeq M, Ahmed OM. Evolution of mobile wireless communication to 5G revolution, Technology Reports of Kansai University. 2020;62:2139-2151.
24. Abdulazeez AM, Zeebaree SR, Sadeeq MA. Design and implementation of electronic student affairs system, Academic Journal of Nawroz University. 2018;7:66-73.
25. Dino HI, Zeebaree S, Salih AA, Zebari RR, Ageed ZS, Shukur HM, et al. Impact of process execution and physical memory-spaces on OS performance, Technology Reports of Kansai University. 2020;62:2391-2401.
26. Hamdi SJ, Ibrahim IM, Omar N, Ahmed OM, Rashid ZN, Ahmed AM, et al. A Comprehensive Study of Malware Detection in Android Operating Systems.
27. Zhao S, Shen L, Li Y, Stones RJ, Wang G, Liu X. An efficient fault tolerance framework for distributed in-memory caching systems, in 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). 2018;553-560.
28. Ageed ZS, Ahmed AM, Omar N, Kak SF, Ibrahim IM, Yasin HM, et al. A State of Art Survey of Nano Technology: Implementation, Challenges, and Future Trends.
29. Abdulqadir MM, Salih AA, Ahmed OM, Hasan DA, Haji LM, Ahmed SH, et al. A Comprehensive Study of Caching Effects on Fog Computing Performance.
30. Yazdeen AA, Zeebaree SR, Sadeeq MM, Kak SF, Ahmed OM, Zebari RR. FPGA implementations for data encryption and decryption via concurrent and parallel computation: A review, Qubahan Academic Journal. 2021;1:8-16.
31. Ageed ZS, Zeebaree SR, Sadeeq MM, Kak SF, Yahia HS, Mahmood MR, et al. Comprehensive survey of big data mining approaches in cloud systems, Qubahan Academic Journal. 2021;1:29-38.
32. Abdulrahman LM, Zeebaree SR, Kak SF, Sadeeq MA, Adel AZ, Salim BW, et al. A state of art for smart gateways issues and modification, Asian Journal of Research in Computer Science. 2021;1-13.
33. Abdulqadir HR, Zeebaree SR, Shukur HM, Sadeeq MM, Salim BW, Salih AA, et al. A study of moving from cloud computing to fog computing, Qubahan Academic Journal. 2021;1:60-70.
34. AL-Zebari A, Zeebaree S, Jacksi K, Selamat A. ELMS–DPU ontology visualization with Protégé VOWL and Web VOWL, Journal of Advanced Research in Dynamic and Control Systems. 2019;11:478-85.
35. Zeebaree A, Adel A, Jacksi K, Selamat A. Designing an ontology of E-learning system for duhok polytechnic university using protégé OWL tool, J Adv Res Dyn Control Syst. 2019;11:24-37.
36. Han S, Jang S, Lee H, Pack S. Switch-centric byzantine fault tolerance mechanism in distributed software defined networks, IEEE Communications Letters. 2020;24:2236-2239.
37. Adel AZ, Zebari S, Jacksi K. Football ontology construction using oriented programming, Journal of Applied Science and Technology Trends. 2020;1:24-30.
38. Selamat SAAZA. Electronic learning management system based on

semantic web technology: A Review, Int. J. Adv. Electron. Comput. Sci. 2017; 4:1-6.
39. Abdullah RM, Abdulazeez AM, Al-Zebari A. Machine learning algorithm of intrusion detection system, Asian Journal of Research in Computer Science. 2021;1-12.
40. Shukur H, Zeebaree SR, Ahmed AJ, Zebari RR, Ahmed O, Tahir BSA, et al. A state of art survey for concurrent computation and clustering of parallel computing for distributed systems, Journal of Applied Science and Technology Trends. 2020;1:148-154.
41. Tahir B, Ali Saktioto J, Fadhali M, Rahman R, Ahmed A. A study of FBG sensor and electrical strain gauge for strain measurements, Journal of Optoelectronics and Advanced Materials. 2008;10:2564-2568.
42. Harki N, Ahmed A, Haji L. CPU scheduling techniques: A review on novel approaches strategy and performance assessment, Journal of Applied Science and Technology Trends. 2020;1:48-55.
43. Ahmed A, Ahmed O. Correlation pattern among morphological and biochemical traits in relation to tillering capacity in sugarcane (Saccharum Spp), Acad J Plant Sci. 2012;5:119-122.
44. Ahmed AJ, Mohammed FH, Majedkan NA. An evaluation study of an e-learning course at the Duhok Polytechnic University: A Case Study, Journal of Cases on Information Technology (JCIT). 2022;24:1-11.
45. Ahmed O, Geraldes R, Ahmed A, DeLuca G, Palace J. Multiple sclerosis and the risk of venous thrombosis: A systematic review, in Multiple Sclerosis Journal. 2017;757-758.
46. Salim NO, Abdulazeez AM. Human diseases detection based on machine learning algorithms: A review, International Journal of Science and Business. 2021; 5:102-113.
47. Sallow AB, Sadeeq M, Zebari RR, Abdulrazzaq MB, Mahmood MR, Shukur HM, et al. An investigation for mobile malware behavioral and detection techniques based on android platform, IOSR Journal of Computer Engineering (IOSR-JCE). 2020;22:14-20.
48. Salim NO, Zeebaree SR, Sadeeq MA, Radie A, Shukur HM, Rashid ZN. Study for food recognition system using deep learning, in Journal of Physics: Conference Series. 2021;012014.
49. Dino H, Abdulrazzaq MB, Zeebaree S, Sallow AB, Zebari RR, Shukur HM, et al. Facial expression recognition based on hybrid feature extraction techniques with different classifiers, TEST Engineering & Management. 2020;83:22319-22329.
50. Salim NO, Abdulazeez AM. Science and Business, International Journal. 5:102-113.
51. Zeebaree S, Zebari RR, Jacksi K. Performance analysis of IIS10. 0 and Apache2 Cluster-based Web Servers under SYN DDoS Attack, TEST Engineering & Management. 2020;83:5854-5863.
52. Eesa AS, Sadiq S, HASSAN M, Orman Z. Rule generation based on modified cuttlefish algorithm for intrusion detection system, Uludağ University Journal of The Faculty of Engineering. 2021;26:253-268.
53. Khalili M, Zhang X, Cao Y, Polycarpou MM, Parisini T. Distributed fault-tolerant control of multiagent systems: An adaptive learning approach, IEEE Transactions on Neural Networks and Learning Systems. 2019;31:420-432.
54. Jader OH, Zeebaree S, Zebari RR. A state of art survey for web server performance measurement and load balancing mechanisms, International Journal of Scientific & Technology Research. 2019;8:535-543.
55. Eesa AS. Optimization algorithms for intrusion detection system: A review, International Journal of Research-Granthaalayah. 2020;8:217-225.
56. Zeebaree S, Zebari RR, Jacksi K, Hasan DA. Security approaches for integrated enterprise systems performance: A Review, Int. J. Sci. Technol. Res. 2019;8.
57. Hussain Z, Cui X, Znati T, Melhem R. Color: Co-located rescuers for fault tolerance in hpc systems, in 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). 2018;569-576.
58. Sari A, Akkaya M. Fault tolerance mechanisms in distributed systems, International Journal of Communications, Network and System Sciences. 2015;8:471.
59. Haji SH, Zeebaree SR, Saeed RH, Ameen SY, Shukur HM, Omar N, et al. Comparison of software defined networking with traditional networking,

Asian Journal of Research in Computer Science. 2021;1-18.
60. Knasmüller B, Hochreiner C, Schulte S. Pathfinder: Fault tolerance for stream processing systems, in 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService). 2019;29-39.
61. Ibrahim BR, Zeebaree SR, Hussan BK. Performance measurement for distributed systems using 2TA and 3TA based on OPNET Principles, Science Journal of University of Zakho. 2019;7:65-69.
62. Zebari S, Yaseen NO. Effects of parallel processing implementation on balanced load-division depending on distributed memory systems, J. Univ. Anbar Pure Sci. 2011;5:50-56.
63. Li Y, Tan C. A survey of the consensus for multi-agent systems, Systems Science & Control Engineering. 2019;7:468-482.
64. Zeebaree SR, Sallow AB, Hussan BK, Ali SM. Design and simulation of high-speed parallel/sequential simplified DES code breaking based on FPGA, in 2019 International Conference on Advanced Science and Engineering (ICOASE). 2019;76-81.
65. Malallah H, Zeebaree SR, Zebari RR, Sadeeq MA, Ageed ZS, Ibrahim IM, et al. A comprehensive study of kernel (issues and concepts) in different operating systems, Asian Journal of Research in Computer Science. 2021;16-31.
66. Vladimirova T, Fayyaz M. Fault-tolerant distributed attitude and orbit control system for space applications, in 2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS). 2018;43-50.
67. Zebari DA, Haron H, Zeebaree SR, Zeebaree DQ. Multi-Level of DNA encryption technique based on DNA arithmetic and biological operations, in 2018 International Conference on Advanced Science and Engineering (ICOASE). 2018;312-317.
68. Yasin HM, Zeebaree SR, Sadeeq MA, Ameen SY, Ibrahim IM, Zebari RR, et al. IoT and ICT based smart water management, monitoring and controlling system: A review, Asian Journal of Research in Computer Science. 2021;42-56.
69. Ibrahim IM. Task scheduling algorithms in cloud computing: A review, Turkish Journal of Computer and Mathematics Education (TURCOMAT). 2021;12:1041-1053.
70. Zebari IM, Zeebaree SR, Yasin HM. Real time video streaming from multi-source using client-server for video distribution, in 2019 4th Scientific International Conference Najaf (SICN). 2019;109-114.
71. Yasin HM, Zeebaree SR, Zebari IM. Arduino based automatic irrigation system: Monitoring and SMS controlling, in 2019 4th Scientific International Conference Najaf (SICN). 2019;109-114.
72. Ibrahim BR, Khalifa FM, Zeebaree SR, Othman NA, Alkhayyat A, Zebari RR, et al. Embedded system for eye blink detection using machine learning technique, in 2021 1st Babylon International Conference on Information Technology and Science (BICITS). 2021;58-62.
73. Hasan DA, Zeebaree SR, Sadeeq MA, Shukur HM, Zebari RR, Alkhayyat AH. Machine learning-based diabetic retinopathy early detection and classification systems-a survey, in 2021 1st Babylon International Conference on Information Technology and Science (BICITS). 2021;16-21.
74. Zeebaree S, Yasin HM. Arduino based remote controlling for home: Power saving, security and protection, International Journal of Scientific & Engineering Research. 2014;5:266-272.
75. Jijo BT, Zeebaree SR, Zebari RR, Sadeeq MA, Sallow AB, Mohsin S, et al. A comprehensive survey of 5G mm-wave technology design challenges, Asian Journal of Research in Computer Science. 2021;1-20.
76. Kareem FQ, Zeebaree SR, Dino HI, Sadeeq MA, Rashid ZN, Hasan DA, et al. A survey of optical fiber communications: Challenges and processing time influences, Asian Journal of Research in Computer Science. 2021;48-58.
77. Zeebaree S, Zebari I. Multilevel client/server peer-to-peer video broadcasting system, International Journal of Scientific & Engineering Research. 2014;5:260-265.
78. Boem F, Gallo AJ, Raimondo DM, Parisini T. Distributed fault-tolerant control of large-scale systems: An active fault diagnosis approach, IEEE Transactions on Control of Network Systems. 2019;7:288-301.
79. Ledmi A, Bendjenna H, Hemam SM. Fault tolerance in distributed systems: A survey, in 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS). 2018;1-5.

80. Bravo M, Rodrigues L, Neiheiser R, Rech L. Policy-based adaptation of a byzantine fault tolerant distributed graph database, in 2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS). 2018;61-71.
81. Abdullah SMSA, Ameen SYA, Sadeeq MA, Zeebaree S. Multimodal emotion recognition using deep learning, Journal of Applied Science and Technology Trends. 2021;2:52-58.
82. Sadeeq MA, Zeebaree S. Energy management for internet of things via distributed systems, Journal of Applied Science and Technology Trends. 2021;2:59-71.
83. Arafa Y, Barai A, Zheng M, Badawy AHA. Fault tolerance performance evaluation of large-scale distributed storage systems HDFS and Ceph case study, in 2018 IEEE High Performance extreme Computing Conference (HPEC). 2018;1-7.
84. Omer MA, Zeebaree SR, Sadeeq MA, Salim BW, Mohsin SX, Rashid ZN, et al. Efficiency of malware detection in android system: A survey, Asian Journal of Research in Computer Science. 2021;59-69.
85. Maulud DH, Zeebaree SR, Jacksi K, Sadeeq MAM, Sharif KH. State of art for semantic analysis of natural language processing, Qubahan Academic Journal. 2021;1:21-28.
86. Sadeeq MM, Abdulkareem NM, Zeebaree SR, Ahmed DM, Sami AS, Zebari RR. IoT and Cloud computing issues, challenges and opportunities: A review, Qubahan Academic Journal. 2021;1:1-7.
87. Hasan DA, Hussan BK, Zeebaree SR, Ahmed DM, Kareem OS, Sadeeq MA. The impact of test case generation methods on the software performance: A review, International Journal of Science and Business. 2021;5:33-44.
88. Jacksi K, Ibrahim RK, Zeebaree SR, Zebari RR, Sadeeq MA. Clustering documents based on semantic similarity using HAC and K-mean algorithms, in 2020 International Conference on Advanced Science and Engineering (ICOASE). 2020;205-210.
89. Sadeeq MA, Abdulazeez AM. Neural networks architectures design, and applications: A review, in 2020 International Conference on Advanced Science and Engineering (ICOASE). 2020;199-204.
90. Ageed ZS, Ibrahim RK, Sadeeq M. Unified ontology implementation of cloud computing for distributed systems, Current Journal of Applied Science and Technology. 2020;82-97.
91. Fochi V, Caimi LL, da Silva MH, Moraes FG. Fault-tolerance at the management level in many-core systems, in 2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI). 2018;1-6.
92. Yusuf SI, Junaidu SB. Parallel and distributed intra query transient fault tolerance model via parity checking, in 2018 14th International Conference on Electronics Computer and Computation (ICECCO). 2018;206-212.
93. Deng C, Che WW, Shi P. Cooperative fault-tolerant output regulation for multiagent systems by distributed learning control approach, IEEE Transactions on Neural Networks and Learning Systems. 2019;31:4831-4841.
94. Ghosh P, Eisele S, Dubey A, Metelko M, Madari I, Volgyesi P, et al. On the Design of fault-tolerance in a decentralized software platform for power systems, in 2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC). 2019;52-60.
95. Loutskii H, Volokyta A, Rehida P, Honcharenko O, Ivanishchev B, Kaplunov A. Increasing the fault tolerance of distributed systems for the Hyper de Bruijn topology with excess code, in 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT). 2019;1-6.
96. Mahjoubi A, Zeynalpour O, Eslami B, Yazdani N. LBFT: Load Balancing and Fault Tolerance in distributed controllers, in 2019 International Symposium on Networks, Computers and Communications (ISNCC). 2019;1-6.
97. Pareek S, Sharma N. Fault tolerance in distributed database management systems-improving reliability with RAID, in 2019 Innovations in Power and Advanced Computing Technologies (i-PACT). 2019; 1-5.
98. Shi J. Cooperative Fault-tolerant formation control for nonlinear multi-agent systems with actuator faults, in 2019 Chinese Control Conference (CCC). 2019;4890-4895.

99. Zhang P, Xue, H Gao S. Fault-tolerant control for multi-agent with actuator fault, in 2020 39th Chinese Control Conference (CCC). 2020;4255-4260.

100. Perez I, Goodloe A, Edmonson W. Fault-tolerant swarms, in 2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT). 2019;47-54.

Peer-review history:
The peer review history for this paper can be accessed here:
https://www.sdiarticle4.com/review-history/73754