Received November 20, 2021, accepted December 13, 2021, date of publication December 30, 2021, date of current version February 3, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3139730

Fault-Tolerant Load Balancing in Cloud Computing: A Systematic Literature Review

VAHID MOHAMMADIAN[1], NIMA JAFARI NAVIMIPOUR[2,3], MEHDI HOSSEINZADEH[4], AND ASO DARWESH[5]
[1]Department of Computer Engineering, Qeshm Branch, Islamic Azad University, Qeshm, Iran
[2]Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz 5157944533, Iran
[3]Department of Computer Engineering, Kadir Has University, 34083 Istanbul, Turkey
[4]Pattern Recognition and Machine Learning Laboratory, Gachon University, Sujeonggu, Seongnam 13120, Republic of Korea
[5]Information Technology Departments, University of Human Development, Sulaimaniyah 35147-99442, Iraq

Corresponding author: Mehdi Hosseinzadeh (mehdi@gachon.ac.kr)

ABSTRACT Nowadays, cloud computing is growing daily and has been developed as an effective and flexible paradigm in solving large-scale problems. It has been known as an Internet-based computing model in which computing and virtual resources, such as services, applications, storage, servers, and networks, are shared among numerous cloud users. Since the number of cloud users and their requests are increasing rapidly, the loads on the cloud systems may be underloaded or overloaded. These situations cause different problems, such as high response time and power consumption. To handle the mentioned problems and improve the performance of cloud servers, load balancing methods have a significant impact. Generally, a load balancing method aims to identify under-loaded and overloaded nodes and balance the load among them. In the recent decade, this problem has attracted a lot of interest among researchers, and several solutions have been proposed. Considering the important role of fault-tolerant in load balancing algorithms, there is a lack of an organized and in-depth study in this field yet. This gap prompted us to provide the current study aimed to collect and review the available papers in the field of fault tolerance load balancing methods in cloud computing. The existing algorithms are divided into two categories, namely, centralized and distributed, and reviewed based on vital qualitative parameters, such as scalability, makespan, reliability, resource utilization, throughput, and overhead. In this regard, other criteria such as the type of detected faults and adopted simulation tools are taken into account.

INDEX TERMS Cloud computing, load balancing, systematic review, fault tolerance, SLR.

I. INTRODUCTION

With the recent technical and scientific progress in artificial intelligence, Internet of Things (IoT), and cloud computing, research efforts are moving towards facilitating communication among different devices [1]. Cloud computing as an innovative paradigm develops an environment to provide unlimited virtual applications and resources via the Internet, which are ubiquitously accessible, rapidly provisioned, customizable, and available on-demand [2]. Computing resources such as servers, storage, and computer networks are available in four different forms, including Software as a Service (SaaS) [3], Infrastructure as a Service (IaaS) [4], Platform as a Service (PaaS) [5], and Expert as a Service (EaaS) [6], [7]. In SaaS, the applications are deployed in the cloud and the cloud clients are allowed to access them through the Internet [8]. IaaS provides virtual private network hardware and software, hardware and software firewalls, storage, processing power, and network infrastructure [9]. PaaS permits clients to use predefined tools prepared by cloud providers to deploy web applications and other programming software [10]. EaaS provides expert knowledge and human resources as a service [6].

A. PROBLEM STATEMENT

Since the number of cloud providers is growing day by day, and the load on the cloud servers is also increasing, the scheduling problem becomes a key issue in this environment and the cost also increases [11], [12]. During tasks scheduling on Virtual Machines (VMs), there may happen

The associate editor coordinating the review of this manuscript and approving it for publication was Ghufran Ahmed.

a situation that some of the VMs are over-utilized while others remain under-utilized [13]. Therefore, in order to organize the scheduling procedure and balance the loads on servers, an efficient scheduler is required. Load balancing has been known as an efficient method to distribute a system's total load among involved VMs to balance the load among them [14], [15]. In fact, this technique ensures that each VM performs an approximately equal number of tasks. It improves performance parameters such as throughput, response time, reliability, and resource utilization and prevents system bottlenecks, which may happen due to load imbalance [16]. The schedulers and load balancers in the cloud environment may be crashed or failed due to different reasons. This causes non-availability of the system and loss of credibility to cloud computing. Generally, the faults in cloud computing can occur in four main places, namely, among service providers, inside of providers, between provider and client, and among clients. Failures in service providers can lead to loss of money and more power consumption. Failures in clients can raise the response time for required services [17], [18]. Fault tolerance is considered a vital and key feature of cloud computing. It refers to offering cloud services in the presence of faults and enables system to discover the type and the location of the fault and attempt to tolerate it [19].

B. RELATED WORKS AND OUR MOTIVATION

In the recent decade, many researchers have studied the load balancing methods in the cloud environment and offered a solid foundation for understanding the diverse sides of this issue. This section reviews the previous survey studies and specifies our motivation for presenting this paper.

A survey on multiple algorithms for load balancing in cloud computing has been done in [20], in which the advantages and shortcomings of the reviewed algorithms have been specified, and available challenges have been discussed to improve these algorithms. This paper explicitly has explored technical details, but future research directions have not been discussed. Also, some optimization algorithms such as Ant Colony Optimization (ACO), PSO, GA, and ABC for load balancing problems have been reviewed in [21]. This paper shows that the reviewed algorithms have good performance compared to traditional ones in terms of makespan, response time, etc. Nevertheless, this survey paper is limited to published papers from 2012 to 2015 and is not written in a systematic structure.

In another work, the authors in [22] have reviewed existing load balancing algorithms for cloud computing such as ACO, Round Robin (RR), Honey bee, Carton, Max-Min, and Min-Min. They have discussed the advantages and weaknesses of the algorithms and compared them with each other based on vital parameters such as response time, overhead, fault tolerance, throughput, complexity, resource utilization, fairness. However, a few papers are reviewed, and the procedure of paper selection is not clear. Also, the existing load balancing methods and approaches, as well as essential requirements for providing efficient load balancing techniques for cloud environments, have been reviewed by researchers in [23]. In this work, a new classification of load balancing techniques has been presented, in which the selected techniques have been evaluated and compared with each other based on suitable parameters.

Furthermore, the researchers of [24] have reviewed the load balancing techniques in two classes, including dynamic and hybrid approaches. They have presented the main features of these techniques, their challenging problems, advantages, and weaknesses. However, the static techniques have been ignored. Moreover, the authors in [25] have reviewed the existing scheduling methods, purposes, and load balancing techniques. The selected techniques have been classified into four classes including heuristic-based, genetic, agent-based, and dynamic. However, their research has not been written in a systematic way, and many important papers in this area have been ignored.

A remarkable survey paper has been proposed in [26], in which the existing load balancing techniques have been reviewed in seven categories, including workflow specific, network-aware, application-oriented, general, agent-based, natural phenomena, and Hadoop map-reduce. Some techniques have been discussed and analyzed in each category based on significant load balancing metrics, such as resource utilization, throughput, scalability, makespan, response time, and energy. Moreover, some future works and research directions to offer efficient techniques have been suggested. Nevertheless, fault tolerance as an essential factor in load balancing has been ignored, and existing works in this field have not been covered.

The authors of [27] have presented a thorough examination of conventional resource scheduling algorithms, emphasizing the technical characteristics and challenges of cloud computing. The issues faced by cloud computing in terms of service provider success, customer satisfaction, resources consumption, high computation cost, and high energy consumption of distributed data centers have been recognized

A review of existing tools and methods for load balancing in cloud computing has been presented in [28]. The reviewed methods have been assessed based on some metrics and parameters such as scalability, resource utilization, throughput, reaction time, overhead, and performance. However, newly published papers have been neglected. Also, the proposed survey paper in [29] has reviewed the existing techniques in three main classes, including meta-heuristic, heuristic, hybrid. It has specified the main pros, cons, and optimization measures of each technique. However, these survey papers have ignored the recently published papers.

The authors of [30] have presented a systematic study of current research in the field of workflow scheduling in cloud computing with the goal of identifying distinct trends in the problem. They have classified methods into three groups, including heuristic, meta-heuristic, and hybrid schemes, and explored different factors such as workflow types and QoS constraints, and specified practical impacts and multi-disciplinary applications.

TABLE 1. Some related studies in load balancing in cloud computing.

Reference	Main contributions	Considered parameters								
		Availability	Scalability	Reliability	Response time	Overhead	Throughput	Resource utilization	Makespan	Type of detected faults
[20]	Reviewing load balancing mechanisms in cloud computing without any categorization.	✓	✗	✗	✓	✓	✓	✓	✓	✗
[21]	Reviewing meta-heuristic algorithms proposed for load balancing in cloud computing and highlighting their main pros and cons.	✗	✓	✓	✓	✓	✗	✗	✓	✗
[22]	Categorizing load balancing mechanisms into two classes, namely, dynamic and static, and reviewing selected methods considering key load balancing metrics.	✗	✗	✗	✓	✓	✓	✓	✗	✗
[23]	Proposing a new classification of load balancing methods and evaluating existing works based on suitable load balancing parameters.	✗	✓	✗	✓	✓	✓	✗	✗	✗
[24]	Offering a systematic study of dynamic load balancing approaches, specifying the main pros and cons of the selected techniques, and highlighting open issues and future trends.	✗	✓	✗	✓	✗	✓	✓	✓	✗
[25]	Providing a brief review of scheduling and load balancing techniques and overviewing selected mechanisms in four main groups, including heuristic-based, genetic, agent-based, and dynamic.	✗	✗	✗	✗	✗	✗	✓	✗	✗
[26]	Offering a new classification of load balancing approaches, reviewing selected methods based on main parameters, and outlining future works and research directions.	✗	✓	✗	✓	✗	✓	✓	✓	✗
[27]	Examining conventional resource scheduling algorithms with an emphasis on the technical characteristics and challenges of cloud computing	✗	✗	✗	✗	✗	✓	✓	✗	✗
[28]	Discussing load balancing tools and methods and evaluating selected methods based on key parameters.	✗	✓	✗	✓	✓	✓	✓	✗	✗
[29]	Overviewing existing load balancing methods in three classes, including meta-heuristic, heuristic, hybrid.	✓	✗	✓	✓	✗	✓	✓	✓	✗
[30]	Reviewing current research in the field of workflow scheduling in cloud computing with the goal of identifying distinct trends in the problem	✗	✗	✗	✓	✓	✓	✓	✓	✓
[31]	Highlighting existing research challenges, recent efforts, configuration parameters, and tools.	✓	✗	✗	✓	✓	✓	✓	✓	✗
[32]	Describing, comparing, and assessing the published works between 2015 and 2018.	✗	✓	✗	✓	✗	✓	✓	✓	✗
[33]	Highlighting the role of meta-heuristic algorithms in task scheduling in cloud computing	✓	✓	✓	✓	✓	✓	✓	✓	✗
Our survey	Categorizing existing works between 2010 and 2020 into two main classes, namely, centralized and distributed, reviewing them based on important qualitative parameters, specifying challenging problems, and suggesting future research directions.	✓	✓	✓	✓	✓	✓	✓	✓	✓

The existing research challenges related to loading balancing have been checked in [31], in which some of the previous works have been reviewed, and their used methods, configuration parameters, and tools have been highlighted as well. Moreover, the survey paper proposed in [32] has specified, described, compared, and assessed the published works between 2015 and 2018.

The researchers of [33] have highlighted the role of meta-heuristic algorithms in task scheduling in cloud computing. They have presented the rudimentary

TABLE 2. Abbreviation table.

Abbreviation	Definition	Abbreviation	Definition
ALO	Ant lion optimizer	PM	Physical machine
ABC	Artificial bee colony	PaaS	Platform as a service
DBPS	Deadline based preemptive scheduling	QoS	Quality of service
EaaS	Expert as a service	RQ	Research question
GA	Genetic algorithm	RR	Round robin
HMC	Heterogeneous mobile cloud	SLA	service level agreement
IaaS	Infrastructure as a service	SaaS	Software as a service
IoT	Internet of Things	SLR	Systematic literature review
MRR	Modified round-robin	TLBC	Throttled load balancing approach for cloud
PSO	Particle swarm optimization	VM	Virtual machine

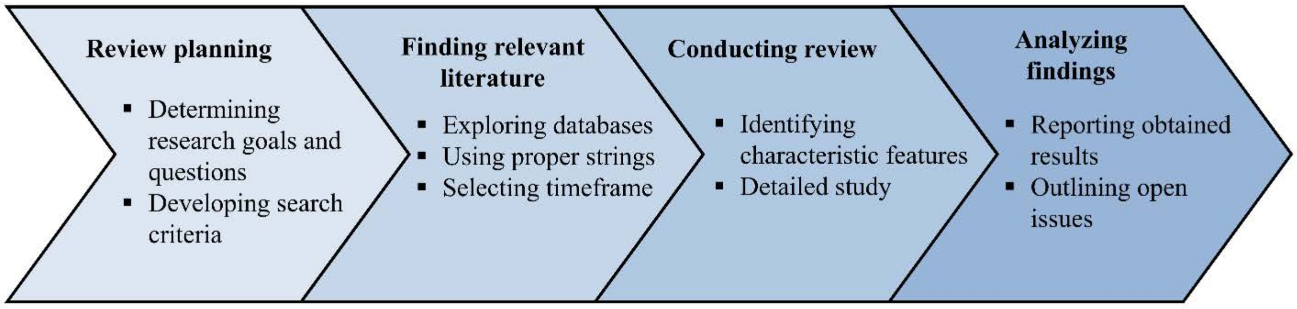
Review planning
▪ Determining research goals and questions
▪ Developing search criteria
Finding relevant literature
▪ Exploring databases
▪ Using proper strings
▪ Selecting timeframe
Conducting review
▪ Identifying characteristic features
▪ Detailed study
Analyzing findings
▪ Reporting obtained results
▪ Outlining open issues

FIGURE 1. Adopted research methodology.

notions of cloud task scheduling and categorized works into various groups based on scheduling constraint, resource-task mapping scheme, and primary objective of scheduling.

Our observation and search indicate that there is not a detailed and organized study about the current fault tolerance load balancing techniques in the literature. Therefore, by adopting a systematic manner, we attempt to cover this gap. For more illustration, Table 1 illustrates a comparison of the reviewed papers, in which the main contributions of each paper and considered parameters by them are specified. Obviously, the current paper compared to other surveys, extensively covers all the major aspects of the fault tolerance load balancing problem. Moreover, fault tolerance plays a central role in load balancing methods in the cloud environment, but none of the discussed papers is a systematic study. To cover this gap, we aim to offer an organized and thorough study of fault tolerance load balancing techniques, which highlights the effective works in this field, provides an abreast comparison of them, specifies challenging problems, and finally, outlines future research directions in this field. Concisely, the main aims of the current study are:

- Clarifying that how a systematic methodology can be conducted in this field.
- Categorizing and studying fault tolerance load balancing techniques in two main classes, centralized and distributed, and specifying their key advantages and disadvantages.
- Highlighting challenging problems and open issues in this field to improve previous works.

C. ORGANIZATION

The content of the current paper is prepared in seven sections. The adopted review method is described in the next section. Related terminologies and rudimentary concepts are presented in Section 3. The selected methods are reviewed in Section 4. Section 5 reports the research result, presents a side-by-side comparison of the reviewed techniques, as well as gives a statistical analysis of them. Section 6 outlines open issues and gives some hints for future trends, and finally, Section 7 concludes the paper. The existing abbreviations of the paper are defined in Table 2.

II. REVIEW METHOD

The current paper is conducted following a Systematic Literature Review (SLR) method. Generally, the SLR aims to provide a detailed outline of available works about a specific subject [34], [35]. In order to specify the challenges, research directions, and concerns, all the available techniques related to a specific problem are evaluated in a detailed manner. The SLR method is used in this article to provide a comprehensive review of fault tolerance load balancing mechanisms in cloud computing. As specified in Figure 1, the adopted methodology comprises the following phases. The first phase, which is described in the next subsection, specifies the research objectives and questions. In the second phase, the articles are selected based on considered criteria. In the third phase, a detailed study regarding existing works is presented. Finally, in the last phase, the research results are reported as well as open issues, and some outstanding recommendations for further research are presented.

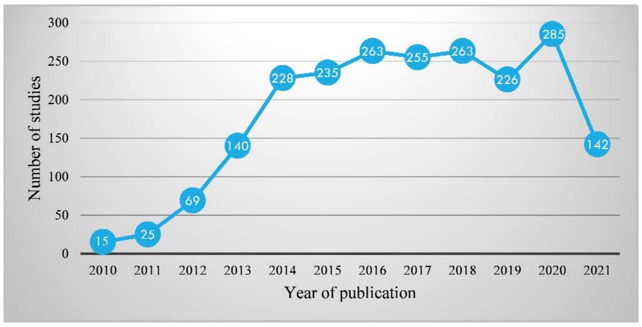

FIGURE 2. Dispersion of published papers by year of publication (after round 1).

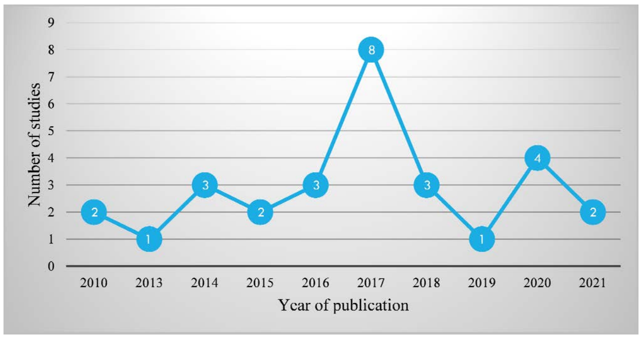

FIGURE 3. Dispersion of selected papers by year of publication (after round 3).

A. REVIEW PLANNING

This section clarifies some research questions that are anticipated to be found while reviewing state-of-the-art methods. Considering the importance of the selected subject and the lack of systematic work in this field, the central goal of this research is to handle the following RQs.

RQ1: What are the main concerns and challenges in developing fault tolerance load balancing approaches?

RQ2: What are the qualitative parameters to evaluate fault tolerance load balancing methods?

Which simulation tools are most often used to implement fault tolerance load balancing methods?

RQ4: What kinds of faults have been addressed in current works?

RQ5: What are the future trends and open issues?

B. FINDING RELEVANT LITERATURE

In order to review the fault tolerance load balancing methods in cloud computing, the authors searched scientific databases, such as IEEE Xplore,[1] Springer link,[2] Science Direct,[3] and Google Scholar[4] using the following terms: "cloud" AND ("load balancing" OR "load balance" OR "load balanced"). Scientific papers published between 2010 and 2021 were selected. Then, some results were removed to ensure that this study would only include data from high-quality publications and papers, including journals and conferences studies. Generally, the process of paper selection is performed in three rounds:

Round 1: An automatic search process is performed based on selected keywords in the mentioned scientific databases; as a result, 2146 studies are found from conferences, journals, and books. The distribution of the studies over the year of publication is illustrated in Figure 2.

Round 2: In order to select high-quality publications, some criteria are adopted. The review articles, non-English papers, working papers, reports, and editorial notes are excluded. Finally, 735 papers are considered for further analysis.

Round 3: The authors read the remaining studies carefully from Round 2 and selected 29 articles. The selected articles are eligible for review and are concerned with fault tolerance load balancing in cloud computing. The distribution of the selected papers by year of publication is illustrated in Figure 3.

[1]Ieeexplore.ieee.org
[2]Link.springer.com
[3]Sciencedirect.com
[4]Scholar.google.com

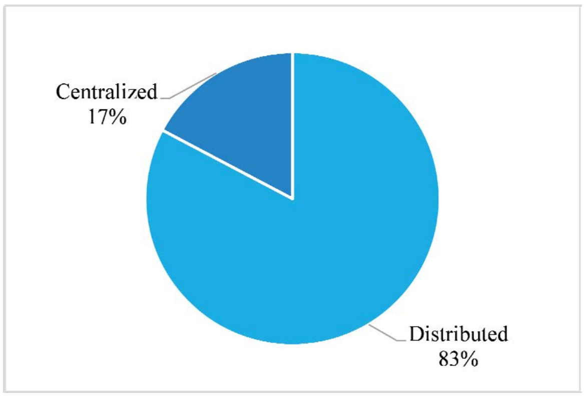

FIGURE 4. Dispersion of papers in each group.

C. CONDUCTING THE REVIEW

After discovering the related studies, an organized and detailed study of the selected approaches is conducted, aiming to find and specify the characteristic features of each work. In this regard, the authors classified the selected papers into two groups, including centralized and distributed. As shown in Figure 4, 24 papers out of 29 (83%) are related to distributed methods (Table 3), and the remaining five papers (17%) belong to centralized methods (Table 4).

D. ANALYZING FINDINGS

Once the existing works are reviewed and their main characteristic features are specified, the obtained results are reported under the following headings. Moreover, available challenges and problems faced by reviewed works, as well as some interesting future research directions, are listed.

- *Dynamic or static*
- *Heuristic or non- heuristic*
- *Adopted basic approach*
- *Adopted simulation tools and type of detected faults*
- *The significance of considered qualitative metrics*

TABLE 3. Details of distributed approaches.

Research	Year	Journal or conference name
[71]	2010	Cluster computing
[72]	2010	Computer architecture news
[73]	2014	IEEE transactions on computers
[74]	2014	Advances in engineering and technology
[75]	2014	Recent trends in information technology
[76]	2015	Computer science trends and technology
[77]	2016	Knowledge-based engineering and innovation
[78]	2016	Intelligent automation and soft computing
[79]	2016	Knowledge-based systems
[80]	2017	Computer communications and networks
[81]	2017	Computer engineering
[82]	2017	Engineering development and research
[83]	2017	Internet of things, data and cloud computing
[84]	2017	Advanced Intelligence Paradigms
[85]	2017	Journal of computer and system sciences
[86]	2017	Information sciences
[51]	2018	Neural computing and applications
[87]	2018	Advanced research journal in science, engineering and technology
[88]	2018	Advances in intelligent systems and computing
[89]	2019	Cluster computing
[90]	2020	Web research
[91]	2020	Electrical and computer engineering innovations
[92]	2020	Arab journal of information technology
[93]	2021	Computing and digital system

TABLE 4. Details of centralized approaches.

Research	Year	Journal or conference name
[99]	2013	Emerging research in management and technology
[95]	2015	Applied engineering research
[96]	2017	Research journal of engineering and technology
[97]	2020	Computers and applications
[98]	2021	Concurrency and computation: practice and experience

III. BACKGROUND

The rudimentary concepts and related terminologies about cloud computing, fault tolerance, and load balancing in cloud computing are presented in this section. First, the characteristics of cloud computing are described. Then, the role of load balancing and fault tolerance in cloud computing is explained.

A. CLOUD COMPUTING CHARACTERISTICS

Cloud computing is an on-demand, expandable, cost-effective, virtualized, and all-time available model. It has been known as an effective technology in parallel computing, which offers a range of services such as virtualized resources, metered resource usage, on-demand computing resources access, dynamic and elastic scaling, and ubiquitous computing that can be released and provisioned without effort [9], [36]. In this regard, cloud resources and services are facing significant uncertainty during provisioning. Uncertainty may be offered in various components of the storage, communication, and computational process [37]. To handle uncertainty in an efficient way, the current computing models can be adapted to this evolution, as well as novel resource management strategies can be designed. The management of cloud infrastructure is a challenging task. Cost-efficiency, performance stability, QoS, security, and reliability are vital problems in these systems [38], [39]. Generally, the following five main characteristics should be considered in cloud computing.

- *Measuring service:* To control and maximize the use of cloud resources, cloud computing systems are capable of using metering abilities related to a specific service type. As a matter of fact, the consumption of resources can be tracked, measured, and reported to create transparency for service clients and providers [40].
- *Rapid elasticity:* Cloud computing abilities can be quickly released and elastically provisioned. These abilities often appear to be unlimited and can be bought at any time in any quantity [41].
- *Broad network access:* All the cloud services are accessible through the Internet and support various client platforms [41].
- *Resource pooling:* Computing resources, including memory, storage, processing, and network bandwidth can be combined to become a multi-tenant model [42].
- *On-demand self-service*: The cloud clients are capable of utilizing computing capabilities independently and without human intervention 40].

B. FAULT TOLERANCE AND LOAD BALANCING IN CLOUD COMPUTING

Load balancing has been known as a challenging issue and major problem in cloud computing. In order to keep the cloud system steady without being overloaded or under-loaded and improve resource utilization, it should be ensured that all computing resources are distributed over servers effectively. The load can be CPU, memory, and network loads [43], [44]. Different load balancing algorithms have solved this problem in the recent decade. A brief description of existing load balancers is provided below.

- *Hardware load balancer:* It is a physical device that handles network servers and distributes web traffic across several network servers. A hardware load balancer needs at least two VMs. The system administrator configures it with specific rules to guarantee the best performance 15].
- *Network load balancer:* A network load balancer effectively balances TCP traffic that operates on layer 4 of the OSI model or network layer. Each subnet is assigned a static IP address that can be used by applications as the front-end IP of the balancer. Network traffic is distributed among several VMs within a cluster to avoid overloading [45], [46].
- *HTTP(S) load balancer:* It operates on the application layer and routes network traffic and web visitors across all web application clusters using session ids, cookies, and HTTP headers [47].
- *HAProxy load balancer:* It operates on Layer 7 of the OSI model and is widely employed in ALOHA load balancer. The ALOHA load balancer offers reliable and scalable infrastructures. It has developed several open-source load balancing software utilizing HAProxy [48].
- *Classic load balancer:* It has been designed for the Elastic Cloud Compute (EC2) classic network applications and works at both the request and connection levels [49].
- *Elastic load balancer:* It is also known as AWS load balancer, in which the incoming tasks are distributed over multiple Amazon EC2 instances [50].

Another principal challenge in cloud computing is fault tolerance. It is the ability of the cloud scheduler and load balancer to protect and safeguard the delivery of tasks even with the existence of failures in the clouds system [51]. Fault tolerance aims to obtain dependability and robustness in a cloud system. Generally, fault tolerance mechanisms can be classified into two main groups, reactive methods and proactive methods.

Reactive fault tolerance: Reactive fault tolerance policies decrease the influence of failures when the faults or failures occur. This technique makes the system more robust. In other words, it is known as an on-demand fault tolerance [19]. Some of the important approaches based on this policy are described in the following.

- *Checkpointing/restart:* These techniques continuously store the states of tasks execution. In case of any failure, tasks are restarted from the last stored state instead of restarting from the beginning. Portability, transparency, and scalability are the desired features of any checkpoint restart approach. Due to the dual applicability of checkpoint/restart techniques, these kinds of techniques have found great applicability in fault-tolerant systems. In fact, these techniques can be utilized as both auxiliary as well as stand-alone fault tolerance methods. Considering the failure rates of the system components, the frequency of taking checkpoints can be controlled to optimize the overhead [52].
- *Replication:* The involved tasks are operated on multiple execution instances. In case of any instance failures, the execution of tasks remains continuous in other instances.
- *Job migration:* In this method, the tasks that are facing any faults can be migrated to another machine [53].
- *Task resubmission:* In case of any failure, tasks are resubmitted to the different or same resource at run time [53].

Proactive fault tolerance: Prediction forms the core of proactive fault tolerance algorithms [54]. Indeed, proactive fault tolerance predicts the faults proactively and swaps the suspected components by valid components [55].

- *Software rejuvenation:* This method is specially used and planned for a periodic reboot of the system [56].
- *Self-healing:* The self-healing method is a characteristic of a system that permits it to automatically discover and reform hardware and software faults. These kinds of systems are formed of multiple components that are deployed on multiple VMs [57].
- *Preemptive migration:* In this method, an application is continually observed and examined [58].

Generally, the major types of faults that may occur in the cloud environment can be categorized into two groups, which are described in the following.

- *Network faults:* Include faults that occurred in a network due to various reasons such as packet loss, packet corruption, destination failure, link failure, and network partition [59].
- *Physical faults:* These faults refer to faults in storage, memory, and CPUs [59].

IV. FAULT TOLERANCE LOAD BALANCING APPROACHES

This section reviews current techniques about fault tolerance load balancing in cloud computing. As a matter of fact, a clear trend of fault tolerance load balancing is provided by reviewing valid and effective techniques in this field. The techniques' innovation, differences, advantages, and disadvantages are also presented. According to the suggested classification in [60], depending on where the load balancing decisions are made, these methods can be categorized into two groups, distributed and centralized. In the centralized mode, there is a central node that has a global view of the system's state and is responsible for managing the compute load of nodes, while, in distributed load balancing methods, all the nodes are involved in making load balancing decisions. This study has discussed the selected papers in two groups, centralized (5 articles) and distributed (24 articles). Several performance metrics are required to evaluate each load balancing technique, determine which technique is better, and realize the benefits and drawbacks of each technique. System stability is improved by balancing the load among the available virtualized resources. A better scheduler is required to have a better load balancing technique. There are *n* VMs and *m* input tasks. The assignment of these m tasks to n VMs affects several system performance metrics [61]. In this respect, various qualitative metrics are used in load balancing techinques, which are defined below:

- *Availability:* It is defined as the probability that a system functional correctly during a specific time in the stated situation [62], 63].
- *Scalability:* This parameter refers to the ability of a load balancing algorithm to perform uniformly in a system according to the requirements upon growing the number of objects [26].
- *Reliability:* It specifies that how a cloud computing system consistently offers its services without failure and interruption. In fact, it refers to the ability of a system to perform a required function correctly under stated conditions for a stated time period [64], [65].
- *Response time:* It is defined as the time taken to respond/reply to a specific algorithm [66].
- *Overhead:* This parameter refers to the amount of overhead involved while implementing a load balancing algorithm [67].
- *Throughput:* It is defined as calculating the number of processes or tasks completed within a stipulated time period [68].
- *Resource utilization:* It specifies that what degree of VMs uses the tools. In fact, it determines a part of accessible services among the total available resources [69].
- *Makespan:* It is defined as the time taken to process a set of tasks for its complete execution [70].

A. REVIEW OF DISTRIBUTED APPROACHES

Utilizing the self-healing technique, researchers of [71] have developed a proactive fault tolerance framework, called SHelp, an improved version of the earlier proposed framework, called ASSURE [72]. ASSURE presented the idea of rescue points, which are points inside the application's code where a specific set of programmer-anticipated problems can be handled. These rescue points can be reused to deal with unexpected faults. By periodically capturing application checkpoints, the ASSURE framework keeps an execution log. Lightweight instrumentation methods are used for fault detection and system monitoring. A triage system is used to perform fault analysis, which deploys the shadow of the application. When a fault is found, the application is transferred to the triage system from its most recently checkpointed state, and the issue is replicated there in order to identify an appropriate rescue point. The SHelp design varies from ASSURE in terms of how the rescue points are selected. ASSURE searches for an acceptable rescue point by traversing a rescue-trace graph. It leads to increasing the overhead of fault tolerance. In contrast, in the SHelp framework, a weight to each rescue point is assigned. The weight of each point is initially set to zero, but it changes proportionately with the number of times a given rescue point is applied. When a fault is detected, the rescue spots are searched in order of decreasing weight. The performance of SHelp has been evaluated by implementing it on Linux and tested over various web server applications. Due to the adoption of weighted rescue points, which significantly reduces the amount of time spent searching, the SHelp framework operates more quickly and with lower overhead than ASSURE.

The authors of [73] have suggested a fault-tolerant scheduling strategy for real-time tasks in virtualized clouds, in which the primary backup method is utilized for fault tolerance. Real-time controller, backup copy controller, and resource controller are three main components of a scheduler that receives user tasks from an input buffer. When a task is submitted, the backup copy controller generates a backup. In order to accomplish the task before the deadline, the resource controller searches for two virtual resources in different hosts. The task is rejected if the required resources are not found. Otherwise, both instances of the task are scheduled on the respective resources. The performance of the proposed mechanism has been proven through simulation experiments over Google cloud trace logs and randomly generated workloads.

A load-balancing method using the ACO algorithm has been offered in [74]. The researchers have focused on balancing the load of the system while trying to keep the reliability of the system by generating a fault-tolerant [68] system. The suggested fault management system has two main processes, fault detection and fault handling. For fault detection, a fault detector has been applied to the system, which works based on the scholastic Petri nets algorithm. To handle the faults and increase the reliability of the system, a modified algorithm of ACO with implementing checkpoints has been provided. Nevertheless, the proposed approach has not been compared to existing works.

The proposed mechanism in [75] performs load balancing by estimating the finish time of tasks before job allocation. In this regard, it considers both the current load of VMs and the time taken to finish the execution of tasks. During tasks allocation, when faults occur in VMs, the tasks are returned to the main controller and then allocated to another VM. To reach cost-effectiveness, the DBPS algorithm has been used, which minimizes user payments. Considering this algorithm, since jobs with a hard deadline have higher priority by pre-empting the soft deadline jobs, the completion time and cost are reduced. Moreover, to reach effective resource allocation, TLBC has been used. However, the suggested mechanism considers limited failure aspects.

As another distributed technique, the proposed load balancing mechanism in [76] balances the incoming loads from various hosts in a resource pool, as well as preserves the fault tolerance, availability, and reliability properties by maintaining the redundant copies of services in various hosts. The proposed approach maintains the status of all hosts, such as the number of VMs and its service number. Once a heavily loaded host is found, the approach tries to migrate VM to lightly loaded hosts. During VMs migration, the proposed approach ensures the fault-tolerant levels of the system; for instance, if a specific host is down for some reason, the redundant VM should respond to the request. Nevertheless, the proposed method has not been simulated.

Moreover, a load balancing architecture using fuzzy logic to decrease the energy consumption and increase fault tolerance has been proposed by the authors of [77]. They have designed three fuzzy inference engines to prioritize VMs and tasks aimed at repeating tasks. The suggested method improves reliability and throughput, but it has a high overhead.

The proposed adaptive fault tolerance method in [78] operates in two main phases. In the first step, various fault tolerance strategies are ranked based on the constraints provided by the user. The work has considered four fault tolerance strategies, including active, N-version programming, recovery block, and retry. User-provided constraints include resource consumption, failure rate, and response time. The optimal fault tolerance mechanism is then selected in the context of the user-provided constraint. The VMs are placed according to the mechanism chosen in the second phase.

Researchers of [79] have offered an immunological mechanism inspired scheduling algorithm is proposed for workflow in Cloud systems. It consists of four modules, including a surveillance unit that monitors possible faults for each VM in the resources pool, a memory unit that contains several rescheduling strategies, a response unit that triggered when a resource fault is detected, and it searches either the memory unit or the learning unit for a suitable rescheduling plan. To narrow the search scope in the learning unit, the available resources are grouped into various clusters. If none of the existing VMs can fulfill the Quality of Services, a new VM for the faulty resource is generated. The performance of the proposed framework has been evaluated through simulation on both randomly generated workflows and four real-world workflows, such as Montage, Epigenomics, CyberShake, and Inspiral. The results indicate that the proposed mechanism can provide effective rescheduling methods concerning resource failures and outperforms similar algorithms in various situations.

An energy-efficient and load-balanced distributed storage and processing system has been proposed by researchers in [80]. They have proposed a Heterogeneous Mobile Cloud (HMC) computing design, in which the computation and communication resources are utilized to support data processing and data storage services in a group of mobile devices. Generally, this work confirms that 1) the stored data are fault-tolerant, 2) the heterogeneity of devices is considered during task allocation and system-wide load balancing, 3) the computation and communication tasks are performed in an energy-efficient method. The proposed approach supports three main data operations, namely, data creation, data recovery, and data processing. During file creation, Reed-Solomon code is used to encode the file, and some data fragments are created. Then, data fragments are sent to a set of storage nodes. To recover and read the original file, k of the n data fragments from the network is searched and retrieved. This coding way ensures the stored data is fault-tolerant. Notwithstanding the good performance of the proposed method, it suffers from complex implementation.

A load-balancing method based on clustering and Bayes theorem with some constraints has been introduced in [81]. Aiming to reach a task deployment method with a global search capability regarding the performance of computing resources, the proposed method makes a limited constraint about all physical hosts. The clustering process is combined with the Bayes theorem to obtain optimal clustering of the physical hosts. The goal of the proposed system is to ensure that every computing resource can handle tasks quickly and effectively while improving resource utilization. In order to handle the system failures, a backup plan is prepared. The mechanism has decreased the number of task failures and improved throughput of the cloud data center, but limited experimentation remains a problem.

The researchers in [82] have suggested a load balancing approach, in which the CPU temperature has been considered to predict a problem on the PMs, and a migration algorithm has also been used to migrate VMs to some optimal PM. Considering the heterogeneous nature of cloud resources, the suggested mechanism has taken into account the heterogeneity of VMs. The incoming requests at the VM allocation stage are scheduled using Modified Round Robin (MRR) method, which efficiently avoids occurring faults at the initial stage. It allocates VMs to the hosts in a cyclic way, but before assigning them, it checks whether the same service type is already running in the host. The suggested algorithm is implemented and evaluated in the Cloudsim environment. The main goal of this algorithm is to preserve the fault tolerance level of services during VMs migration. It avoids allocating the VMs with the same service type to hosts. Nevertheless, limited experimentation as well as considering limited aspects of failure cannot prove the efficiency of the work.

Considering the particular feature of performance optimization within the cloud, the researchers of [83] have introduced a load balancing architecture based on the MapReduce concept. The suggested mechanism, by taking advantage of the MapReduce principles, holds the massive number of available resources to find the most appropriate load balancer regarding the requirements of users' requests. It improves fault tolerance and response time in the cloud. The main weakness of this method is limited experimentation.

Aiming to balance load across VMs, activate recovery process at the time of VMs failure, and decrease the power consumption of VMs, ant colony-based load balancing and fault recovery (ACB-LBR) algorithm has been proposed in [84]. The suggested algorithm uses the behavior of artificial ants for balancing tasks among VMs that leads to high throughput. Moreover, it recovers the lost resource at failure time and manages less power consumption, but it suffers from low scalability.

A fault-tolerant scheduling framework using pre-emptive migration for stream computing has been proposed in [85]. It operates by integrating four main work spaces, including user space, graph space, storm space, and hardware space. Users submit their data streams in the user space, which are then turned into Directed Acyclic Graphs (DAGs) in the graph space. The critical and non-critical paths of the DAGs are also specified in the graph space. A scheduling approach with a fault tolerance technique is used in the storm space. The hardware space includes a variety of data center resources. The arrival rate of the data streams is monitored constantly. A large fluctuation in the arrival rate can increase response time. In this situation, a vertex from the critical path is preemptively transferred to another computing node to maintain the minimum response time. The suggested architecture has been applied in Storm, an open-source distributed computing platform, and its reliability, response time, and throughput have been verified.

A fault-tolerant workflow scheduling algorithm using job migration and primary-backup methods has been proposed in [86]. Each workflow is modeled as a DAG, with vertices representing tasks and edges representing task dependencies. Instead of adopting a single deadline for a complete workflow, the workflow deadline is divided into task deadlines based on the size of each task and the number of tasks. Each workflow task contains two copies, primary and backup. If the primary fails, execution is continued at the backup server. Furthermore, if the backup of a task fails, it is migrated to another host but not to any of the hosts placing the predecessors of a task. The performance of the proposed fault tolerance framework has been evaluated through simulation on four real-world workflows, including. Montage, Epigenomics, CyberShake, and Inspiral.

The authors of [51] presented a dynamic clustering league championship algorithm scheduling approach with fault tolerance awareness to handle cloud task execution in a way that takes into account available resources and minimizes untimely task failure. It has been designed based on using elasticity and resource utilization as QoS objectives. It utilizes reactive mechanism, replication protocol, and Kafka technology. The experimental results indicate the superiority of the proposed method in terms of reducing the fault rate and makespan. However, it does not consider the reliability, execution cost, and throughput of the system.

Using the ACO algorithm, a novel approach to load balancing has been offered in [87] to control resource failure. The forward-backward ant mechanism, max-min rule, and checkpoint-based rollback recovery as main strategies have been used. The proposed method provides a dynamic load balancing method for cloud computing with less searching time. Not only does it improve the network performance, but it also handles tasks failures. However, simulation results are not presented.

In order to extend the single load balancer, the work in [88] has presented a fault-tolerant multiple synchronized parallel load balancing mechanism. It has a number of load balancers that are able to balance the tasks across multiple processors. These schedulers cooperate with each other for gathering information about the tasks in the input queue and tasks status. Also, the tasks are distributed to other processors in the data center based on processors' capabilities. The suggested mechanism decreases average overhead, but its efficiency cannot be verified with limited experimentation.

A novel technique for adaptive fault tolerance during load balancing in cloud computing has been proposed in [89]. It has presented a concept of fault management with an emphasis on the network and physical faults handling. Generally, the proposed work aims to develop effective cloud architecture in order to tolerate faults, suggest appropriate solutions to maintain data, and make the system more reliable and flexible.

A task scheduling approach based on the honeybee algorithm aiming to load balancing has been proposed in [90]. In order to minimize load redundancy, available tasks are sent to the most proper VMs. After assigning tasks, the state of VMs is predicted. Since the proposed algorithm prevents possible additional loads in VMs, load balancing among VMs is created. It decreases makespan and increases the degree of load balancing. Moreover, it tracks the task execution states in each VM to improve the system‘s reliability. VMs are selected based on their reliability, and they are removed based on their improper performance. In fact, the node that has had many failures recently compared to other nodes has less priority to receive tasks. Simulation outcomes illustrate that the suggested approach outperforms existing works in terms of average makespan, waiting time, and reliability. However, the scalability and overhead of the approach have not been evaluated.

Researchers in [91] have aimed to predict and avoid failure in High-Performance Computing (HPC) systems in cloud computing. The proposed approach includes four main modules, which are utilized to specify the hosts' state. It uses five key parameters to predict and prevent failures, namely, fan speed, voltage, number of users' requests, CPU utilization, and CPU utilization. When the system faces an alarm state, a failure may occur in the current host. Therefore, the most optimal host among available hosts is chosen, and the process-level migration is done. The proposed method, in comparison to existing works, has better performance in terms of response time, energy consumption, makespan, and task execution costs, but it has not been evaluated in terms of resource utilization.

A fault-tolerance load balancing approach based on resource load and fault index value has been presented in [92]. It runs in two stages, resource selection and task execution. In the first stage, suitable resources for tasks execution are selected. Suitable resources include the resources with the least resource load and fault index value. In the second stage, to save the task state, periodically checkpoints are set at various intervals based on the resource fault index. Obtained results from CloudSim indicate that the proposed algorithm has better performance in terms of overhead, throughput, makespan, and response time, but its low scalability remains a problem.

Finally, the researchers of [93] have developed a model of fault tolerance that is driven by SLAs formed between cloud providers and consumers. The suggested model involves two main stages. The first stage is based on the use of idles VMs according to selection methods. The second stage is based on advanced QoS degradation operations as well as VM selection methods. The advanced degradation operation consists of optimal combinations of VMs distribution among customers, which results in the avoidance of SLA violation penalties. The suggested fault tolerance model includes three methods, fault tolerance with the strategy low capacity, fault tolerance with the strategy high capacity, and fault tolerance with the strategy max available. The developed general SLA representation model can be applied to various platforms. This model specifies the type of resources requested, the acceptable margin of degradation, and the various regular and irregular situations in which consumers use platform resources. Experimental findings indicate that the suggested fault tolerance model reduces the number of considered SLA violations.

B. REVIEW OF CENTRALIZED APPROACHES

Researchers in [94] have offered a dynamic and fault-aware load balancing technique, in which a load balancer as an intermediate node among cloud and clients manages the load of virtual machines. It receives the users' requests and checks the CPU utilization of each active server. If the CPU utilization is less than 80%, the dynamic load balancer admits load, and hence a response is delivered; otherwise, it shifts the request to another server with the lowest processor and memory utilization. The mechanism also checks the fault occurrence of servers. If any fault occurs, then the VMs will be shifted to another server whose memory and processor utilization is less than 80%. In this work, several fault tolerance methods have been used, such as replication and job migration. Also, it has considered important factors such as node selection, estimation and comparison of load, nodes interaction, and stability. Moreover, it has high scalability. However, simulation results are not presented.

Moreover, in [95], a fault-aware load balancing method in cloud storage has been offered, in which a load of storage servers is balanced, and the server capabilities and resources considering the faulty behavior of servers are utilized effectively. In this respect, the proposed algorithm considers four main parameters of servers, including fault rate, processing time, server service rate, and server request queue size. The experimental outcomes show that the suggested algorithm provides better fault tolerance and leverages the overall system performance. Moreover, obtained results show that more client requests are processed by the system without delay, and in case of overloading and failure, the load balancer distributes the requests accordingly to neighbor servers.

The researchers of [96] have improved cloud performance through load balancing with fault tolerance. They have used checkpoints and fault handlers to detect and remove the faulty nodes. Each VM has its own success ratio that is calculated based on its past performance. Considering success ratio and current load, the priority of each VM is calculated that is used as a deciding factor for the selection of suitable VMs. Limited types of faults are handled by this mechanism.

An adaptive method to predict and discover failures in the cloud system has been proposed in [97], in which a fuzzy logic-based algorithm is used to detect the faults, and a predictive approach is implemented to monitor the system. Job migration, timing check, and task resubmission have been utilized to increase the error tolerance. Also, checkpointing method is employed to reduce the time as well as processing costs of job migration. Moreover, to assess the nature of errors, a mechanism has been provided that offers a proper response to the diagnosed faults. In this respect, two fuzzy inference engines have been presented to balance the load when a fault occurs in the system. To detect faults, a fuzzy system with input parameters of throughput, workload, and response time has been designed, and in order to generate a proper response and increase the fault tolerance of the system, some parameters such as VM throughput rate, number of failed repeats of the current job, a current job waiting time, and node state have been considered. However, the scalability of the mechanism and involved computational overhead have not been checked.

A proactive fault tolerance model with load balancing has been presented by researchers of [98]. The suggested approach tolerates CPU faults of VMs in order to maximize the reliability and availability of the cloud computing infrastructure. CPU faults can arise during VM operation. The primary aim of the proposed model is to monitor changes in CPU utilization and to take action when a high value of CPU utilization is detected. VM migration has been selected as one of the constructive fault tolerance techniques used to decrease assigned host loads. To balance loads of VMs, a VM selection algorithm that chooses one of the VMs to migrate it from one cloud host to another is needed. Therefore, a new machine selection algorithm has been introduced called Maximum Faulty-one, which chooses VMs with the lowest faults. The model has been implemented on a physical cloud computing network comprised of five nodes, including a cloud controller node, a cloud network node, and three cloud compute nodes. The cloud controller node is the central management node involving some modules such as subroutines, historical server, and telemetry software in addition to cloud infrastructure modules. The cloud network node is in charge of VM connections, device servers, and controller.

V. RESEARCH RESULTS

In this section, the research results are summarized, and statistical analysis of the discussed load balancing techniques is provided. In this regard, the research questions, RQ2, RQ3, and RQ4 mentioned in section 1.4, are considered. In the previous section, the selected fault tolerance load balancing techniques were categorized into two groups and then analyzed based on important parameters, including reliability, response time, availability, scalability, overhead, throughput, resource utilization, and makespan. Moreover, some crucial cases such as the adopted basic approach, type of detected faults, and adopted simulation tools were considered. Table 5 shows more details about the discussed techniques. The obtained results of the research are outlined and presented in the rest of this section.

A. DYNAMIC OR STATIC

Load balancing mechanisms can be categorized into two main groups, dynamic and static. In the static methods, prior knowledge of the system status is needed, and the current condition of the system is not taken into account. In fact, earlier information about the structure and different parameters of the system, such as limits on the storage device, system nodes processing and memory, as well as correspondence time, are required. On the other hand, the dynamic methods consider the status and current condition of the system, and hence they are able to manage the dynamic load conditions. In these methods, the users' requests can be effectively handled with dynamic procedures. Although the dynamic methods offer better performance compared to static ones, it is difficult to develop an algorithm for a dynamic cloud environment. As specified in Figure 5, just 3% (one method [96]) of the methods have been done based on a static manner.

B. HEURISTIC OR NON-HEURISTIC

All of the reviewed approaches are categorized into two distinct groups including, heuristic-based and non-heuristic techniques. Heuristic-based methods refer to approaches that have used a heuristic or meta-heuristic algorithm either in a simple way or in a hybrid structure. As specified in Figure 6, 90% of the researchers have chosen a non-heuristic algorithm in their proposed innovation.

C. ADOPTED BASIC APPROACH

Figure 7 outlines the percentage of adopted basic fault tolerance techniques in the reviewed techniques. The research results confirm that constant monitoring of the system is needed in the proactive techniques. They highly rely on prediction and learning using artificial intelligence and probability theory. In this regard, the tasks executed remain uninterrupted until the system behaves according to the probability of the system's future state. Nevertheless, in case of any inaccurate prediction or any deviation in system

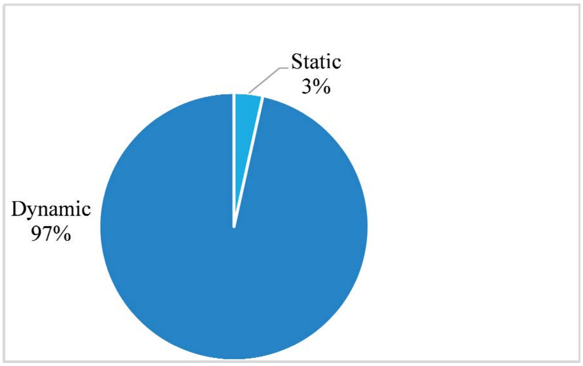

FIGURE 5. Percentage of adopted dynamic or static approach.

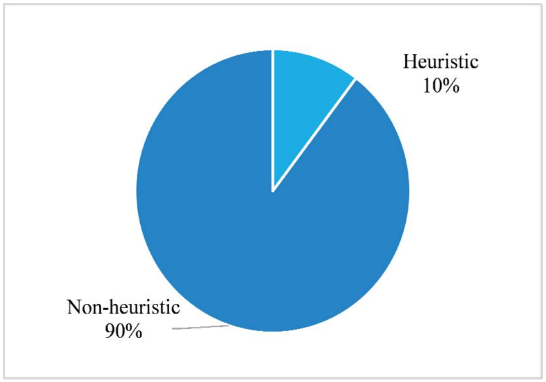

FIGURE 6. Percentage of adopted heuristic or meta-heuristic approach.

behavior, these methods become ineffective. Although reactive approaches, such as replication, job migration, and checkpoint, improve resource availability, these techniques waste a lot of resources and increase execution cost and overhead.

D. ADOPTED SIMULATION TOOLS AND TYPE OF DETECTED FAULTS

To answer RQ3, the authors highlighted the simulation tools used in the reviewed fault tolerance load balancing techniques. Figure 8 illustrates the percentage of adopted simulation tools. Moreover, in order to answer RQ4, the authors specified the type of detected faults in the reviewed papers, which are shown in Table 5. Considering Figure 9, 48% of the papers have attempted to address network faults.

E. THE SIGNIFICANCE OF CONSIDERED QUALITATIVE METRICS

The previous section reviewed the selected fault tolerance load balancing methods based on important metrics. As specified in Figure 10, the reviewed techniques have taken into account some metrics while neglecting the others.

VI. FUTURE TRENDS AND OPEN ISSUES

To address the RQ5 question, this section discusses some of the challenges and problems faced by previous works in the field of fault tolerance load balancing. The study findings indicate that there is no effective work for improving the

TABLE 5. Some metrics for examining load balancing approaches (N/A = Not available).

Category	Research	Dynamic or static	Heuristic or Non-heuristic	Basic approach (Reactive, proactive or hybrid) methods used	Simulation tool	Type of detected faults	Performance metrics improvement and advantages	Weakness
Distributed	[71]	Dynamic	Non-heuristic	Proactive (restart, checkpoint, and self-healing)	N/A	Network faults	Faster operation and lower overhead	It tolerates limited fault types
	[72]	Dynamic	Non-heuristic	Proactive (restart, checkpoint, and self-healing)	N/A	Network faults	Operates without base OS kernel changes	It may be slow at times
	[73]	Dynamic	Heuristic	Reactive (Replication)	Cloudsim	Physical faults	High resource utilization (110%)	It is not applicable if primary and backup fail simultaneously
	[74]	Dynamic	Heuristic	Hybrid (Stochastic Petri nets and checkpoint)	N/A	Physical faults	Low energy consumption (46%)	Limited experimentation
	[75]	Dynamic	Non-heuristic	Reactive (Task resubmission)	OpenNebua	Physical and network faults	Low execution time (27%) and cost (5%)	Limited failure aspects are considered
	[76]	Dynamic	Non-heuristic	Reactive (Replication)	N/A	Network faults	Availability and reliability	Without experimentation
	[77]	Dynamic	Non-heuristic	Reactive (Replication)	Matlab	Network faults	Energy consumption (80%) and success rate (6%)	High overhead
	[78]	Dynamic	Non-heuristic	Reactive (Replication)	N/A	Network faults	It is an adaptive framework that permits users' interaction for fault tolerance. Response time improvement (40%)	It is not suitable for inexperienced users
	[79]	Dynamic	Non-heuristic	Reactive (Job migration)	WorkflowSim	Physical faults	Low makespan (58%)	It is not suitable for hard deadline applications
	[80]	Dynamic	Non-heuristic	Proactive (Self-healing)	Jist/Swans	Network faults	Scalability, heterogeneity, and energy consumption (70%)	Complex implementation
	[81]	Dynamic	Non-heuristic	Reactive (Replication)	N/A	Network and process faults	Resource utilization	Limited experimentation
	[82]	Dynamic	Non-heuristic	Reactive (job migration)	Cloudsim	Physical faults	Throughput and number of migrations	Limited experimentation and limited failure aspects are considered
	[83]	Dynamic	Non-heuristic	Proactive (Self-healing)	CloudAnalyst	Network faults	Response time and energy consumption	Limited experimentation
	[84]	Dynamic	Heuristic	Proactive (Self-healing)	vSphere	Network faults	Throughput (16%)	It suffers from low scalability
	[85]	Dynamic	Non-heuristic	Proactive (Pre-emptive migration)	Storm platform	Network faults	High reliability (5%) and low response time (33%)	Limited fault applicability
	[86]	Dynamic	Non-heuristic	Reactive (Replication and job migration)	N/A	Physical faults	High resource utilization (30%)	It is not applicable if primary and backup fail simultaneously
	[51]	Dynamic	Non-heuristic	Reactive (Job migration)	Cloudsim	Network faults	Low makespan (60%, 38.9%, 31.5%, and 31.2% compared to MTCT, MAXMIN, ant colony optimization and genetic algorithm-based NSGA-II, respectively.)	It has not been evaluated in terms of resource utilization

TABLE 5. ***(Continued.)*** **Some metrics for examining load balancing approaches (N/A = Not available).**

	[87]	Dynamic	Heuristic	Reactive (Check point)	N/A	Network faults	Response time	Simulation results are not presented
	[88]	Dynamic	Non-heuristic	Reactive (Job migration)	Cloudsim	Network and physical faults	Overhead	Limited experimentation
	[89]	Dynamic	Non-heuristic	Hybrid (Monitoring, Migration, Prediction)	Java	Network and physical faults	Cost (50%), execution time (82%), and makespan (30%)	Limited applicability for real-time applications
	[90]	Dynamic	Non-heuristic	Proactive (Self-healing)	Cloudsim	Physical faults	Makespan (20%), response time (14%), and resource utilization (20%)	The scalability and overhead of the approach have not been evaluated
	[91]	Dynamic	Non-heuristic	Proactive (Self-healing)	Cloudsim	Physical faults	Makespan (24%), response time (33%), failure rate (20%), and energy consumption (50%)	It has not been evaluated in terms of resource utilization
	[92]	Dynamic	Non-heuristic	Reactive (Check point)	Cloudsim	Network and physical faults	Throughput (10%), makespan (10%), overhead (77%), and response time (6%)	Its low scalability remains a problem
	[93]	Dynamic	Non-heuristic	Reactive (Check point and replication)	Cloudsim	Physical faults	Execution time (5%) and SLA violation (20%)	The scalability and overhead of the approach have not been evaluated
Centralized	[94]	Dynamic	Non-heuristic	Reactive (Replication and job migration)	N/A	Physical faults	Scalability and overhead	Simulation results are not presented
	[95]	Dynamic	Non-heuristic	Reactive (Job migration)	N/A	Physical faults	Number of completed requests (18%), resource utilization (30%), and response time (7%)	Limited failure aspects are considered
	[96]	Static	Non-heuristic	Hybrid (Check point and fault handling)	Cloudsim	Network faults	Success rate	Tolerate limited fault types
	[97]	Dynamic	Non-heuristic	Hybrid (Check point, task resubmission, job migration)	OpenNebula	Network and physical faults	Makespan (16%) and response time (20%)	The scalability of the mechanism and involved computational overhead have not been checked
	[98]	Dynamic	Non-heuristic	Reactive (Job migration)	N/A	Network and physical faults	Reliability (5%)	It has not been evaluated in terms of resource utilization

entire load balancing parameters. For instance, some methods have taken into account response time, reliability, and throughput, while others have neglected these parameters. It seems that some parameters are mutually exclusive. For instance, relying on reliability for load balancing may cause an increase in overhead. Availability is another metric that has been ignored by most of the researchers in the reviewed techniques. Therefore, offering an effective technique considering all issues involved in load balancing is recommended for further studies.

In order to improve cloud performance, some important cases such as resource provisioning, SLA, and QoS should be considered. SLAs are designed based on QoS rules, and in case of any violation of the SLA, a service provider must pay the penalty. Automatic resource provisioning reduces the interaction between cloud service providers and cloud users. To maintain QoS and SLA, load balancing techniques are required for suitable use of provisioned resources. Furthermore, obtained results from previous sections show that it is not obvious how the researchers handle highly heterogeneous and distributed cloud platforms. Most of the techniques are not scalable and require manual intervention for proper configuration and operation. In this regard, it is recommended future works in this field are developed based on automation. Some interesting hints for further studies are listed in the following.

- Since demand for cloud services is increasing day by day and consumed energy by cloud data centers is also

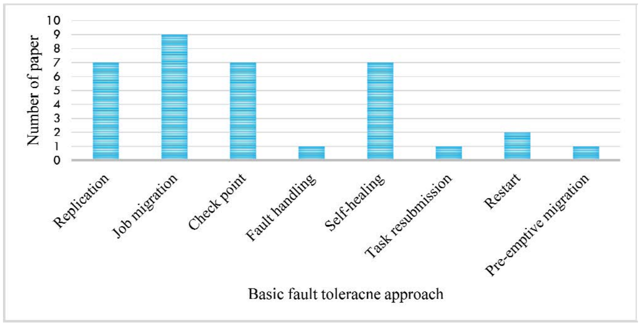

FIGURE 7. Percentage of adopted basic fault-tolerance approach.

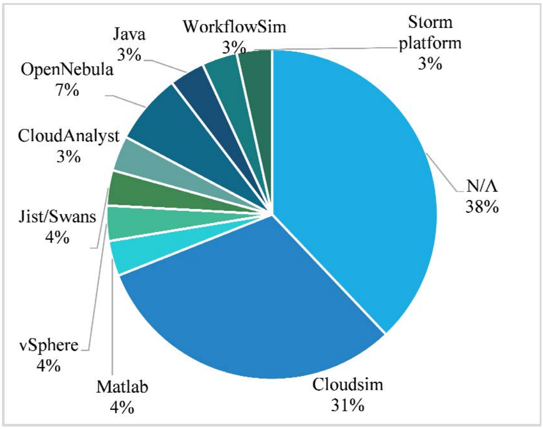

FIGURE 8. Percentage of adopted simulation tools.

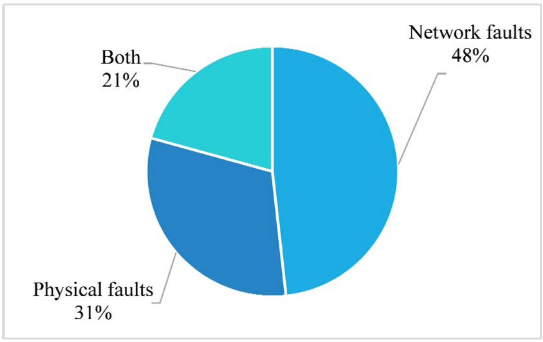

FIGURE 9. Percentage of type of detected faults.

growing, reducing energy consumption becomes a significant issue.

- Utilizing checkpoint-based approaches and component-level testing to improve the reliability of cloud systems is another interesting future trend.
- During transferring a workload among cloud providers, the difference among data and service policies, and data lock-in become a challenging problem. To resolve this kind of issue, some policies are required.
- Since the number of cloud service providers is increasing, cloud clients are facing an important challenge to discover proper service providers.
- Management of applications and resources in the dynamic and heterogeneous cloud environment is another challenging problem that requires further research.

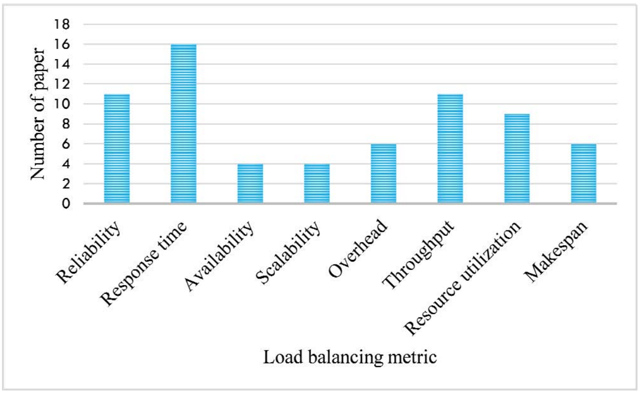

FIGURE 10. Considered load balancing metrics in the reviewed methods.

VII. CONCLUSION

Considering the importance of fault tolerance load balancing in cloud computing, this paper presented a detailed and systematic review of the existing methods in this. The methods were identified, classified, and analyzed using the well-known SLR method. The selected methods were classified into two groups and reviewed based on vital qualitative metrics, such as scalability, response time, availability, throughput, reliability, and overhead. In this regard, other criteria such as the adopted dynamic or static approach, adopted heuristic or meta-heuristic approach, adopted reactive or proactive fault tolerance approach, simulation tools, and type of detected faults were also considered. Moreover, a side-by-side comparison of discussed methods was offered, and challenges, research trends, and open issues to improve the existing works were also highlighted. The research results specify that in the static methods, since prior knowledge about the status of the system is needed and the current condition is ignored, these methods are not effective in terms of resource utilization and reliability. On the other hand, the dynamic methods are capable of managing the dynamic load conditions and improving resource utilization in an effective manner compared to the static ones. Although the dynamic methods effectively handle the users' requests with dynamic procedures and provide better performance compared to static methods, developing an algorithm for the dynamic cloud environment is a challenging matter.

REFERENCES

[1] B. Pourghebleh and N. J. Navimipour, "Data aggregation mechanisms in the Internet of Things: A systematic review of the literature and recommendations for future research," *J. Netw. Comput. Appl.*, vol. 97, pp. 23–34, Nov. 2017.

[2] E. Kristiani, C.-T. Yang, C.-Y. Huang, Y.-T. Wang, and P.-C. Ko, "The implementation of a cloud-edge computing architecture using OpenStack and Kubernetes for air quality monitoring application," *Mobile Netw. Appl.*, vol. 26, pp. 1070–1092, Jul. 2020.

[3] A. Souri, P. Asghari, and R. Rezaei, "Software as a service based CRM providers in the cloud computing: Challenges and technical issues," *J. Service Sci. Res.*, vol. 9, no. 2, pp. 219–237, Dec. 2017.

[4] M. Adhikari, S. Nandy, and T. Amgoth, "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud," *J. Netw. Comput. Appl.*, vol. 128, pp. 64–77, Feb. 2019.

[5] M. T. Sandikkaya, Y. Yaslan, and C. D. Özdemir, "DeMETER in clouds: Detection of malicious external thread execution in runtime with machine learning in PaaS clouds," *Cluster Comput.*, vol. 23, pp. 2565–2578, Dec. 2019.

[6] N. Jafari Navimipour, A. Habibizad Navin, A. M. Rahmani, and M. Hosseinzadeh, "Behavioral modeling and automated verification of a cloud-based framework to share the knowledge and skills of human resources," *Comput. Ind.*, vol. 68, pp. 65–77, Apr. 2015.

[7] M. Ashouraie and N. J. Navimipour, "Priority-based task scheduling on heterogeneous resources in the expert cloud," *Kybernetes*, vol. 44, no. 10, pp. 1455–1471, Nov. 2015.

[8] E. Iranpour and S. Sharifian, "A distributed load balancing and admission control algorithm based on fuzzy type-2 and game theory for large-scale SaaS cloud architectures," *Future Gener. Comput. Syst.*, vol. 86, pp. 81–98, Sep. 2018.

[9] O. Sohaib, M. Naderpour, W. Hussain, and L. Martinez, "Cloud computing model selection for e-commerce enterprises using a new 2-tuple fuzzy linguistic decision-making method," *Comput. Ind. Eng.*, vol. 132, pp. 47–58, Jun. 2019.

[10] A. Iqbal and R. Colomo-Palacios, "Key opportunities and challenges of data migration in cloud: Results from a multivocal literature review," *Proc. Comput. Sci.*, vol. 164, pp. 48–55, Jan. 2019.

[11] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," *Cluster Comput.*, vol. 24, pp. 2673–2696, May 2021.

[12] P. Azad, N. J. Navimipour, and M. Hosseinzadeh, "A fuzzy-based method for task scheduling in the cloud environments using inverted ant colony optimisation algorithm," *Int. J. Bio-Inspired Comput.*, vol. 14, no. 2, pp. 125–137, 2019.

[13] S. K. Zaman, T. Maqsood, M. Ali, K. Bilal, S. A. Madani, and A. R. Khan, "A load balanced task scheduling heuristic for large-scale computing systems," *Comput. Syst. Sci. Eng.*, vol. 34, no. 2, pp. 79–90, 2019.

[14] B. Pourghebleh and V. Hayyolalam, "A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things," *Cluster Comput.*, vol. 23, pp. 641–661, Jun. 2019.

[15] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 32, no. 2, pp. 149–158, 2020.

[16] P. Tyagi and A. Kishor, "Load balancing in cloud computing," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 10, pp. 22–27, 2018.

[17] M. Amoon, N. El-Bahnasawy, S. Sadi, and M. Wagdi, "On the design of reactive approach with flexible checkpoint interval to tolerate faults in cloud computing systems," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 11, pp. 4567–4577, Nov. 2019.

[18] A. Marahatta, Y. Wang, F. Zhang, A. K. Sangaiah, S. K. S. Tyagi, and Z. Liu, "Energy-aware fault-tolerant dynamic task scheduling scheme for virtualized cloud data centers," *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 1063–1077, Jun. 2019.

[19] M. Nazari Cheraghlou, A. Khademzadeh, and M. Haghparast, "New fuzzy-based fault tolerance evaluation framework for cloud computing," *J. Netw. Syst. Manage.*, vol. 27, no. 4, pp. 930–948, Oct. 2019.

[20] S. B. Shaw and A. K. Singh, "A survey on scheduling and load balancing techniques in cloud computing environment," in *Proc. Int. Conf. Comput. Commun. Technol. (ICCCT)*, Sep. 2014, pp. 87–95.

[21] A. A. Salah Farrag, S. A. Mahmoud, and E. S. M. El-Horbaty, "Intelligent cloud algorithms for load balancing problems: A survey," in *Proc. IEEE 7th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2015, pp. 210–216.

[22] S. Aslam and M. A. Shah, "Load balancing algorithms in cloud computing: A survey of modern techniques," in *Proc. Nat. Softw. Eng. Conf. (NSEC)*, Dec. 2015, pp. 30–35.

[23] M. Mesbahi and A. M. Rahmani, "Load balancing in cloud computing: A state of the art survey," *Int. J. Mod. Educ. Comput. Sci.*, vol. 8, no. 3, pp. 64–78, Mar. 2016.

[24] A. S. Milani and N. J. Navimipour, "Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends," *J. Netw. Comput. Appl.*, vol. 71, pp. 86–98, Aug. 2016.

[25] P. K. Tiwari and S. Joshi, "A review on load balancing of virtual machine resources in cloud computing," in *Proc. 1st Int. Conf. Inf. Commun. Technol. Intell. Syst.* (Smart Innovation, Systems and Technologies), vol. 51. Ahmedabad, India: Springer, 2016, pp. 369–378.

[26] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Load-balancing algorithms in cloud computing: A survey," *J. Netw. Comput. Appl.*, vol. 88, pp. 50–71, Jun. 2017.

[27] R. Eswaraprasad and L. Raja, "A review of virtual machine (VM) resource scheduling algorithms in cloud computing environment," *J. Statist. Manage. Syst.*, vol. 20, no. 4, pp. 703–711, Jul. 2017.

[28] M. O. Ahmad and R. Z. Khan, "Load balancing tools and techniques in cloud computing: A systematic review," in *Advances in Computer and Computational Sciences* (Advances in Intelligent Systems and Computing), vol. 554. Singapore: Springer, 2018, pp. 181–195.

[29] A. Hota, S. Mohapatra, and S. Mohanty, "Survey of different load balancing approach-based algorithms in cloud computing: A comprehensive review," in *Computational Intelligence in Data Mining* (Advances in Intelligent Systems and Computing), vol. 711. Singapore: Springer, 2019, pp. 99–110.

[30] S. Kaur, P. Bagga, R. Hans, and H. Kaur, "Quality of service (QoS) aware workflow scheduling (WFS) in cloud computing: A systematic review," *Arabian J. Sci. Eng.*, vol. 44, no. 4, pp. 2867–2897, Apr. 2019.

[31] K. R. Jothi, S. Anto, M. Kohar, M. Chadha, and P. Madhavan, "Smart load balancing algorithms in cloud computing—A review," in *Role of Edge Analytics in Sustainable Smart City Development: Challenges and Solutions*. Hoboken, NJ, USA: Wiley, 2020, pp. 189–218.

[32] A. Jyoti, M. Shrimali, S. Tiwari, and H. P. Singh, "Cloud computing using load balancing and service broker policy for IT service: A taxonomy and survey," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 11, pp. 4785–4814, 2020.

[33] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends," *Swarm Evol. Comput.*, vol. 62, Apr. 2021, Art. no. 100841.

[34] B. Pourghebleh, V. Hayyolalam, and A. Aghaei Anvigh, "Service discovery in the Internet of Things: Review of current trends and research challenges," *Wireless Netw.*, vol. 26, no. 7, pp. 5371–5391, Oct. 2020.

[35] M. Mohammadi, T. A. Rashid, S. H. T. Karim, A. H. M. Aldalwie, Q. T. Tho, M. Bidaki, A. M. Rahmani, and M. Hosseinzadeh, "A comprehensive survey and taxonomy of the SVM-based intrusion detection systems," *J. Netw. Comput. Appl.*, vol. 178, Mar. 2021, Art. no. 102983.

[36] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Highly reliable architecture using the 80/20 rule in cloud computing datacenters," *Future Gener. Comput. Syst.*, vol. 77, pp. 77–86, Dec. 2017.

[37] H. Zhang, G. Chen, and X. Li, "Resource management in cloud computing with optimal pricing policies," *Comput. Syst. Sci. Eng.*, vol. 34, no. 4, pp. 249–254, 2019.

[38] A. Tchernykh, U. Schwiegelsohn, E.-G. Talbi, and M. Babenko, "Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability," *J. Comput. Sci.*, vol. 36, Sep. 2019, Art. no. 100581.

[39] E. Azhir, N. J. Navimipour, M. Hosseinzadeh, A. Sharifi, and A. Darwesh, "Query optimization mechanisms in the cloud environments: A systematic study," *Int. J. Commun. Syst.*, vol. 32, no. 8, p. e3940, 2019.

[40] F. Firouzi and B. Farahani, "Architecting IoT cloud," in *Intelligent Internet of Things*. Cham, Switzerland: Springer, 2020, pp. 173–241.

[41] G. S. Prakash, "A literature review of QoS with load balancing in cloud computing environment," in *Big Data Analytics* (Advances in Intelligent Systems and Computing), vol. 654. Singapore: Springer, 2018, pp. 667–675.

[42] B. Liu, Y. Chen, A. Hadiks, E. Blasch, A. Aved, D. Shen, and G. Chen, "Information fusion in a cloud computing era: A systems-level perspective," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 29, no. 10, pp. 16–24, Oct. 2014.

[43] A. Jyoti and M. Shrimali, "Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing," *Cluster Comput.*, vol. 23, no. 1, pp. 377–395, Mar. 2020.

[44] S. K. Mishra, D. Puthal, B. Sahoo, S. Sharma, Z. Xue, and A. Y. Zomaya, "Energy-efficient deployment of edge dataenters for mobile clouds in sustainable IoT," *IEEE Access*, vol. 6, pp. 56587–56597, 2018.

[45] V. D. Chakravarthy and B. Amutha, "A novel software-defined networking approach for load balancing in data center networks," *Int. J. Commun. Syst.*, vol. 35, no. 2, p. e4213, 2019.

[46] Z. Wu, C. Li, J. Cao, and Y. Ge, "On scalability of association-rule-based recommendation: A unified distributed-computing framework," *ACM Trans. Web*, vol. 14, no. 3, pp. 1–21, 2020.

[47] B. Alankar, G. Sharma, H. Kaur, R. Valverde, and V. Chang, "Experimental setup for investigating the efficient load balancing algorithms on virtual cloud," *Sensors*, vol. 20, no. 24, p. 7342, Dec. 2020.

[48] A. A. Abdelltif, E. Ahmed, A. T. Fong, A. Gani, and M. Imran, ''SDN-based load balancing service for cloud servers,'' *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 106–111, Aug. 2018.

[49] R. Kumari and V. K. Jha, ''Performance analysis of load balancing algorithms in Amazon cloud,'' in *Proc. 5th Int. Conf. Microelectron., Comput. Commun. Syst.*, 2021, pp. 401–412.

[50] N. Hota and B. K. Pattanayak, ''Cloud computing load balancing using Amazon web service technology,'' in *Progress in Advanced Computing and Intelligent Engineering*. Singapore: Springer, 2021, pp. 661–669.

[51] S. M. Abdulhamid, M. S. Abd Latiff, S. H. H. Madni, and M. Abdullahi, ''Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm,'' *Neural Comput. Appl.*, vol. 29, no. 1, pp. 279–293, Jan. 2018.

[52] M. Hasan and M. S. Goraya, ''Fault tolerance in cloud computing environment: A systematic survey,'' *Comput. Ind.*, vol. 99, pp. 156–172, Aug. 2018.

[53] U. Dwivedi and H. Dev, ''A review on fault tolerance techniques and algorithms in green cloud computing,'' *J. Comput. Theor. Nanosci.*, vol. 15, no. 9, pp. 2689–2700, Sep. 2018.

[54] A. Tikotekar, G. Vallée, T. Naughton, S. L. Scott, and C. Leangsuksun, ''Evaluation of fault-tolerant policies using simulation,'' in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2007, pp. 303–311.

[55] F. Salfner, M. Lenk, and M. Malek, ''A survey of online failure prediction methods,'' *ACM Comput. Surv.*, vol. 42, no. 3, pp. 1–42, Mar. 2010.

[56] S. Prathiba and S. Sowvarnica, ''Survey of failures and fault tolerance in cloud,'' in *Proc. 2nd Int. Conf. Comput. Commun. Technol. (ICCCT)*, Feb. 2017, pp. 169–172.

[57] M. A. Mukwevho and T. Celik, ''Toward a smart cloud: A review of fault-tolerance methods in cloud systems,'' *IEEE Trans. Services Comput.*, vol. 14, no. 2, pp. 589–605, Mar. 2021.

[58] R. Taskeen Zaidi, ''Modeling for fault tolerance in cloud computing environment,'' *J. Comput. Sci. Appl.*, vol. 4, no. 1, pp. 9–13, 2016.

[59] P. Kumari and P. Kaur, ''A survey of fault tolerance in cloud computing,'' *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 33, no. 10, pp. 1159–1176, 2021.

[60] A. Semmoud, M. Hakem, B. Benmammar, and J. Charr, ''Load balancing in cloud computing environments based on adaptive starvation threshold,'' *Concurrency Comput., Pract. Exper.*, vol. 32, no. 11, pp. 1–14, Jun. 2020.

[61] B. Cao, S. Fan, J. Zhao, S. Tian, Z. Zheng, Y. Yan, and P. Yang, ''Large-scale many-objective deployment optimization of edge servers,'' *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3841–3849, Jun. 2021.

[62] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, ''Reliability and high availability in cloud computing environments: A reference roadmap,'' *Hum.-Centric Comput. Inf. Sci.*, vol. 8, no. 1, p. 20, 2018.

[63] Z. Lv, D. Chen, R. Lou, and H. Song, ''Industrial security solution for virtual reality,'' *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6273–6281, Apr. 2021.

[64] M. Rahimi, N. J. Navimipour, M. Hosseinzadeh, M. H. Moattar, and A. Darwesh, ''Toward the efficient service selection approaches in cloud computing,'' *Kybernetes*, Jun. 2021, doi: 10.1108/K-02-2021-0129.

[65] V. Hayyolalam, B. Pourghebleh, and A. A. P. Kazem, ''Trust management of services (TMoS): Investigating the current mechanisms,'' *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 10, p. e4063, Oct. 2020.

[66] V. Mohammadian, N. Jafari Navimipour, M. Hosseinzadeh, and A. Darwesh, ''Comprehensive and systematic study on the fault tolerance architectures in cloud computing,'' *J. Circuits, Syst. Comput.*, vol. 29, no. 15, 2020, Art. no. 2050240.

[67] A. Zaouch and F. Benabbou, ''Load balancing for improved quality of service in the cloud,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 6, no. 7, pp. 184–189, 2015.

[68] S. Akhbarifar, H. H. S. Javadi, A. M. Rahmani, and M. Hosseinzadeh, ''A secure remote health monitoring model for early disease diagnosis in cloud-based IoT environment,'' *Pers. Ubiquitous Comput.*, pp. 1–17, Nov. 2020.

[69] V. Hayyolalam, B. Pourghebleh, M. R. Chehrehzad, and A. A. P. Kazem, ''Single-objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends,'' *Concurrency Comput., Pract. Exper.*, p. e6698, Nov. 2021, doi: 10.1002/cpe.6698.

[70] A. Mehbodniya, R. Neware, S. Vyas, M. R. Kumar, P. Ngulube, and S. Ray, ''Blockchain and IPFS integrated framework in bilevel fog-cloud network for security and privacy of IoMT devices,'' *Comput. Math. Methods Med.*, vol. 2021, Dec. 2021, Art. no. 7727685.

[71] G. Chen, H. Jin, D. Zou, B. B. Zhou, W. Qiang, and G. Hu, ''SHelp: Automatic self-healing for multiple application instances in a virtual machine environment,'' in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2010, pp. 97–106.

[72] S. Sidiroglou, O. Laadan, C. Perez, N. Viennot, J. Nieh, and A. D. Keromytis, ''ASSURE: Automatic software self-healing using rescue points,'' *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 1, pp. 37–48, 2009.

[73] J. Wang, W. Bao, X. Zhu, L. T. Yang, and Y. Xiang, ''FESTAL: Fault-tolerant elastic scheduling algorithm for real-time tasks in virtualized clouds,'' *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2545–2558, Sep. 2015.

[74] S. Khan and N. Sharama, ''Effective scheduling algorithm for load balancing (SALB) using ant colony optimization in cloud computing,'' *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 4, no. 2, pp. 966–973, 2014.

[75] M. R. Sumalatha, C. Selvakumar, T. Priya, R. T. Azariah, and P. M. Manohar, ''CLBC–cost effective load balanced resource allocation for partitioned cloud system,'' in *Proc. Int. Conf. Recent Trends Inf. Technol.*, Apr. 2014, pp. 1–5.

[76] G. Gayathri and N. Prabakaran, ''Ensuring reliability and high availability in cloud by employing a fault tolerance enabled load balancing algorithm,'' *Int. J. Comput. Sci. Trends Technol.*, vol. 3, no. 3, pp. 54–58, 2015.

[77] A. Moghtadaeipour and R. Tavoli, ''A new approach to improve load balancing for increasing fault tolerance and decreasing energy consumption in cloud computing,'' in *Proc. 2nd Int. Conf. Knowl.-Based Eng. Innov. (KBEI)*, Nov. 2015, pp. 982–987.

[78] X. Chen and J.-H. Jiang, ''A method of virtual machine placement for fault-tolerant cloud applications,'' *Intell. Autom. Soft Comput.*, vol. 22, no. 4, pp. 587–597, Oct. 2016.

[79] G. Yao, Y. Ding, L. Ren, K. Hao, and L. Chen, ''An immune system-inspired rescheduling algorithm for workflow in cloud systems,'' *Knowl.-Based Syst.*, vol. 99, pp. 39–50, May 2016.

[80] C.-A. Chen, R. Stoleru, and G. G. Xie, ''Energy-efficient load-balanced heterogeneous mobile cloud,'' in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2017, pp. 1–9.

[81] N. Jayapandian, R. Menagadevi, S. Abinaya, O. S. Sampoorani, and A. M. J. Z. Rahman, ''Dynamic load balancing cluster and fault-tolerant in cloud environment,'' *Comput. Eng.*, vol. 103, pp. 44520–44523, May 2017.

[82] G. Gayathri and R. Latha, ''Implementing a fault tolerance enabled load balancing algorithm in the cloud computing environment,'' *Int. J. Eng. Dev. Res.*, vol. 5, no. 1, pp. 249–256, 2017.

[83] A. Ragmani, A. E. Omri, N. Abghour, K. Moussaid, and M. Rida, ''An efficient load balancing strategy based on mapreduce for public cloud,'' in *Proc. 2nd Int. Conf. Internet Things, Data Cloud Comput.*, Mar. 2017, pp. 1–10.

[84] B. Balusamy, K. Karthikeyan, and A. K. Sangaiah, ''Ant colony-based load balancing and fault recovery for cloud computing environment,'' *Int. J. Adv. Intell. Paradigms*, vol. 9, nos. 2–3, pp. 204–219, 2017.

[85] D. Sun, G. Zhang, C. Wu, K. Li, and W. Zheng, ''Building a fault tolerant framework with deadline guarantee in big data stream computing environments,'' *J. Comput. Syst. Sci.*, vol. 89, pp. 4–23, Nov. 2017.

[86] Y. Ding, G. Yao, and K. Hao, ''Fault-tolerant elastic scheduling algorithm for workflow in cloud systems,'' *Inf. Sci.*, vol. 393, pp. 47–65, Jul. 2017.

[87] A. Kaur and G. Kaur, ''Resource scheduling based on load balancing with fault tolerance in cloud computing,'' *Int. Adv. Res. J. Sci. Eng. Technol.*, vol. 5, no. 5, pp. 59–66, 2018.

[88] S. Sreelekshmi and K. R. R. Babu, *Fault Tolerant Multiple Synchronized Parallel Load Balancing in Cloud*, vol. 734. Delhi, India: Springer, 2018.

[89] T. Tamilvizhi and B. Parvathavarthini, ''A novel method for adaptive fault tolerance during load balancing in cloud computing,'' *Cluster Comput.*, vol. 22, no. S5, pp. 10425–10438, Sep. 2019.

[90] F. Ebadifard, S. M. Babamir, and S. Barani, ''A dynamic task scheduling algorithm improved by load balancing in cloud computing,'' in *Proc. 6th Int. Conf. Web Res. (ICWR)*, Apr. 2020, pp. 177–183.

[91] H. Jahanpour, H. Barati, and A. Mehranzadeh, ''An energy efficient fault tolerance technique based on load balancing algorithm for high-performance computing in cloud computing,'' *J. Elect. Comput. Eng. Innov.*, vol. 8, no. 2, pp. 169–182, 2020.

[92] A. Shukla, S. Kumar, and H. Singh, ''Fault tolerance based load balancing approach for web resources in cloud environment,'' *Int. Arab J. Inf. Technol.*, vol. 17, no. 2, pp. 225–232, 2020.

[93] S. Setaouti, D. Djamel Amar Bensaber, R. Adjoudj, and M. Rebbah, ''Fault tolerance directed by service level agreement in cloud computing environments,'' *Int. J. Comput. Digit. Syst.*, Aug. 2021.

[94] A. Roy, ''Dynamic load balancing: Improve efficiency in cloud computing,'' *Int. J. Emerg. Res. Manag. Technol.*, vol. 9359, no. 4, pp. 78–82, 2013.

[95] P. Gupta and S. P. Ghrera, ''Fault and load aware load balancing in cloud storage,'' *Int. J. Appl. Eng. Res.*, vol. 10, no. 69, pp. 280–285, 2015.

[96] E. S. Sharma, E. I. Ahmad, E. S. Mirdha, and M. Tech, "Improving cloud performance through performance based load balancing approach," *Int. Res. J. Eng. Technol.*, vol. 4, no. 6, pp. 523–528, 2017.
[97] A. Rezaeipanah, M. Mojarad, and A. Fakhari, "Providing a new approach to increase fault tolerance in cloud computing using fuzzy logic," *Int. J. Comput. Appl.*, pp. 1–9, Jan. 2020.
[98] S. M. A. Attallah, M. B. Fayek, S. M. Nassar, and E. E. Hemayed, "Proactive load balancing fault tolerance algorithm in cloud computing," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 10, May 2021, e6172.
[99] A. Ray, "Dynamic load balancing: Improve efficiency in cloud computing," *Int. J. Emerg. Res. Manag. Technol.*, vol. 9359, no. 24, p. 2278, 2013.

VAHID MOHAMMADIAN received the B.S. degree in software computer engineering from Meybod Branch, Islamic Azad University, Meybod, Iran, in 2010, and the M.Sc. degree in computer engineering, software engineering, from International Qeshm Branch, Islamic Azad University, Qeshm, Iran, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Computer Engineering. His research interests include cloud systems, social networks, data mining, big data analytics, and recommender systems.

NIMA JAFARI NAVIMIPOUR received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering from Islamic Azad University, Iran, in 2008, 2009, and 2014, respectively. He is currently an Associate Professor. He has been giving invited tutorials/talks in IEEE conferences and has been invited to give lectures in different universities. His research interests include cloud and distributed computing, the Internet of Things (IoT), software-defined networking (SDN), information systems, computational intelligence, evolutionary computing, and quantum computing. He has published many papers in various journals and conference proceedings as well as supervising/co-supervising several Ph.D. and master's students in these research areas. He is a Senior Member of the IEEE Communications Society and the IEEE Young Professionals. He is a Technical Committee Member, a Guest Editor, and an Associate Editor of some high-ranked journals, such as *IET Quantum Communication*, *Optik*, *Journal of Management and Organization*, *Computer Communication*, *Cluster Computing*, and *Kybernetes*. Furthermore, he is a chair member of many prestigious conferences and a reviewer of several high-ranked journals. He also won the Publons Top Peer Review Awards in 2018 and 2019. He has been featured among the World's Top 2% Scientists List, according to a conducted study by U.S.-based Stanford University in 2020.

MEHDI HOSSEINZADEH received the B.S. degree in computer hardware engineering from Islamic Azad University, Dezful Branch, Iran, in 2003, and the M.Sc. and Ph.D. degrees in computer system architecture from the Science and Research Branch, Islamic Azad University, Tehran, Iran, in 2005 and 2008, respectively. He is the author/coauthor of more than 150 publications in technical journals and conferences. His research interests include SDN, information technology, data mining, big data analytics, e-commerce, e-marketing, and social networks.

ASO DARWESH received the B.S. degree in mathematics from the University of Sulaimani, Iraq, in 2001, the M.S. degree in computer science from the University of Rene Descartes, France, in 2007, and the Ph.D. degree in computer science from the University of Pierre and Mari Curie, France, in 2010. He is currently an Associate Professor with the Information Technology Department, University of Human Development, Sulaymaniyah, Iraq. His research interests include serious games, adaptive learning cognitive diagnosis in e-learning, learning systems, computer networks, networking security, and data mining.