Fault-tolerant load balancing in cloud computing: A systematic literature review

Vahid Mohammadian1,*, Aso Darwesh2

1Department of Computer Engineering, Qeshm Branch Islamic Azad University, Qeshm, Iran
2Information Technology Departments, University of Human Development Sulaimaniyah, Iraq

Corresponding Author: Vahid Mohammadian
Email address: henavidd@gmail.com

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 detect 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, response time, reliability, availability, 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. 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 [1]. Computing resources such as servers, storage, and computer networks are provided in four main forms, such as Software as a Service (SaaS) [2], Infrastructure as a Service (IaaS) [3], Platform as a Service (PaaS) [4], and Expert as a Service (EaaS) [5], [6]. In SaaS, the applications are deployed in the cloud and the cloud clients are allowed to access them through the Internet [7]. IaaS provides virtual private network hardware and software, hardware and software firewalls, storage, processing power, and network infrastructure [8]. PaaS permits clients to use predefined tools prepared by cloud providers to deploy web applications and other programming software [9]. EaaS provides expert knowledge and human resources as a service [5].

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 [10]. During tasks scheduling on Virtual Machines (VMs), there may happen a situation that some of the VMs are over-utilized while others remain under-utilized. 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 [11], [12]. In fact, this technique ensures that each VM performs an approximately equal number of tasks. It improves QoS metrics such as throughput, response time, reliability, and resource utilization and prevents system bottlenecks, which may happen due to load imbalance [13]. 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 [14], [15]. Fault tolerance is considered a vital and key feature of cloud computing. It refers to offering cloud services even in the presence of faults so that the system can discover the type and the location of the fault and attempt to tolerate it [16].

B. RELATED WORKS AND OUR MOTIVATION

In the recent decade, many researchers have studied the load balancing approaches 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 [17], 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 [18]. They have suggested the implementation of ALO as an efficient algorithm based on the cloud environment. 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 [19] 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 [20]. 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. However, future topics and open issues have not been discussed.

Furthermore, the researchers of [21] 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 [22] 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 [23], 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 throughput, scalability, makespan, response time, energy, and resource utilization. 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.

A review of existing tools and methods for load balancing in cloud computing has been presented in [24]. The reviewed methods have been assessed based on some metrics and parameters such as resource utilization, throughput, scalability, fault tolerance, reaction time, overhead, and performance. However, newly published papers have been neglected. Also, the proposed survey paper in [25] 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 existing research challenges related to loading balancing have been checked in [26], 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 [27] has specified, described, compared, and assessed the published works between 2015 and 2018. It has classified and analyzed methods based on the important metrics in cloud computing techniques.

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 organized into seven sections. The next section describes the adopted review method. 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 follows a Systematic Literature Review (SLR) method to carry out the research. Generally, the SLR aims to provide a detailed outline of available works about a specific subject [28], [29]. 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. This article presents a comprehensive review of fault tolerance load balancing mechanisms in cloud computing using the SLR method. As specified in Figure 1, the adopted methodology consists of 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 remarkable hints for further studies are presented.

A. REVIEW PLANNING

This section clarifies some Research Questions (RQs) that are expected to be found while reviewing the fault tolerance load balancing methods in cloud computing. 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 challenging problems of fault tolerance load balancing methods?

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

RQ3: What are the popular simulation tools to implement fault tolerance load balancing methods?

RQ4: What kinds of faults have been addressed in current works in the field of fault tolerance load balancing?

RQ5: What are the future trends and open issues about fault tolerance load balancing in cloud computing?

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 Xplore1, Springer link2, Science Direct3, and Google Scholar4 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.

FIGURE 1. Adopted research methodology

| 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 |

1 ieeexplore.ieee.org
2 Link.springer.com
3 Sciencedirect.com
4 Scholar.google.com
Round 3: The authors read the remaining studies carefully from Round 2 and selected 23 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.

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

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 [8]. 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. 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 [30]. 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 [31].
- **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 [32].
- **Broad network access:** All the cloud services are accessible through the Internet, and are supporting various client platforms [32].
Fault tolerance aims to obtain dependability and robustness in a cloud system steady without human intervention [31]. This problem has been solved by different load balancing algorithms in the recent decade. Another principal challenge in cloud computing is fault tolerance. It is the capability 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 [35]. 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 [16]. Some of the important approaches based on this policy are described in the following.

- **Resource pooling**: Computing resources, such as memory, storage, network bandwidth, and processing, can be combined to become a multi-tenant model [33].
- **On-demand self-service**: The cloud clients are capable of utilizing computing capabilities independently and without human intervention [31].

### 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 [34]. This problem has been solved by different load balancing algorithms in the recent decade. Another principal challenge in cloud computing is fault tolerance. It is the capability 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 [35]. 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.

| Research | Year | Journal or conference name |
|----------|------|-----------------------------|
| [52]     | 2014 | Advances in engineering and technology |
| [53]     | 2014 | Recent trends in information technology |
| [54]     | 2015 | Computer science trends and technology |
| [55]     | 2016 | Knowledge-based engineering and innovation |
| [56]     | 2017 | Computer communications and networks |
| [57]     | 2017 | Computer engineering |
| [58]     | 2017 | Engineering development and research |
| [59]     | 2017 | Internet of things, data and cloud computing |
| [60]     | 2017 | Advanced Intelligence Paradigms |
| [61]     | 2018 | Advanced research journal in science, engineering and technology |
| [62]     | 2018 | Advances in intelligent systems and computing |
| [63]     | 2019 | Cluster computing |
| [64]     | 2020 | Web research |
| [65]     | 2020 | Electrical and computer engineering innovations |
| [66]     | 2020 | Arab journal of information technology |
| [67]     | 2021 | Computing and digital system |

| Research | Year | Journal or conference name |
|----------|------|-----------------------------|
| [73]     | 2013 | Emerging research in management and technology |
| [69]     | 2015 | Applied engineering research |
| [70]     | 2017 | Research journal of engineering and technology |
| [71]     | 2020 | Computers and applications |
| [72]     | 2021 | Concurrency and computation: practice and experience |

Proactive fault tolerance: Prediction forms the core of proactive fault tolerance algorithms [38]. Indeed, proactive fault tolerance predicts the faults proactively and swaps the suspected components by valid components [39].
Software rejuvenation: This method is specially used and planned for a periodic reboot of the system [40].

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 [41].

Preemptive migration: In this method, an application is continually observed and examined [42]. 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 [43].

Physical faults: These faults refer to faults in storage, memory, and CPUs [43].

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 [44], 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 (16 articles). In this respect, important requirements and metrics have been considered, which are defined below:

Availability: It is defined as the probability that a system functional correctly during a specific time in the stated situation [45].

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 [23].

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 [46].

Response time: It is defined as the time taken to respond/reply to a specific algorithm [47].

Overhead: This parameter refers to the amount of overhead involved while implementing a load balancing algorithm [48].

Throughput: It is defined as calculating the number of processes or tasks completed within a stipulated time period [49].

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 [50].

Makespan: It is defined as the time taken to process a set of tasks for its complete execution [51].

A. REVIEW OF DISTRIBUTED APPROACHES

A load-balancing method using the ACO algorithm has been offered in [52]. 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 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 [53] 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 [54] 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 [55]. 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.

An energy-efficient and load-balanced distributed storage and processing system has been proposed by researchers in [56]. 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. Each node in the network is permitted to submit a task to process a subset of stored files. Processing tasks include multiple independent tasks where each task corresponds to processing a single file on a selected processor node. 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 [57]. 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 [58] 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 [59] 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 [60]. 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.

Using the ACO algorithm, a novel approach to load balancing has been offered in [61] 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 [62] 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 [63]. 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 [64]. 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 [65] 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 [66]. 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, throughpup, makespan, and response time, but its low scalability remains a problem.

Finally, the researchers of [67] 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 [68] 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 [69], 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 [70] 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 [71], 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 [72]. 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 a 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. Moreover, a side-by-side comparison of these methods is shown in Table 6. The obtained results of the research are outlined and presented in the rest of this section.

![Table 5: Some metrics for examining load balancing approaches (N/A = Not available)](image)

| 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 | Weakness |
|----------|----------|-------------------|---------------------------|------------------------------------------------------------|----------------|------------------------|---------------------------------|---------|
|          | [52]     | Dynamic           | Heuristic                 | Hybrid (Stochastic Petri nets and checkpoint)              | N/A            | Physical faults        | Reliability                      | Simulation results are not presented |
|          | [53]     | Dynamic           | Non-heuristic             | Reactive (Replication)                                    | OpenNebua      | Physical and network faults | Execution and cost               | Limited failure aspects are considered |
|          | [54]     | Dynamic           | Non-heuristic             | Reactive (Replication)                                    | N/A            | Network faults          | Availability and reliability      | Without experimentation |
|          | [55]     | Dynamic           | Non-heuristic             | Reactive (Replication)                                    | Matlab         | Network faults          | Energy consumption and success rate | High overhead |
|          | [56]     | Dynamic           | Non-heuristic             | Proactive (Self-healing)                                  | Jist/Swans     | Network faults          | Scalability, heterogeneity, and energy consumption | Complex implementation |
|          | [57]     | Dynamic           | Non-heuristic             | Reactive (Replication)                                    | N/A            | Network and process faults | Resource utilization              | Limited experimentation |
|          | [58]     | Dynamic           | Non-heuristic             | Reactive (job migration)                                  | Cloudsim       | Physical faults         | Throughput and number of migrations | Limited experimentation and limited failure aspects are considered |
|          | [59]     | Dynamic           | Non-heuristic             | Proactive (Self-healing)                                  | CloudAnalyst   | Network                 | Response time                    | Limited |

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
| Category | Research | Load balancing qualitative metrics | Faults and energy consumption | Experimentation |
|----------|----------|---------------------------------|------------------------------|-----------------|
| [60] Dynamic Heuristic | Proactive (Self-healing) vSphere | Network faults Throughput | Suffers from low scalability |
| [61] Dynamic Heuristic | Reactive (Check point) N/A | Network faults Response time | Simulation results are not presented |
| [62] Dynamic Non-heuristic | Reactive (Job migration) Cloudsim | Network and physical faults Overhead | Limited experimentation |
| [63] Dynamic Non-heuristic | Hybrid (Monitoring, Migration, Prediction) Java | Cost, execution time, and scalability Makespan and response time | Limited applicability for real-time applications |
| [64] Dynamic Non-heuristic | Proactive (Self-healing) Cloudsim | Physical faults Makespan, response time, failure rate, and energy consumption | The scalability and overhead of the approach have not been evaluated |
| [65] Dynamic Non-heuristic | Proactive (Self-healing) Cloudsim | Physical faults Makespan, response time, failure rate, and energy consumption | It has not been evaluated in terms of resource utilization |
| [66] Dynamic Non-heuristic | Reactive (Check point) Cloudsim | Network and physical faults Throughput | Its low scalability remains a problem |
| [67] Dynamic Non-heuristic | Reactive (Check point and replication) Cloudsim | Physical faults Makespan, response time, failure rate, and energy consumption | The scalability and overhead of the approach have not been evaluated |
| [68] Dynamic Non-heuristic | Reactive (Replication and job migration) N/A | Physical faults Scalability and overhead | Simulation results are not presented |
| [69] Dynamic Non-heuristic | Reactive (Job migration) N/A | Physical faults Number of completed requests, resource utilization, and response time | Limited failure aspects are considered |
| [70] Static Non-heuristic | Hybrid (Check point and fault handling) Cloudsim | Network faults Success rate | Tolerate limited fault types |
| [71] Dynamic Non-heuristic | Hybrid (Check point, task resubmission, job migration) OpenNebula | Network and physical faults Makespan, response time, and workload | The scalability of the mechanism and involved computational overhead have not been checked |
| [72] Dynamic Non-heuristic | Reactive (Job migration) N/A | Network and physical faults Reliability and availability | It has not been evaluated in terms of resource utilization |

**TABLE 6.** A comparison of load balancing approaches based on considered parameters (↓: Decreased, ↑: Increased)
**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 5% (one method [70]) of the methods have been done based on a static manner.

**FIGURE 5.** Percentage of adopted dynamic or static approach

**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, 86% of the researchers have chosen a non-heuristic algorithm in their proposed innovation.
FIGURE 6. Percentage of adopted heuristic or meta-heuristic approach

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 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.

FIGURE 7. Percentage of adopted basic fault-tolerance approach

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, 43% of the papers have attempted to address network faults.

FIGURE 8. Percentage of adopted simulation tools

FIGURE 9. Percentage of type of detected faults

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. A side-by-side comparison of these approaches considering the relevant metrics is presented in Table 6.

FIGURE 10. Considered load balancing metrics in the reviewed methods

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 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 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.

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] 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,” Mob. Networks Appl., pp. 1–23, 2020.
[2] A. Souri, P. Asghari, and R. Rezaei, “Software as a service based CRM providers in the cloud computing: Challenges and technical issues,” J. Serv. Sci. Res., vol. 9, no. 2, pp. 219–237, 2017.
[3] 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, 2019.
[4] 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., pp. 1–14, 2019.
[5] 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, 2015.
[6] M. Ashouraie and N. Jafari Navimipour, “Priority-based task scheduling on heterogeneous resources in the Expert Cloud,” Kybernetes, vol. 44, no. 10, pp. 1455–1471, 2015.
[7] 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,” Futur. Gener. Comput. Syst., vol. 86, pp. 81–98, 2018.
[8] 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, 2019.
[9] A. Iqbal and R. Colomo-Palacios, “Key Opportunities and Challenges of Data Migration in Cloud: Results from a Multivocal Literature Review,” *Procedia Comput. Sci.*, vol. 164, pp. 48–55, 2019.

[10] 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.*, pp. 1–24, 2021.

[11] B. Pourghebleh and V. Hayyolalam, “A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things,” *Cluster Comput.*, pp. 1–21, 2019.

[12] J. Prassanna and N. Venkataraman, “Threshold Based Multi-Objective Memetic Optimized Round Robin Scheduling for Resource Efficient Load Balancing in Cloud,” *Mob. Networks Appl.*, vol. 24, no. 4, pp. 1214–1225, 2019.

[13] P. Tyagi and A. Kishor, “Load Balancing in Cloud Computing,” *Int. J. Comput. Sci. Eng.*, vol. 6, no. 10, pp. 22–27, 2018.

[14] M. Amoon, N. El-Bahnsawy, 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. Humaniz. Comput.*, vol. 10, no. 11, pp. 4567–4577, 2019.

[15] 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,” *Mob. Networks Appl.*, vol. 24, no. 3, pp. 1063–1077, 2019.

[16] M. N. Cheraghliou, A. Khademzadeh, and M. Hagharast, “New fuzzy-based fault tolerance evaluation framework for cloud computing,” *J. Netw. Syst. Manag.*, vol. 27, no. 4, pp. 930–948, 2019.

[17] S. B. Shaw and A. K. Singh, “A survey on scheduling and load balancing techniques in cloud computing environment,” in *Proceedings - 5th IEEE International Conference on Computer and Communication Technology, ICCCT 2014*, 2015, pp. 87–95.

[18] A. A. S. Farrag, S. A. Mahmoud, and E. S. M. El-Horbaty, “Intelligent cloud algorithms for load balancing problems: A survey,” in *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2016, pp. 210–216.

[19] S. Aslam and M. A. Shah, “Load balancing algorithms in cloud computing: A survey of modern techniques,” in *2015 National Software Engineering Conference, NSEC 2015*, 2016, pp. 30–35.

[20] M. Mesbahi and A. Masoud 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, 2016.

[21] 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, 2016.

[22] P. K. Tiwari and S. Joshi, “A review on load balancing of virtual machine resources in cloud computing,” in *Smart Innovation, Systems and Technologies*, 2016, vol. 51, pp. 369–378.

[23] E. Jafarnejad Ghomi, A. Masoud Rahmani, and N. Nasih Qader, “Load-balancing algorithms in cloud computing: A survey,” *J. Netw. Comput. Appl.*, vol. 88, pp. 50–71, 2017.

[24] M. O. Ahmad and R. Z. Khan, “Load balancing tools and techniques in cloud computing: A systematic review,” in *Advances in Intelligent Systems and Computing*, vol. 554, Springer, 2018, pp. 181–195.

[25] A. Hota, S. Mohapatra, and S. Mohanty, “Survey of Different Load Balancing Approach-Based Algorithms in Cloud Computing: A Comprehensive Review,” in *Advances in Intelligent Systems and Computing*, vol. 711, Springer, 2019, pp. 99–110.

[26] K. R. Jothis, S. Anto, M. Kohar, M. Chadha, and P. Madhavan, “Smart Load Balancing Algorithms in Cloud Computing—A Review,” *Role Edge Anal. Sustain. Smart City Dev. Challenges Solut.*, pp. 189–218, 2020.

[27] 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. Humaniz. Comput.*, pp. 1–30, 2020.

[28] T. Mostafaei, F. Modarres Khiyabani, and N. J. Navimipour, “A systematic study on meta-heuristic approaches for solving the graph coloring problem,” *Comput. Oper. Res.*, vol. 120, p. 104850, 2020.

[29] B. Pourghebleh, V. Hayyolalam, and A. Aghaei Anvigh, “Service discovery in the Internet of Things: review of current trends and research challenges,” *Wirel. Networks*, Oct. 2020.

[30] A. Tchernykh, U. Schwiegelsohn, E. Talbi, and M. Babenko, “Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability,” *J. Comput. Sci.*, vol. 36, p. 100581, 2019.

[31] F. Firouzi and B. Farahani, “Architecting IoT Cloud,” in *Intelligent Internet of Things*, Springer, 2020, pp. 173–241.

[32] Geeta and S. Prakash, “A literature review of qos with load balancing in cloud computing environment,” *Adv. Intell. Syst. Comput.*, vol. 654, pp. 667–675, 2018.

[33] B. Liu et al., “Information fusion in a cloud
computing era: a systems-level perspective,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 29, no. 10, pp. 16–24, 2014.

[34] 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, 2020.

[35] M. S. A. 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, 2018.

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

[37] U. Dwivedi and H. Dev, “A Review on Fault Tolerance Techniques and Algorithms in Green Cloud Computing,” *J. Comput. Theor. Nanosci.*, vol. 15, no. 9–10, pp. 2689–2700, 2018.

[38] A. Tikotekar, G. Vallée, T. Naughton, S. L. Scott, and C. Leangsuksun, “Evaluation of fault-tolerant policies using simulation,” in *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, 2007.

[39] F. Salfner, M. Lenk, and M. Malek, “A survey of online failure prediction methods,” *ACM Computing Surveys*. 2010.

[40] S. Prathiba and S. Sowvarnica, “Survey of failures and fault tolerance in cloud,” in *2017 2nd International Conference on Computing and Communications Technologies (ICCCCT)*, 2017, pp. 169–172.

[41] M. A. Mukwevho and T. Celik, “Toward a smart cloud: A review of fault-tolerance methods in cloud systems,” *IEEE Trans. Serv. Comput.*, 2018.

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

[43] P. Kumari and P. Kaur, “A survey of fault tolerance in cloud computing,” *J. King Saud Univ. Inf. Sci.*, 2018.

[44] A. Semmoud, M. Hakem, B. Benmammar, and J. C. Charr, “Load balancing in cloud computing environments based on adaptive starvation threshold,” *Concurrency Comput., No. May 2019*, pp. 1–14, 2020.

[45] M. R. Mesbah, A. M. Rahmani, and M. Hosseinzadeh, “Reliability and high availability in cloud computing environments: a reference roadmap,” *Human-centric Comput. Inf. Sci.*, vol. 8, no. 1, p. 20, 2018.

[46] Y. Sharma, B. Javadi, W. Si, and D. Sun, “Reliability and energy efficiency in cloud computing systems: Survey and taxonomy,” *J. Netw. Comput. Appl.*, vol. 74, pp. 66–85, 2016.

[47] V. Mohammadian, N. Jafari Navimipour, M. Hosseinzadeh, and A. Darwesh, “Comprehensive and systematic study on the fault tolerance architectures in the cloud computing,” *J. Circuits, Syst. Comput.*, 2020.

[48] 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.

[49] M. M. Golchi, S. Saraeian, and M. Heydari, “A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation,” *Comput. Networks*, vol. 162, p. 106860, 2019.

[50] M. A. Shahid, N. Islam, M. M. Alam, M. M. Su’ud, and S. Musa, “A Comprehensive Study of Load Balancing Approaches in the Cloud Computing Environment and a Novel Fault Tolerance Approach,” *IEEE Access*, vol. 8, pp. 130500–130526, 2020.

[51] K. Cao et al., “Affinity-driven modeling and scheduling for makespan optimization in heterogeneous multiprocessor systems,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1189–1202, 2018.

[52] 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.

[53] 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,” 2014 Int. Conf. Recent Trends Inf. Technol. ICRTIT 2014, 2014.

[54] 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.

[55] A. Moghtadaeipour and R. Tavoli, “A new approach to improve load balancing for increasing fault tolerance and decreasing energy consumption in cloud computing,” *Conf. Proc. 2015 2nd Int. Conf. Knowledge-Based Eng. Innov. KBEI 2015*, pp. 982–987, 2016.

[56] C. A. Chen, R. Stoleru, and G. G. Xie, “Energy-efficient load-balanced heterogeneous mobile cloud,” *2017 26th Int. Conf. Comput. Commun. Networks, ICCCN 2017, 2017*.

[57] N. Jayapandian, R. Menagadevi, S. Abinaya, O. S. Sampoornani, and A. M. J. Z. Rahman, “Dynamic Load Balancing Cluster and Fault-Tolerant In Cloud Environment Dynamic Load Balancing Cluster and Fault-Tolerant In Cloud Environment,” *Comput. Eng.*, vol. 103, no. May, pp. 44520–44523, 2017.

[58] 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.

[59] A. Ragmani, A. El Omri, N. Abghour, K. Moussaid, and M. Rida, “An efficient load balancing strategy based on MapReduce for public cloud,” *ACM Int. Conf. Proceeding Ser.*, pp. 1–10, 2017.

[60] B. Balusamy, K. Karthikeyan, and A. K. Sangaiah, “Ant colony-based load balancing and fault recovery for cloud computing environment,” *Int. J. Adv. Intell. Paradig.*, vol. 9, no. 2–3, pp. 204–219, 2017.

[61] 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.

[62] S. Sreelekhsmi and K. R. Remesh Babu, *Fault tolerant multiple synchronized parallel load balancing in cloud*, vol. 734. Springer International Publishing, 2018.

[63] T. Tamilvizhi and B. Parvathavarthini, “A novel method for adaptive fault tolerance during load balancing in cloud computing,” *Cluster Comput.*, vol. 22, pp. 10425–10438, 2019.

[64] F. Ebadifard, S. M. Bahamir, and S. Barani, “A dynamic task scheduling algorithm improved by load balancing in cloud computing,” in *2020 6th International Conference on Web Research (ICWR)*, 2020, pp. 177–183.

[65] H. Jahanpour, H. Barati, and A. Mehranadeh, “An Energy Efficient Fault Tolerance Technique Based on Load Balancing Algorithm for High-Performance Computing in Cloud Computing,” *J. Electr. Comput. Eng. Innov.*, vol. 8, no. 2, pp. 169–182, 2020.

[66] 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.

[67] S. Setouhi, D. Djamel Amar Bensaber, R. Adjoudj, and M. Rebbah, “Fault Tolerance Directed by Service Level Agreement in Cloud Computing,” *Int. J. Comput. Digit. Syst.*, 2021.

[68] A. Roy, “Dynamic load balancing: improve efficiency in cloud computing,” *Int. J. Emerg. Res. Manag. &Technology*, vol. 9359, no. 4, pp. 78–82, 2013.

[69] 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.

[70] 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, 2017.

[71] A. Rezaeipanah, M. Mohorad, and A. Fakhari, “Providing a new approach to increase fault tolerance in cloud computing using fuzzy logic,” *Int. J. Comput. Appl.*, pp. 1–9, 2020.

[72] S. M. A. Attallah, M. B. Fayek, S. M. Nassar, and E. E. Hemayed, “Proactive load balancing fault tolerance algorithm in cloud computing,” *Concurr. Comput. Pract. Exp.*, vol. 33, no. 10, p. e6172, 2021.

[73] A. Ray, “Dynamic Load Balancing: Improve Efficiency in Cloud Computing,” *Int. J. Emerg. Res. Manag. &Technology*, vol. 9359, no. 24, pp. 2278–9359, 2013.