IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 May 2022, accepted 2 July 2022, date of publication 5 July 2022, date of current version 12 July 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3188645*

TOPICAL REVIEW

# Cloud Dynamic Load Balancing and Reactive Fault Tolerance Techniques: A Systematic Literature Review (SLR)

**TAWFEEG MOHMMED TAWFEEG[1], ADIL YOUSIF[2], ALZUBAIR HASSAN[3,4], SAMAR M. ALQHTANI[5], RAFIK HAMZA[6], MOHAMMED BAKRI BASHIR[7,8], AND AWAD ALI[2]**

[1]Department of Computer Science, Faculty of Computer Science and Information Technology, University of Science and Technology, Khartoum 14411, Sudan
[2]Department of Computer Science, College of Science and Arts-Sharourah, Najran University, Sharourah 68341, Saudi Arabia
[3]Department of Computer Science, School of Computer Science and Informatics, University College Dublin, Dublin, D04 V1W8 Ireland
[4]Lero-the Irish Software Research Centre, University of Limerick, Limerick, V94 NYD3 Ireland
[5]Department of Information Systems, College of Computer Science and Information Systems, Najran University, Najran 61441, Saudi Arabia
[6]Institute for International Strategy, Tokyo International University, Saitama 350-1197, Japan
[7]Department of Mathematics, Turubah University College, Taif University, Taif 26571, Saudi Arabia
[8]Department of Computer Science, Faculty of Computer Science and Information Technology, Shendi University, Shendi 41601, Sudan

Corresponding author: Adil Yousif (ayalfaki@nu.edu.sa)

This work was supported by the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia through the Institutional Funding Committee at Najran University, Saudi Arabia, under Project NU/IFC/ENT/01/013.

**ABSTRACT** The computational cloud aims to move traditional computing from personal computers to cloud providers on the internet. Cloud security represents an important research area. Confidentiality, integrity, and availability are the main cloud security characteristics addressed. Cloud providers apply dynamic load balancing and reactive fault tolerance techniques to build secure cloud services to achieve high service availability. Dynamic cloud load balancing approaches distribute submitted tasks to virtual machines during tasks execution and the load of virtual machines is updated based on the system's state. Reactive cloud fault tolerance is activated for system process failures after failure effectively happens. Reactive cloud fault tolerance handles failure after the fault has occurred. Despite the significance of dynamic load balancing and reactive fault tolerance techniques and mechanisms, few reviews focus on these approaches in a systematic, unbiased method focusing on integrating cloud dynamic load balancing and reactive fault tolerance techniques. This paper conducts a systematic literature review of the existing literature concerning reactive fault tolerance, dynamic load balancing, and their integration in their basic approaches, types, frameworks, and future directions.

**INDEX TERMS** Dynamic load balancing, reactive fault tolerance, cloud, systematic literature review.

## I. INTRODUCTION

Cloud computing is based on five features: elasticity, multitenancy or shared resources, scalability, pay as you go, and self-provisioning of resources [1]. Elasticity allows customers to increase and reduce their hardware and software resources as required speedily. In multitenancy, cloud computing relies on a practical model in which software and hardware resources area and units have been shared at network, host, and application levels. Organizations may need thousands of systems to obtain massive scalability, and cloud computing allows them to scale thousands of systems. Cloud computing provides organizations of on demand number of virtual nodes to scale bandwidth and storage capacity massively. In pay as you go scheme, users procure the hardware and software resources solely once they need them. Finally, Self-provisioning allows users specified resources, such as different systems processing capabilities, software, storage, and network resources. A common framework for describing cloud computing services is defined as SPI [1]. Figure 1

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

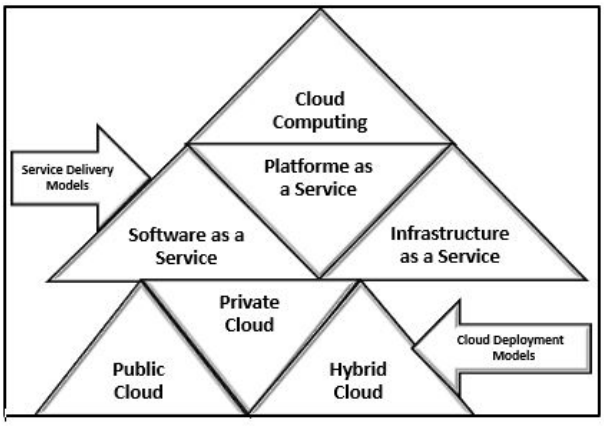


**FIGURE 1.** SPI service model.

illustrates the connection between services, uses, and kinds of clouds.

The first cloud service model is software as a service (SaaS). SaaS cloud suppliers deliver software hosted on the cloud as web-based primarily services for end-users without installation of the appliance on the customer's device. The second cloud service model is Platform as a Service (PaaS). In PaaS cloud suppliers deliver platforms, tools, and different business services that allow customers to develop, deploy and manage their applications. In PaaS, cloud customers need not place any platform and support tools on their native machines. The third cloud service model is the infrastructure as a service (IaaS). In IaaS the computation resources, storage, and network are delivered by cloud suppliers as web-based services.

Security techniques are required to provide cloud service availability. On of the main security technique used to maintain cloud availability is the load balancing and fault tolerance schemes. Load balancing techniques are implemented in the cloud to ensure that nodes are balanced and not overloaded. In addition, cloud fault tolerance techniques aim to handle cloud system failures after and before they occur [2]. Techniques such as load balancing help provide availability over the cloud. It distributes the dynamic workload all over data-centers fairly to ensure that no data-center was overloaded [3].

Few existing work have reviewed cloud load-balancing and fault tolerance in a systematic approach, unbiased method focusing on the relationship between dynamic cloud load balancing and reactive fault tolerance techniques [4] The study in [5]presented a systematic review of fault tolerance techniques that have been proposed in open source clouds. A systematic review of different fault-tolerance methods is discussed in [6]. The study in [6] claimed that checkpointing, migration techniques and replication schemes are the primary fault-tolerance techniques. The study's authors in [7] conducted a systematic review of multiple fault types. Their study handled the reason and different fault tolerance methods involved in the cloud and provided a comparative analysis of the review frameworks. Similarly, in [8] the study performed an SLR of the current load balancing methods and provided a comparative evaluation analysis.

Cloud providers have applied load balancing methods and cloud fault tolerance techniques to build secure cloud services by achieving high service availability. Dynamic cloud load balancing approaches distribute submitted tasks to virtual machines during tasks execution and the load of virtual machines is updated based on the system's state. Reactive cloud fault tolerance is activated for system process failures after failure effectively happens. Reactive cloud fault tolerance handles failure after the fault has occurred. However, despite the importance of load balancing and fault tolerance techniques and mechanisms, very few reviews focus on these approaches in a systematic, unbiased method focusing on the relationship between cloud dynamic load balancing and reactive fault tolerance techniques. This study conducts an SLR of the state of the art literature concerning cloud reactive fault tolerance, dynamic load balancing, and the relation between fault tolerance and load balancing, considering their fundamental methods, types, frameworks, and future directions.

The rest of this paper is structured as follows: Systematic literature methodology is presented in Section II. The cloud fault tolerance is reviewed in Section III. The review of cloud load balancing is illustrated in Section IV. The relation between cloud fault tolerance and load balancing is presented in Section V. Section VI concludes the paper and presents the future directions and conclusion.

## II. SYSTEMATIC LITERATURE REVIEW

This section illustrates the elements of the systematic literature review, including research questions, review protocol, and the search strategy.

### A. NEEDS FOR SYSTEMATIC LITERATURE REVIEW

The computational cloud is developing as the next computational generation representing a platform for various user services. Nevertheless, significant issues need to be defined to obtain cloud security, for example, integrity, data confidentiality, and cloud service availability. Availability of cloud service providers needs good management decisions to avoid outages or failures such as hardware failure and denials of service attacks during responses to client requests.

Numerous load balancing methods and fault tolerance schemes are developed for cloud computing security. However, few types of research provide a review on security issues related to availability services in cloud computing, such as fault tolerance or load balancing, without clarifying the relation between cloud fault tolerance and load balancing.

This study aims to systematically review and analyze the two techniques, fault tolerance and load balancing, and link them. Furthermore, the study illustrates the state of arts fault tolerance and load balancing algorithms by focusing on the execution times, methods of evaluation and simulation and the primary measures used in the evaluation process. Moreover, the study provides an unbiased SLR of the current fault tolerance and cloud load balancing methods. The systematic

review in this paper presents an overview of current challenges and issues that faces cloud services availability using load balancing methods and cloud fault tolerance solution. Finally, the systematic review explores the future challenges for cloud computing and the roles and contributions of load balancing and fault tolerance.

### B. RESEARCH QUESTIONS

Through studying, understanding, and analyzing the techniques, models, and algorithms used to achieve fault tolerance and load balancing, whether they are used with each other or individually, the objective of this study is to answer the following research questions:

1) what are the primary techniques that are used in achieving fault tolerance, load balancing, and their integration;
2) what are the differences between fault tolerance and load balancing techniques using a comparative analysis;
3) does load balancing is considered a technique used to achieve fault tolerance? or are they two separate technologies that can be run together for achieving high availability;
4) what are the evaluation methods that are used in assessing fault tolerance and load balancing techniques;
5) what do researchers recommend the future directions in this area;

### C. REVIEW PROTOCOL AND THE SEARCH STRATEGY

The research followed a systematic review methodology, the study protocol is described in Figure 2.

#### 1) STEP 1 CREATING THE SEARCH STRATEGIES

This step defined which keywords will be used in the selected search tools in the standard and advanced search. For this work, the study utilized the following expressions:

1) what are the primary techniques that are used in achieving fault tolerance, load balancing, and their combination;
2) fault tolerance " AND " reactive " AND" algorithm " AND " cloud;
3) dynamic load-balancing algorithm and reactive cloud fault tolerance methods.

#### 2) STEP 2 DEFINING SEARCH SOURCES

A manual search was conducted on different libraries of computer science publishers. IEEE Computer Society, Science Direct, Springer, and ACM Digital Library were used to perform the SLR search.

#### 3) STEP 3 INCLUSION AND EXCLUSION CRITERIA FOR THE SELECT STUDIES

In this step, the review identified the primary studies from search results to be included or excluded based the criteria described in Table 1.

**TABLE 1. Criteria information.**

| Criteria | Inclusion | Exclusion |
|---|---|---|
| The date of journals and conferences articles published | 2015-2021 | All except 2015-2021 |
| Topic coverage | Reactive cloud fault tolerance Dynamic cloud load balancing | Proactive cloud fault tolerance Static cloud load balancing |
| Evaluation | Method studies with evaluation Simulation/Real implementation results | Theoretical, Review, State of Arts, Surveys, Taxonomy, Methods without evaluation and results |

#### 4) STEP 4 INITIAL DATA EXTRACTION TO EVALUATE THE STUDIES

This step extracts data based on a set of items to be filled for each article, such as the abstract, the proposed model, and future works.

#### 5) STEP 5 QUALITY OF STUDIES ASSESSMENT

To evaluate the quality of studies, the review protocol stated the following four scores:

- Score one: Does the research answer the research questions correctly?
- Score two: Have the research problem and contributions represented Cleary?
- Score three: Are data collection and analysis methods conducted to handle the study area and fairly described?
- Score four: Are the experiments and simulation results measured and evaluated?

#### 6) STEP 6 EXTRACT AFTER STUDYING

This Step involves applying a detailed data extraction procedure for studies defined in Step 4 to the primary studies selected in step 5.

#### 7) STEP 7 SYNTHESIZE DATA AND WRITE THE REPORT (DOCUMENTING)

In this Step, the study reviews all included articles to classify and clarify them according to the research questions. The result of this Step is presented in a section named "load balancing and fault tolerance techniques algorithms and models."

#### 8) STEP 8 DOCUMENTING COMPARATIVE ANALYSIS (META-ANALYSIS)

After the studies review, the paper developed a discussion to answer the research questions. The results of this Step are presented as:

1) A crucial assessment for cloud dynamic load balancing methods;
2) Comparative analysis for fault tolerance models;
3) A critical evaluation for cloud dynamic load balancing methods that include fault tolerance techniques;
4) Comparative analysis for the evaluation methods in dynamic load balancing methods and cloud fault tolerance schemes;
5) The relation between reactive fault tolerance methods and dynamic cloud load balancing schemes;

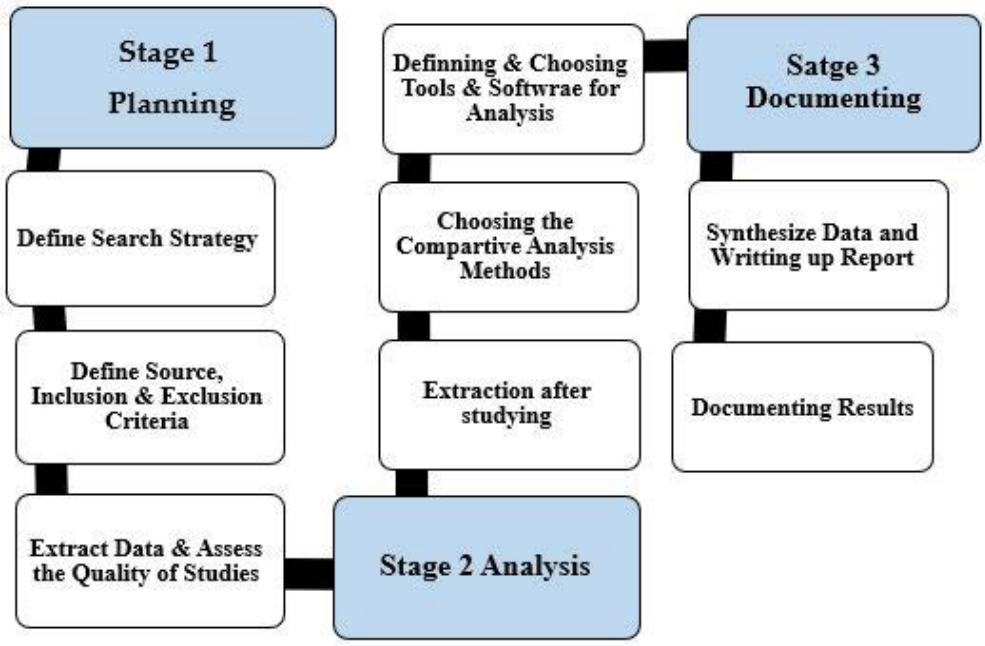


**FIGURE 2.** SLR review protocol.

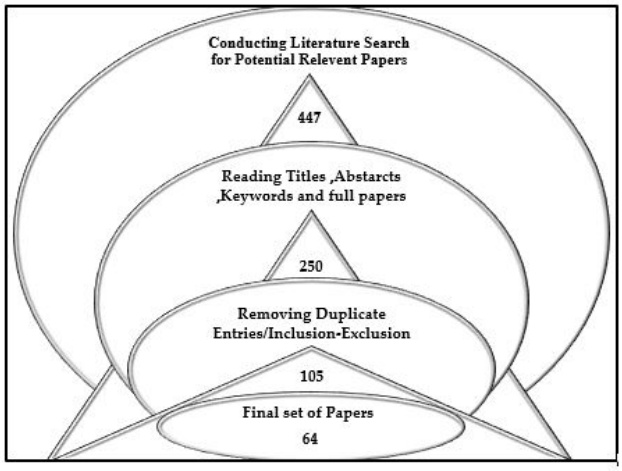


**FIGURE 3.** Papers selection process.

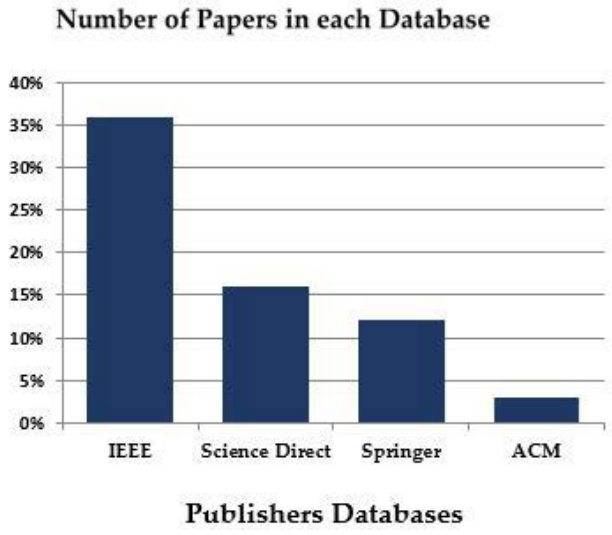


**FIGURE 4.** Number of papers in each database.

6) The future directions in this area and conclusion;

Figure 3 shows the process of papers selection and the number of selected papers in each step. Furthermore, Figure 4 shows the number of documents in each database.

## III. CLOUD REACTIVE FAULT TOLERANCE

This section illustrates the review of reactive cloud fault tolerance researchers and the comparative analysis. This section is organized into the following subsections:

### 9) FAULT TOLERANCE

Any computing failure in a very computer system will cause a system crash. for instance, suppose a computer system is employed in some essential areas, like operational a heavier-than-air craft and dominant nuclear energy plants. In this case, the crash of the pc system might end in a disaster. Therefore, developing a reliable system could be a difficult issue. Nevertheless, heaps of effort have been taken to style and implement a reliable system that gives continuous, correct, and secure services [9].

The term fault tolerance describes the power of a system to retort to an unexpected computing failure. There are several levels of fault tolerance, the bottom being the power to continue operating in the event of an influence failure, the upper

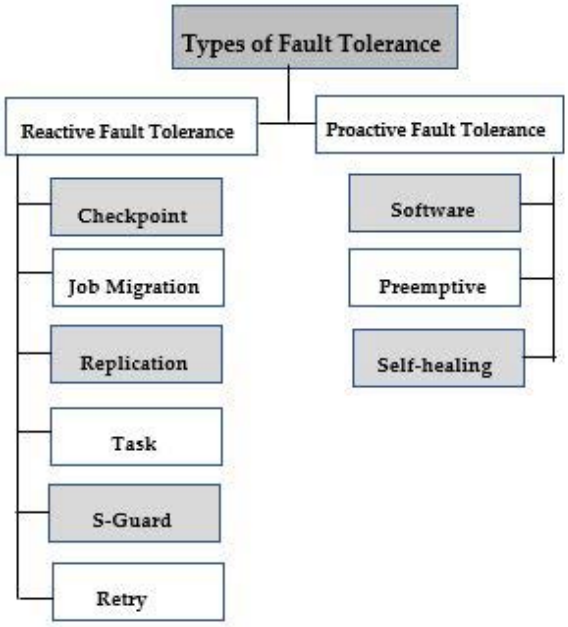


**FIGURE 5.** SPI service model.

being the power to confirm strictly correct behaviour despite sure sorts of fault [10].

### A. TYPES OF FAULT TOLERANCE

Cloud fault tolerance is classified into two main types, hardware and software. In the hardware cloud fault tolerance methods, the processes of the systems are guided automatically to recover from failure initiated by hardware parts. Each hardware element of the system is stored in a safe place, so that if one of the components fails, the backup can be activated to perform the functions. The general hardware and software fault tolerance schemes are categorized into reactive and proactive schemes [11]. N-version implementation utilizes static redundancy in separate written systems that perform the same operation. It's meant to make a backup for application tasks [12].

### B. FAULT TOLERANCE TECHNIQUES

Figure 5 shows the different fault tolerance techniques, methods and classifications [9], [10], [13]:

### C. PROACTIVE FAULT TOLERANCE

The concepts of proactive computational cloud fault tolerance techniques calculate the fault and avoid the retrieval process from fault, errors, and failures. This process is achieved by replacing the suspected component that has been detected before faults occur. In Software Rejuvenation, the system periodically reboots; the process reboots the system with a clear state and helps to scrub begin [11], [13].

Preemptive migration is preventive migration that estimates a feedback-loop management mechanism. The appliance is consistently monitored and analyzed [11], [13]. In Self-Healing, the big tasks are divided into small components. This division process is accomplished for higher performance. Once various instances of associate in Nursing application square measure running on various virtual machines, it mechanically handles failure of application instances [11], [13].

### D. REACTIVE FAULT TOLERANCE

Reactive computational cloud fault tolerance is activated for failures of application execution after failure essentially happens. Reactive cloud fault tolerance handles failure after the fault has occurred. Checkpointing is a reactive fault tolerance, during which the system writes each modification in the system a checkpoint. This process is building an image of the system state. Once a task fails, checkpointing technique restarts that task from the last stop point taken instead of resuming the task from starting point [10], [13].

Job Migration is another reactive fault tolerance technique. In the job migration technique, once a task cannot complete its execution on a specific host, resulting from one fault that happened to that host at the time of failure. The task is migrated to a different host. Job migration is often enforced in tools such as HA-Proxy [10], [13].

Replication reactive fault tolerance are the techniques that employ replication concepts. Various tasks are duplicated and run on multiple hosts to obtain the required results. Replication is often enforced in tools such as HA-Proxy, Hadoop, and AmazonEc2 [10], [13].

In Safety-bag checks, the obstruction of commands is completed. This process is accomplished for the commands that don't meet the protection properties. The S-Guard, relies on rollback recovery. Retry is the only method that repeats the unsuccessful jobs on a similar machine. During this case, task resubmission could fail whenever unsuccessful jobs are identified. If this case occurs at the operating time, the jobs are reallocated to a similar or a different host for operation [10], [13].

### E. METRICS THAT USED IN FAULT TOLERANCE EVALUATION

Multiple metrics are used to measure and analyze the fault tolerance [10], [13]. All processes are mechanically executed in step with the conditions in the adaptive measurement. Performance measurement is applied to measure the strength of the applications. It's to be enhanced at an affordable value, e.g., amend back response time while maintaining appropriate delays. Response time is defined as the time needed to reply using a specific algorithmic program. The throughput computes the number of jobs with successful execution. Reliability aims to produce correct and acceptable results in a particular time interval. Availability describes the system's chance of functioning properly once requested or intended to be used. In the usability measurement, clients use an associate degree of innovation to complete the jobs expeditiously and effectively.

### F. FAULT TOLERANCE MODELS

This section discusses and analyses cloud fault tolerance models, their features, ad-vantages, and limitations. Low latency fault tolerance (LLFT) may be a model introduced to deliver fault tolerance with an occasional delay. This model has provided fault tolerance and the capability to develop data-centers. LLFT uses the master/slaves replication technique. In LLFT, each group's primary process is selected to represent the group backup. The communication between members of groups occurs using virtual contacts. The original version of the goal group provides for implementing the manage orders to perform unsure operations performed by backup replicas [14], [15]. The advantage of LLFT is the low overhead that makes it appropriate for distributed software implemented in cloud providers or datacenters. A new model is proposed for fault tolerance cloud workflow job scheduling (FTWS). FTWS uses activation and spread methods, as stated by the importance of jobs [7], [14]. As a result, FWSS keep an effective resource utilization and task accomplishment rate. However, FTWS is not valid to all cloud systems.

A model based on adaptive cloud fault tolerance systems (AFTRS) for real-time considers jobs execution time when jobs are received and stored in an input buffer for processing. AFTRS is based on FCFS (first come, first serve) mechanism. Every job is duplicated in N virtual host, integrated with multiple mechanisms for real-time job operations. The outcome created by each mechanism is transferred for the acceptance test (AT), at which the rightness of the outcome is confirmed. Based on the obtained results, reliabilities of hosts are adapted using the reliability assessor (RA) component. Then, the decision scheme module (DM) chooses the output considering the data centre with the highest reliability [14], [16].

### G. FAULT TOLERANCE SIMULATION TOOLS

There are many tools and environments to simulate fault tolerance. Table 2 shows the different environments and tools used in cloud fault tolerance [10], [12], [13]:

### H. REACTIVE FAULT TOLERANCE FRAMEWORK

This section reviews several cloud fault tolerance mechanisms stated within the state of the art literature. Om Kumar C.U. et al [17]proposed fuzzy economic energy to enhance the management of workloads. The fuzzy economic energy mechanism scales back migration time and execution of instances in cloud computing by developing three steps for scheduling that maintain cloud resources. This framework aims to realize the workload consolidation and deploy fuzzy decision-makers to measure resources with utilization concerns. Furthermore, the framework provides the workload categories to observe the failure of virtual machines and start the fuzzy migration of virtual machines.

Amoon *et al.* [18] presented a cloud fault tolerance model using checkpoint technique to improve the cloud efficiency when failures occur. The approach applies a flexible length of checkpointing. The length of the checkpointing for an application can rely upon the magnitude relation of server failures that adapt to the virtual machine. Sivagami and Easwarakumar [19] introduced a new strategy for improvement management known as dynamic fault-tolerant VM migration (DFTM) to control data center infrastructure reliability. DFTM is based on an advanced recovery mechanism of VN (Virtual Network) demands. The submission of jobs to resources depends on monitoring the network traffic and load limit through VM.

Mohammed *et al.* [20] proposed a fault tolerance model intended to handle cloud resource fauilures primarily using the reliability of each resource using checkpoint techniques. The algorithm pro-posed by Yao *et al.* [21] is a scheduling mechanism known as a Hybrid Fault-Tolerant scheduling algorithm program (HFTSA). The HFTSA scheduling algorithm selects a cloud fault-tolerant method from resubmissions and replications for each specified job. These tasks support task and cloud resources' characteristics and adapt the appropriate resources.

AbdElfattah *et al.* [22] introduced a fault tolerance model using replication and resubmission techniques. The model determines the simplest virtual machine, counting on the reliability assessments. Even if the fault happens, the failed task has a replica to be activated. Amoon [23] introduced a cloud fault tolerance framework focusing on adaptability. The framework utilizes replica and checkpoint strategies over replication to maintain cloud provider resources. Moreover, the algorithm determines each virtual machine's most ap-plicable fault tolerance methodology.

Hasan and Goraya [24] proposed a flexible cloud fault tolerance model (FFTF). FFTF framework provides users with a key to their jobs to activate the suitable degree of fault tolerance (FT). User tasks are performed on a shared cluster in the cloud to implement an FT level. The FFTF efficiency is tested and investigated through large simulation testing on artificial and real workloads regarding its FT capability and resource consumption. To predict Byzantine failures, Beheshti and Esfahani [25] proposed a BFPF-Cloud framework with multiple features using support vector machine (SVM) mechanism. BFPF-Cloud framework has a reactive and a proactive policies to han-dle failures and maintain reliability and system availability.

Alaei *et al.* [26] introduced an adaptive model aims to decrease the, the makespan, energy consumption and the total cost, and tolerate faults. Mohammed *et al.* [27] proposed an integrated virtualized failover strategy (IVFS) that tolerates cloud failure using the reliability of for datacenter or virtual resources. The proposed model removes the computing data-center or virtual host from the availability list if the efficiency of the re-source is less than the standard performance.

Table 3. summarizes a comparative analysis of each reactive fault tolerance framework. And table 4 shows the metrics the used in each framework.

Fault tolerance is a technique used to achieve and provide service availability on cloud computing. In this review, many

**TABLE 2.** Fault tolerance simulation tools.

| Tools name | Programming Language | Environment | Fault tolerance scheme |
|---|---|---|---|
| HAProxy Fault tolerance Tool | Java Programming Language | Virtual Machine | Task Migration scheme, Self-Healing method, and Replication technique |
| SHelp Fault tolerance Tool | SQL Database and Java programming Language | Virtual Machine | Cloud fault Checkpointing scheme |
| Assure Fault tolerance Tool | Java programming Language | Virtual Machine | Cloud Checkpointing scheme and fault tolerance Self-Healing method |
| Hadoop Fault tolerance Tool | HTML, Java, CSS | Cloud Environment | Job Migration, Rescue Workflow, Replication |
| Amazon EC2 Fault tolerance Tool | Amazon Machine Image, Amazon Map | Cloud Environment | Replication, Task Resubmission |

**TABLE 3.** Comparative analyses of each framework.

| Authors Names | The type of the proposed methods | Techniques Types | Methods | Applications | Advantages | limitations |
|---|---|---|---|---|---|---|
| Om Kumar C.U. et al. [17] | Adaptive fuzzy, fault tolerance | Reactive | Live migration | Cloudsim | Reduce migrations and execution time | Weak in failure aware allocation |
| Amoon et al. [18] | Enhancement | Reactive | checkpoint | ACS | Better performance | The checkpoint interval isn't fixed that make the availability is not high |
| Sivagami and Easwarakumar [19] | Enhancement | Reactive | migration | Cloudsim | Minimal complexity | Resource utilization low Migration time high |
| Mohammed et al. [20] | Enhancement | Reactive | checkpoint | Cloudsim | improved performance | Implementation in one datacenter |
| Yao et al. [21] | Hybrid | Reactive | Replication / resubmmission | Google Cloud tracelogs | high resource utilization | Low Response time lower resource utilization |
| AbdElfattah et al. [22] | Hybrid | Reactive | Replication / resubmmission | Cloudsim | Better performance | Low Response time lower resource utilization |
| Amoon [23] | Hybrid | Reactive | Replication / checkpoint | Cloudsim | Better performance Less overhead | lower resource utilization |
| Hasan and Goraya [24] | Hybrid | Reactive | Replication/ checkpoint | Cloudsim | High system scalability High resource utilization | Low resource consumption Low Response time |
| Beheshti and Esfahani [25] | Framework | Proactive/ Reactive | Replication/predicition | Cloudsim/WEKA | Decrease excution time, Decrease Repetaed time | Increse thorhput , Low response time |
| Alaei et al. [26] | hybrid | Proactive/reactive | Fuzzy/migration | real world workflows | minimize energy consumption makespan, total cost | Low response time |
| Mohammed et al. [27] | hybrid | reatcive | Redundancy / checkpoint / replay | clousim | Improve performance | lower resource utilization |

researchers are working in this field, whether by improving, proposing, or integrating technology with another to solve a problem facing fault tolerance. In this review, researchers working on reactive fault tolerance were covered. When an error occurs and how to deal with it. An analytical comparison was made between all researches under the study. The comparison aims to explain the proposal in general, the type of proposal, its advantages, the technology used, the limitations of the proposal, The environment in which it was applied, and the metrics used in the evaluation process. It is noticeable that most researchers use the two techniques of replication and checkpoint in their proposal due to their common use figure 6. shows the most used. However, each of them has its problems based on itself. This is also the tool most used in the application and simulation of the proposal is the cloudsim figure 7. Show the most used. Also, the measures used in the evaluation, we find that they target specific measures depending on their proposal. They did not use all the criteria that guarantee us an assessment of essential performance figure 8. shows the most metrics used.

## IV. LOAD BALANCING

TThis section reviews dynamic load balancing studies on the computational cloud and introduces a comparative analysis of dynamic load balancing methods.

### A. OVERVIEW OF LOAD BALANCING

Cloud load balancing methods are employed to balance the cloud load on multiple computers or clusters. Load balancing aims to achieve high system throughput, best resource utilization and de-creased job execution time. Moreover, load balancing prevents resource overload and reduces the resources'

**TABLE 4.** Evaluation metrics of frameworks.

| Authors Name | Adaptive | Performance | Response Time | Throughput | Reliability | Availability | Usability |
|---|---|---|---|---|---|---|---|
| Om Kumar C.U. et al. [17] | ✓ | × | × | × | × | × | × |
| Amoon et al. [18] | × | ✓ | ✓ | × | × | × | × |
| Sivagami and Easwarakumar [19] | × | × | ✓ | ✓ | ✓ | × | × |
| Mohammed et al. [20] | × | ✓ | × | × | × | × | × |
| Yao et al. [21] | × | × | ✓ | × | × | × | × |
| AbdElfattah et al. [22] | × | ✓ | × | × | × | × | ✓ |
| Amoon [23] | × | ✓ | × | ✓ | × | ✓ | × |
| Hasan and Goraya [24] | × | ✓ | × | × | × | × | × |
| Beheshti and Esfahani [25] | × | × | × | ✓ | × | × | × |
| Alaei et al. [26] | ✓ | × | × | × | × | × | × |
| Mohammed et al. [27] | ✓ | × | × | × | × | × | × |

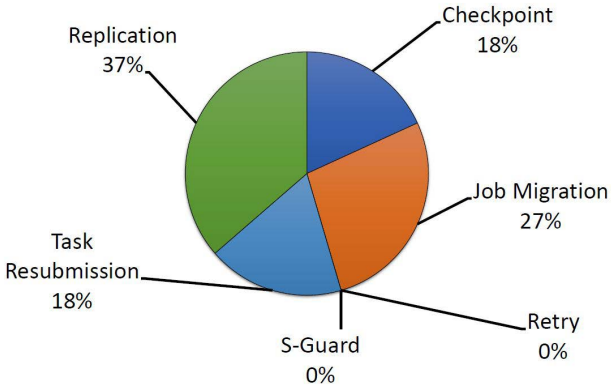


**FIGURE 6.** The most used techniques.

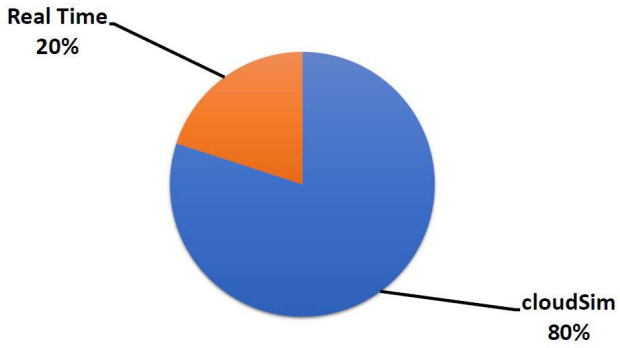


**FIGURE 7.** The most used tools.

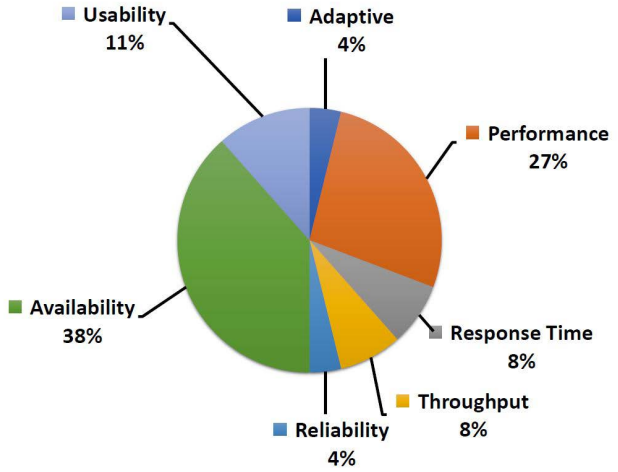


**FIGURE 8.** The most used metrics.

total waiting time [9]. Load balancing in clouds presents techniques to distribute the workload in a fairer way over all nodes. If load balancing schemes are implemented effectively, it affects fail-over, increases system scalability, prevents system bottlenecks, and decreases execution time [28]. The main objectives of load reconciliation include extending the performance, possessing a ready backup in case the system fails, and producing future improvements within cloud systems [23], [29]. Multiple metrics are employed for load balancing measurement and evaluation [30], [31].

### B. TYPES OF LOAD BALANCING ALGORITHM

Considering the process initiator, cloud load balancing methods are classified into three types. The first is the sender initiated when the sender starts the cloud load balancing process. The receiver initiated is when the cloud receiver starts the load balancing process. The last type is the Symmetric type which is a hybrid of each sender initiated, and receiver initiated [23], [28], [29]. Furthermore, several classifications of cloud load balancing are presented in the literature. Figure 9 shows a classification of cloud load balancing techniques. Figure 9 shows a classification of cloud load balancing techniques.

#### 1) STATIC LOAD BALANCING ALGORITHMS

The static cloud load balancing approaches distribute the jobs across virtual hosts before their execution time begins. These algorithms result in resource wastage by loading some virtual hosts at the running time. Static load balancing algorithms raise resource costs and increase Service level agreement violations caused by overload virtual hosts. Several static cloud load balancing approaches are proposed [32]. In round-robin load balancing [32], the shortest task is selected and executed first. The shortest job approach enhances the cloud performance by decreasing the waiting time for the tasks and this prevents the starvation, giving the algorithms an advantage over other approaches. Min-Min cloud load balancing is an algorithm in which all knowledge associated with tasks is

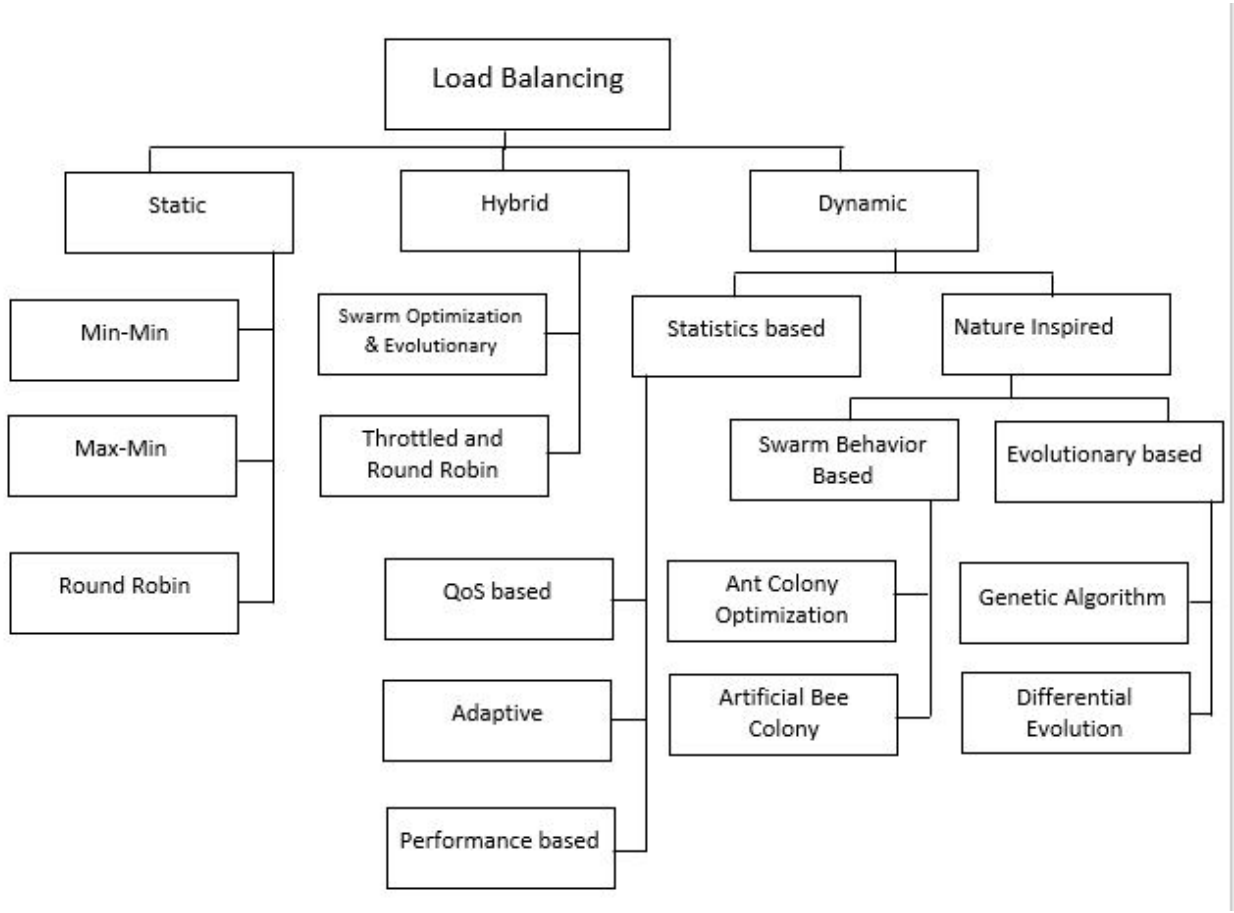


**FIGURE 9.** Load balancing classification.

obtainable in previous. Min-Min formula [33] starts with a collection of all uncompleted jobs.

Min-Min approach determined the time needed to complete each task. Min-Min approach chooses the minimum completion time for all tasks. Then Min-Min approach choose the resource with the least completion time among all jobs. In the last step the approach, maps each selected task. This method continues till all unallocated tasks are assigned. The benefit of this approach is that job which has a te shortest completion time is achieved first. The drawback of this approach is that some jobs might suffer from starvation. Max-Min cloud load balancing approach has an opposite scheme compared to the min-min method. In Max-Min, the most price is executed first. Max-Min [34] is virtually similar to min-min algorithm except that the maximum price is chosen once the lowest execution time of tasks is found.

### 2) DYNAMIC LOAD BALANCING ALGORITHMS

Dynamic cloud load balancing approaches distribute submitted tasks to virtual ma-chines during tasks execution as a load of VMs is updated based on the system's state. The previous data regarding system state is ignored in dynamic load balancing; this can overcome the drawbacks of the static methods. The cloud dynamic load balancing approaches are complicated.However, their performance and fault tolerance are improve compared to cloud load balancing approaches.Statistics based approach is applied to optimize workloads to measure the system performance. Several quality of service-based approaches were proposed to increase the efficiently of resources utilization. Swarm behavior-based and Evolution-Based are inspired by intelligent biological named nature-inspired and nature-based activities, which become two sub-groups. Evolutionary-based methods are presented for largest search space and complexity in the specific dynamic load balancing methods. Several Swarm intelligent approaches have been pro-posed for cloud dynamics load balancing. The power-aware load balancing (PALB) [35], starts by calculating off utilization proportion of each computing node. This process decides the amount of operational computing nodes as different nodes are switch off and not active. A new fuzzy active monitoring method [36] is proposed for dynamic load balancing based on fuzzy logic. The algorithm use two parameters loads on a virtual host and processor speed. Honeybee foraging behavior based on load equalization rule [37], of honey bees in finding and reaping their food. In this method a class of bees known as forager bees is utilized. Forager bees explore for food, and they return for an announcement when obtaining it. They

**TABLE 5. Evaluation tools.**

| Tools name | Programming | Enviroment |
|---|---|---|
| CloudAnalyst | Java | CloudSim |
| GroudSim | Java | ASKALON |
| GreenCloud | C++/OTel | NS 2 |

announce the food by doing a dance called the waggle dance. When receiving the data, the detector bees trace the searcher bees until found the food location for storage function. Then, bees come back to the beehive, and once more do a waggle dance to declare and informing other bees of available food.

The generalized priority algorithm [38] prioritizes the tasks with rules on task dimensions. The task with the largest size gets the best priority within the system and executes early. Conjointly the virtual hosts are prioritized based on their million instruction per second (MIPS ) in cloud hosts. Using this behavior, the host with the highest MIPS become and get highest priority.

#### 3) HYBRID LOAD BALANCING ALGORITHMS

Hybrid algorithms can be defined as combination of dynamic and static cloud load balancing algorithms features.

### C. METRICS THAT USED IN FAULT TOLERANCE EVALUATION

Multiple metrics were used for measuring and analyzing the load balancing [23], [29]. All processes are mechanically executed in step with the conditions in the adaptive measurement. Performance measurement is applied to compute the strength of the cloud. It's to be enhanced at an affordable value, e.g., amend back response time whereas preserve desirable delays. Response time measures the total time that the system absorbing to serve the task. The throughput represents the number of processes completed in right way. Makespan time calculates the highest completion time and it can be defined as the time needed to allocate resources to a user. System reliability aims to produce correct and acceptable results during a specific period. Availability describes the system's chance of functioning properly once requested or intended to be used. In the usability measurement, clients use an associate degree of a product and invention to accomplish the goal expeditiously, satisfaction, and effectiveness. Migration time determine the quantity of time needed to migrate a job from an overloaded host to another one. The degree of cloud load imbalance compute imbalance load among cloud VMs.

### D. LOAD BALANCING SIMULATION TOOLS

There are many tools and environments for simulating dynamic load balancing. Table 5 shows the different tools for dynamic load balancing [39]:

### E. DYNAMIC LOAD BALANCING FRAMEWORK

This section provides an overview of several dynamic load balancing algorithms proposed in the literature. Each framework is sufficiently explained in terms of the basic load balancing approach, the methodology used, the various implementation aspects, and the metrics used for the analysis with a comparative analysis. Zong et al [40] proposed a multi-workflow scheduling algorithm supported by reinforcement learning. A dynamic priority algorithm was used for task scheduling to support the type of progress. Therefore, a fine-grained cloud computing model was used for quality of service by using reinforcement learning to balance cluster nodes in cloud computing.

Ragmani *et al.* [41] proposed a combination algorithm called Fuzzy Ant Colony Improvement Algorithm Rule (FACO) for scheduling virtual hosts to increase efficiency in a cloud. The fuzzy module evaluates historical data to calculate the value of the pheromone. Then, a suitable server is selected to obtain the optimal computation time. The selection of the optimal parameters of the ant colony enhancement algorithms is based on the Taguchi experimental design presented in their experimental work.

Tong *et al.* [42] have presented a unique computational algorithm called Deep Q-learning task scheduling (DQTS), which combines the features of deep neural network and Q-learning algorithm. In a cloud environment, the approach is prepared to handle tasks with directed acyclic graphs (DAG). The central plan of their approach uses the preferred Deep Q-learning (DQL) technique in task scheduling, where DQL primarily stimulates elementary model learning that supports developments in the workflow.

Hamdani *et al.* [43] presented a load balancing algorithm that supported weighting of servers within the cloud platform by using fuzzy logic to represent different nodes in a cloud environment. Chaudhary and Kumar [44] presented a hybrid genetic-gravitational search algorithm (HG-GSA), which is a brand new technique for load balancing to reduce the total computational cost. The full process price includes the total cost of transmission and execution. It works with a hybrid crossover technique based mostly on a gravity search algorithm to find the most effective point of the particle in the search space. The most effective point of the particle is used to calculate the force.

Kashikolaei *et al.* [45] proposed a new hybrid between the imperialistic competition algorithm and the firefly algorithm. An intelligent meta-heuristic algorithm is used to handle user requests and task scheduling for load balancing. Improving scheduling and load balancing in cloud computing with local and global search algorithms was the goal of presenting the proposed algorithm.

Li *et al.* [46] presented a multi-objective task scheduling using a mixture of genetic algorithm and differential evolution algorithm (DE) to reduce and increase the overall time, price and load balancing for virtual machines. The use of DE within GA serves to take advantage of the global search capability of GA while accelerating the algorithm to find the optimal answer by taking advantage of the local search capability and fast convergence speed of DE.

Priya *et al.* [47] proposed an algorithm based on resource scheduling and load balancing. Their algorithm builds a

fuzzy-based four-dimensional resource scheduling model to increase the efficiency of resource scheduling in a cloud environment. Selecting requests from a class based on a multi-dimensional queue load improvement algorithm dynamically increases the utilization of virtual hosts through effective and fair load balancing. A load balancing algorithm is then enforced to achieve optimal resource utilization and avoid latency for each group of requests. To support the behavior of ant colony improvement (ACO) in rapid discovery of reasonable solutions and the rule of artificial bee colony algorithm (ABC) in collective interaction of bees and data sharing through tail dance. Gamal *et al.* [47] proposed a load balancing algorithm called hybrid artificial bee and ant colony (H_ BAC).

Kumar and Shukla [48] presented a fuzzy row penalty algorithm to solve an unbalanced load. The fuzzy row penalty technique is used to find any balanced fuzzy load balancing penalty and any unbalanced fuzzy load balancing penalty in the cloud environment. Khodar *et al.* [49] presented a strategy for planning using a genetic algorithm. Using the historical data and hence the current state of the system, this strategy computes in advance the impact on the system in allocating the desired VM resources. Adhikari and Amgoth [50] proposed an increased dynamic load balancing mechanism in the Infrastructure as a Service (IaaS) cloud. The algorithm first selects the most appropriate virtual machine (VM) instance for each task that supports the users' resource needs, and dynamically distributes the selected VM instance to the least loaded server in the cloud data center. The ED-LB algorithm aims to efficiently use resources on servers and balance the load in the cloud data center. Padmavathi and Basha [51] presented a completely new dynamic algorithm to achieve load balancing using ant colony enhancement to perform load balancing among the systems present in the data centers. Acharya [52] proposed an elevated dynamic load balancer based on the HTV load balancer that allows the user to input the number of hosts, VMs, and job requests as well as the type of application to understand the priorities for job execution. Garg *et al.* [53] developed a load balancing algorithm called Enhanced Active Monitoring Load Balancing (EAMLB) to reduce response time in the cloud.

Babu and Samuel [54] proposed an improved bee colony algorithm. In bees, search behavior is used to distribute the load among virtual machines. Tasks are pulled from full VMs and committed to VMs whose resources are underutilized. The strategy also attempts to reduce the span of VM migrations. Jena *et al.* [55] The authors presented a unique method for dynamic load balancing among virtual hosts by combining an improved Q-learning algorithm (QMPSO) and modified particle swarm optimization (MPSO). The hybrid approach is assigned to adjust the rate of MPSO over the gbest, and pbest supports the most effective action generated by the improved Q-learning. The goal of the hybrid approach was to improve the performance of the machine by distributing the load among the VMs, increasing the throughput of the virtual hosts, and maintaining the balance between the priorities of the tasks by optimizing the waiting time of the tasks.

Chitgar *et al.* [56] proposed dynamic stochastic cloud task scheduling (DSCTS) to distribute work among virtual machines. This technique selects a basic task and therefore filters the machines that could know it. The selection of the appropriate virtual machine supports the applied mathematics to perform the task. Patel and Bhalodia [57] planned a load balancing algorithm by combining a modified honeybee behavior algorithm to handle priority-based tasks and an extended weighted round robin algorithm to handle non-priority-based tasks.

Rajput and Kushwah [58] proposed an improved Load Balanced Min-Min (ILBMM) algorithm using a genetic algorithm (GA). The role behind using genetic algorithm rules is to solve the problem in the min-min algorithm of a larger task that has a longer waiting time by determining the execution time of the task on the virtual host primarily based on millions of instructions (MI) of the task and millions of instructions per second (MIPS) of the virtual hosts. Ghumman and kaur [59] proposed a combination of improved Max-Min and ant colony algorithm. The objective of Max-Min was based on execution time rather than completion time as the basis for selection.

Seddigh and Sharifian [60] proposed a hybrid Ant Colony Optimization (ACO) and VM Dynamic Forecast Scheduling (VM_DFS), a completely new algorithm known as Virtual Machine Dynamic Prediction Scheduling via Ant Colony Optimization (VMDPS-ACO) to solve the drawback of VM scheduling. With this algorithm, the historical memory consumption in each PM is analyzed to predict the future memory consumption of virtual hosts to perform economic allocation of virtual hosts in the cloud infrastructure. Sun *et al.* [61] presented a load balancing algorithm that is adaptive and calls itself adaptive ant colony optimization (SAACO). For the retreat of PACO, which was previously planned in their research, such as choosing parameters and updating pheromones, in SAACO to be self-adaptive, they also implement particle swarm optimization (PSO) to create the parameters of ACO.

Ebadifard *et al.* [62] proposed a dynamic task scheduling method based on Honeybee algorithm to enhance load balancing and reliability in cloud computing. Experimental results show that the proposed algorithm increased reliability and imbalance. Mandal et al [63] planned a unique load balancing strategy to search for underloaded nodes to balance the load from the fully loaded nodes. Azmat *et al.* [64] analysed the different performances of 3 existing algorithms, specifically Round-Robin (RR), Equal Spread Current Execution (ESCE) and the Throttled Algorithm (THR). Therefore, they have developed a new algorithm called Proposed DualIndex (PDI), which has higher performance and overcomes the problems of the existing algorithms. On the way to a brand new optimal LBA, they tend to keep two tables for virtual machines (VMs). One for the market VM and another for the busy VM, eliminating the process of multiple

scanning. This shortens the time required to search for VMs in the market, ultimately reducing the time span and cost.

Mardini and Enizat [65] proposed a completely new approach to load balancing that uses task assignment to virtual hosts by supporting virtual machine capabilities. Adhikariet *et al.* [66] proposed a completely new load balancing algorithm for a long term process called Load Balancing Resource Clustering (LBRC). They used the meta-heuristic Bat algorithm to find an optimal resource composition and its cluster data centres for a faster approach. They also propose a new dynamic task allocation policy to minimise the time span and execution cost under the given constraints. They proposed a completely new load balancing algorithm for a long term process called Load Balancing Resource Clustering (LBRC). They used the meta-heuristic Bat algorithm to find an optimal resource composition and its cluster data centres for a faster approach. They also propose a new dynamic task allocation policy to minimise the time span and execution cost under the given constraints.

Kshama and Shobha [67] proposed a completely new approach to load balancing that uses task assignment to virtual hosts by supporting virtual machine capabilities. Mohapatra *et al.* [68] proposed a forest optimization algorithm for load balancing in a distributed computing structure. This relies on the behavior of trees in the forest and uses seed propagation methods to rationalize the time span. Kumar et al [69] proposed a scheduling algorithm that supports priority of jobs and price of resources. Achieving high quality of service and optimal resource utilization is the goal of the IBA algorithm. Handling job priority and resource cost in IBA is not guaranteed. Therefore, a task scheduling algorithm based primarily on task priority and recourse cost is required.

Kumar and Sharma [70] have proposed a load balancing algorithm. The objective of this algorithm is to achieve optimal resource utilization and minimize the time margin. Kumar and Sharma [71] proposed a dynamic load balancing algorithm for distributing work across all virtual hosts with elastic resource provisioning and extraction that supports the last optimal k-interval. Sadia et al [72] proposed a load balancing algorithm by distributing the load of workloads among different virtual machines using priorities. The reason for implementing this algorithm is to balance different nodes by estimating the most throughput with minimum execution time. To achieve this, virtual machines are sorted by their process performance and task requests are assigned to virtual machines based on their millions of instructions per second and their priorities.

Kaur and Ghumman [73]proposed a load balancing algorithm for task scheduling that supports the process capacities of virtual machines (VMs) in cloud computing. Cheng et al [74] presented a self-adaptive technique for standardizing tasks, the ant colony, which seeks the appropriate configurations for each task running on different hosts. In heterogeneous clusters, Hymenopter first partitions the nodes and classifies homogeneous subclusters based on their hardware configurations. Then, each subcluster is treated as a homogeneous cluster and the self-optimization algorithm is applied to it separately. Finally, the tasks are configured with the configurations selected in this way, and the task configurations are gradually optimized by reproducing the configurations of the tasks with the best performance and ignoring the configurations with poor performance. In order to speed up the standardization of tasks and avoid accommodation in the local optimum, a genetic algorithm is used in adaptive task configuration.

kaushik et al [75] proposed an algorithm that solves the issue of resource utilization, performance degradation, and convenience. This method identifies however a user request arrives at the data centers and checks the quality of the request based on completely different parameters, namely bandwidth, response time, and throughput. It supports queues for storing requests, followed by a sorting thought. They tend to check the provisioning, size (capacity), MERT (minimum expected response time), and least busy parameters and select a VM for a request that increases the recourse utilization. Once we tend to check the least utilization parameters, we tend to use the thought of overloading or underloading the server with the thought of shifting. After completion of all requests, VMs self-destruct and release all resources BEN ALLA et al [76] proposed a novel design based on a new Dynamic Dispatch Queues Algorithm (DDQA) and Particle Swarm Optimization (PSO) formula for scheduling tasks in the cloud. The proposed algorithm DDQA-PSO fully accounts for the dynamic characteristics of the cloud computing environment.

Vanitha and Marikkannu [77] have proposed a new method for load balancing which involves well-organized resource utilization and is called dynamic well-organized load balancing (DWOLB) algorithm. The DWOLB mechanism reduces energy consumption in cloud computing. Dam *et al.* [78] proposed a new load balancing strategy to search for underloaded nodes and then balance the load of shocked nodes. Singh et al [79] proposed a load balancing algorithm called Autonomous Agent based mostly Load balancing algorithm (A2LB) which provides dynamic load balancing for cloud environment. Panwar and Mallick [80] proposed a dynamic load management formula to effectively distribute all requests among virtual machines. The algorithm determines the support status of the virtual machines for sharp allocation of requests.

Vasiel et al [81] introduced a resource-aware hybrid scheduling algorithm for different application types: spurt jobs and workflows. The proposed algorithm assumes hierarchical clusters to divide resources into groups and assign some of the tasks to them, which is done in two stages: First, jobs are assigned to resource groups. Second, a classical scheduling algorithm was used for each resource cluster. The algorithm is suitable for heterogeneous distributed computing, especially for modern high computing (HPC) systems where applications with different requirements (both IO and process-intensive) arise, with a focus on data from multimedia applications.

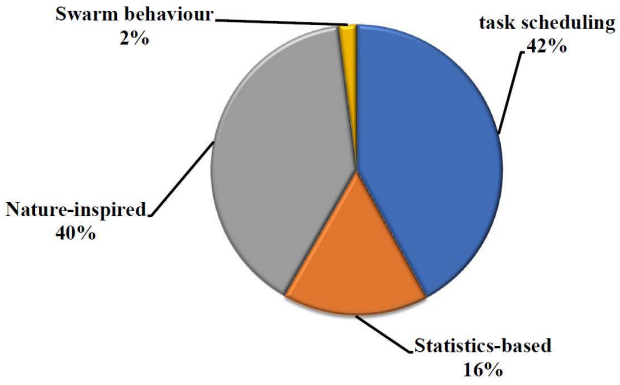


**FIGURE 10.** The most used methods.

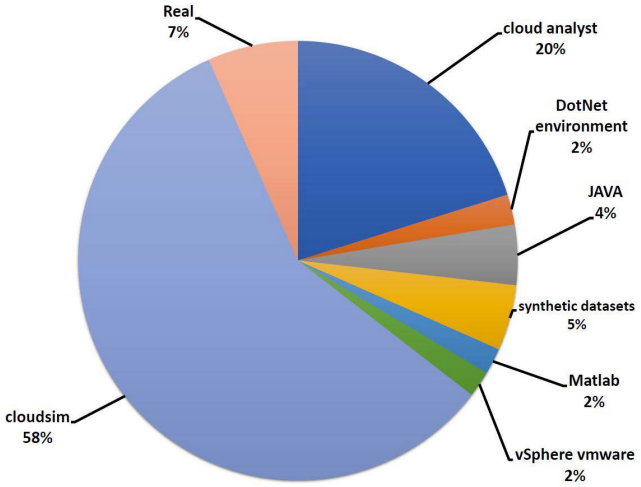


**FIGURE 11.** The most used tools.

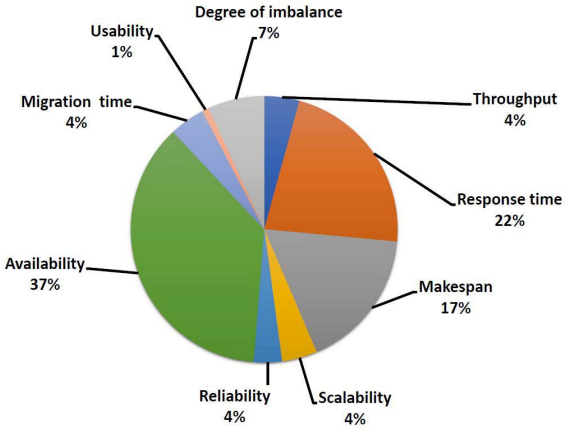


**FIGURE 12.** The most used metrics.

Peng *et al.* [82] studied the Bee Colony Load Balancing Algorithm (BCLB) by proving the mathematical model of the Bee Colony algorithm. After that, it only designs the algorithm implementation process and cloud task scheduling scheme. To show that the BCLB algorithm is better than the other algorithms and the greedy load balancing algorithm. Table 6 shows a comparative analysis for the cloud load balancing framework, and Table 7 shows the evaluation metrics of the framework.

Load balancing is a technique used to achieve and provide availability services on cloud computing. In this review, we find that many researchers are working in this field, whether by improving, proposing, or integrating technology to solve a problem facing load balancing. In this review, researchers working on dynamic load balancing algorithms were covered, i.e., when the requests come and how to deal with them. An analytical comparison was made between all the proposals among researchers by explaining the proposal in general, the type of proposal, its advantages, the technology that was used, The environment in which it was applied, and the metrics used in the evaluation process. It is noticeable that most researchers use task scheduling and Nature-inspired methods in their proposal due to their common use. Figure 10 shows the most used methods in the literature. However, each of them has its own problems based on itself. This is also the tool most used in the application and simulation of the proposal is the cloudsim. Figure 11 shows the most used tools in the literature. Also, the measures used in the evaluation, We find that they target specific measures depending on their proposal. They did not use all the measures that guarantee an assessment of essential performance. Figure 12 shows the most used metrics in the literature.

## V. HYBRID PROACTIVE FAULT TOLERANCE AND DYNAMIC LOAD BALANCING

In this section of the paper, several fault tolerance combined with load balancing algorithms proposed in the literature are surveyed. The frameworks presented in the works are explained in adequate detail concerning the primary load balancing approach, methodology used, various implementation aspects, and metrics used for evaluation with comparative analysis. Sun *et al.* [84] presented a model named QoS-aware scheduling. To ensure the reliability of service in edge-cloud with the time constraints of tasks being satisfied, the model of QoS-aware combined with fault-tolerance in edge-cloud based on standard primary-backup (PB) fault-tolerant technique. To improve the QoS levels of jobs in edge-cloud, a QoS-aware fault-tolerant scheduling algorithms and primary copy allocation, backup copy placement combined with an adjustment mechanism are implemented. The use of primary copy allocation ensures the early execution of the primary copy of a task to satisfy the time necessities of tasks. The backup copy allocation ensures the recent execution of the backup copy of a task. Moreover, reduce the overlapping of the two copies of a task. The adjustment technique is initiated to set up the task replicas associated to a computing node that belongs to the edge cloud when the deallocation of a backup copy on the node, to assist higher the goal-achievement of the first and backup copy scheduling. Xu et al [85] proposed a method for dynamic resource provisioning. The proposed method in [86] was

**TABLE 6.** Comparative analysis of current frameworks.

| Authors | Mechaims | Algorithms | Methods | Experiments | Advantages |
|---|---|---|---|---|---|
| zong et al [40] | model | dynamic | workflow scheduling | cloudsim | Better resource utilization |
| Ragmani et al. [41] | hybrid | dynamic | Nature-inspired | Cloudsim/cloud analyst | Better total cost, response time and processing time |
| tonget al. [42] | hybrid | dynamic | workflow scheduling | cloudsim | lowest makespan Better load-balance |
| Hamdani et al. [43] | Enhancement | dynamic | Nature-inspired | cloud analyst | Better response time |
| Chaudhary and Kumar [44] | hybrid | Dynamic | Nature-inspired | cloudsim | Increase utilization and decrease the cost |
| Kashikolaei et al. [45] | hybrid | dynamic | task scheduling | DotNet environment | Better makespan and stability increasing the efficiency of resources |
| Li et al. [46] | hybrid | dynamic | task scheduling | cloudsim | Reduce cost ,total time and better load balancing |
| Priya et al. [47] | model | dynamic | task scheduling | cloudsim | better performance resource scheduling efficiency response time |
| Gamal et al. [83] | hybrid | dynamic | Nature-inspired | cloudsim | decrease response and execution times and increase system utilization |
| Kumar and Shukla [48] | method | dynamic | Nature-inspired | cloudsim | increase performance and scalability, minimize associated overheads, and avoid bottleneck problem. |
| Khodar et al. [49] | Enhancement | dynamic | Nature-inspired | JAVA | better load distribution reduces dynamic migration. |
| Adhikari and Amgoth [50] | Enhancement | dynamic | task scheduling | synthetic datasets | better resource utilization |
| Padmavathi and Basha [51] | Enhancement | dynamic | Nature-inspired | cloudsim | Better Makespan |
| Acharya [52] | Enhancement | dynamic | Statistics-based | cloudsim | Better performance and utilization of resources |
| Garg et al. [53] | method | dynamic | Statistics-based | CloudAnalyst | enhance system performance and decrease response time |
| Babu and Samuel [54] | Enhancement | dynamic | Swarm behaviourbased | | improvement in the QoS delivered to the customers. Better makespan |
| Jena et al. [55] | hybrid | dynamic | Nature-inspired | cloudsim | decrease the makespan time, increase system throughput and enhance energy utilization |
| Chitgar et al. [56] | method | dynamic | task scheduling | Cloudsim | improves the utilization of the cloud systems |
| Patel and Bhalodia [57] | hybrid | dynamic | Nature-inspired | Cloudsim/real cloud platform | better resource utilization, minimum completion time and improve the system of performance in a cloud |
| Rajput and Kushwah [58] | hybrid | dynamic | Nature-inspired | Cloudsim | decrease the makespan time and enhance resource utilization . |
| ghumman and kaur [59] | hybrid | Dynamic And static | Nature-inspired | Cloudsim | minimizing the total makespan. improve resource utilization and job response time |
| Seddigh and Sharifian [60] | hybrid | dynamic | Nature-inspired | Matlab | lower resource wastage |
| Sun et al. [61] | hybrid | dynamic | Nature-inspired | Cloudsim | better performance,makespan and load balance. |

Continued on next page

**TABLE 6.** *(Continued.)* **Comparative analysis of current frameworks.**

| Author | Mechaims | Algorithm | Methods | Experiments | Advantages |
|---|---|---|---|---|---|
| Ebadifard et al. [62] | new | dynamic | task scheduling | Cloudsim | improves the reliability and degree of imbalance. |
| Mandal et al. [63] | strategy | dynamic | Statistics-based | cloud analyst | Better response time |
| Azmat et al. [64] | Approach | dynamic | task scheduling | cloud analyst | Better response time Reduce processing cost |
| Mardini and Enizat [65] | algorithm | dynamic | Nature-inspired | cloud analyst | better performance |
| Adhikariet al, [66] | mechanism | dynamic | task scheduling | Synthetic dataset | minimum makespan and execution cost |
| Kshama and Shobha [67] | approach | dynamic | Statistics-based | Cloudsim | utilizes the resources efficiently |
| Mohapatra et al. [68] | algorithm | dynamic | Nature-inspired | Cloudsim | improved average response time and the total execution time. |
| Kumar et al. [69] | algorithm | dynamic | task scheduling | Cloudsim | increases utilization of resources |
| Kumar and Sharma [70] | algorithm | dynamic | task scheduling | Cloudsim | reduce the makespan time increase the average resource utilization |
| Kumar and Sharma [71] | algorithm | dynamic | task scheduling | cloudsim | decreases the makespan time and increases the task to meet the deadline |
| Sadia et al. [72] | algorithm | dynamic | task scheduling | heterogeneous virtual machines | increases utilization of resources |
| Kaur and Ghumman [73] | algorithm | dynamic | task scheduling | cloudsim | minimum completion time better resource utilization |
| Cheng et al. [74] | approach | dynamic | task scheduling | Hadoop | improve the average job completion time |
| Kaushik et al. [75] | algorithm | dynamic | task scheduling | Cloudsim/cloud analyst | better resource utilization better response time |
| BEN ALLA et al. [76] | architecture | dynamic | Nature-inspired | Cloudsim | achieve good performance, load balancing, and improve the resource utilization. |
| Vanitha and Marikkannu [77] | method | dynamic | Nature-inspired | vSphere vmware | reducing the energy that is consumed in cloud computing |
| Dam et al. [78] | strategy | dynamic | Statistics-based | cloud analyst | improved the response time |
| Singh et al. [79] | mechanism | dynamic | Statistics-based | JAVA | Improve degree of imbalance and reduces service time. |
| Panwar and Mallick [80] | algorithm | dynamic | Statistics-based | cloud analyst | response time has been improved efficiently |
| vasiel et al. [81] | hybrid | dynamic | task scheduling | CloudSim | Improve scalability and total execution time |
| Peng et al. [82] | Enhacment | dynamic | task scheduling | CloudSim | Improve performance |
| | | | | | |

named DRPM. DRPM planned with fault tolerance for the data-intensive earth science workflows. Technically, the Virtual Layer two (VL2) configuration constructs earth science cloud infrastructure. After that, the non-dominated sorting genetic algorithm II (NSGA-II) reduces the makespan and enhances the load balance. For hybrid tasks over the cloud Han et al [86] introduced a new fault-tolerant scheduling algorithm named ARCHER. This has three vital features: the combined (Primary/Backup) model and the technique of checkpoint to achieve flexibly verify the execution time of the backup replica of tasks. Hence improves resource utilization and also produces longer slots to execute tasks. Secondly, It used the task classification technique to comprehend precise programming for various tasks and virtual machines that

**TABLE 7. Evaluation metrics of frameworks.**

| Author Name | Throughput | Response time | Makespan | Scalability | Reliability | Availability | Migration time | Usability | Degree of imbalance |
|---|---|---|---|---|---|---|---|---|---|
| zong et al [40] | × | × | × | × | × | ✓ | × | × | |
| Ragmani et al. [41] | × | ✓ | × | × | × | ✓ | × | × | × |
| tonget al. [42] | × | × | ✓ | ✓ | × | ✓ | × | × | ✓ |
| Hamdani et al. [43] | × | ✓ | × | × | × | ✓ | × | × | × |
| Chaudhary and Kumar [44] | × | × | × | ✓ | × | ✓ | × | × | × |
| Kashikolaei et al. [45] | × | × | ✓ | × | ✓ | ✓ | × | ✓ | ✓ |
| Li et al. [46] | × | ✓ | × | × | × | ✓ | × | × | × |
| Priya et al. [47] | × | ✓ | × | × | × | ✓ | × | × | × |
| Gamal et al. [83] | × | ✓ | ✓ | × | × | ✓ | × | × | × |
| Kumar and Shukla [48] | × | ✓ | × | ✓ | × | ✓ | × | × | ✓ |
| Khodar et al. [49] | × | ✓ | × | × | × | ✓ | ✓ | × | × |
| Adhikari and Amgoth [50] | ✓ | × | ✓ | × | ✓ | ✓ | × | × | × |
| Padmavathi and Basha [51] | × | × | ✓ | × | × | ✓ | × | × | × |
| Acharya [52] | × | × | × | × | × | ✓ | × | × | × |
| Garg et al. [53] | × | ✓ | × | × | × | ✓ | × | × | × |
| Babu and Samuel [54] | × | × | ✓ | × | × | ✓ | ✓ | × | × |
| Jena et al. [55] | ✓ | ✓ | ✓ | × | × | ✓ | ✓ | × | ✓ |
| Chitgar et al. [56] | × | ✓ | ✓ | × | × | ✓ | × | × | ✓ |
| Patel and Bhalodia [57] | × | ✓ | × | × | × | ✓ | × | × | × |
| Rajput and Kushwah [58] | × | × | × | ✓ | × | ✓ | × | × | × |
| ghumman and kaur [59] | × | ✓ | ✓ | × | × | ✓ | × | × | × |
| Seddigh and Sharifian [60] | × | × | × | × | × | ✓ | × | × | × |
| Sun et al. [61] | × | × | ✓ | × | × | ✓ | × | × | × |
| Ebadifard et al. [62] | × | ✓ | ✓ | × | ✓ | ✓ | × | × | ✓ |
| Mandal et al. [63] | × | ✓ | × | × | × | ✓ | × | × | ✓ |
| Azmat et al. [64] | × | ✓ | × | × | × | ✓ | × | × | × |
| Mardini and Enizat [65] | ✓ | ✓ | × | × | × | ✓ | ✓ | × | × |
| Adhikariet al, [66] | × | ✓ | ✓ | × | ✓ | ✓ | × | × | × |
| Kshama and Shobha [67] | × | ✓ | ✓ | × | × | ✓ | × | × | × |
| Mohapatra et al. [68] | × | ✓ | ✓ | × | × | ✓ | × | × | × |
| Kumar et al. [69] | × | × | × | × | × | ✓ | × | × | × |
| Kumar and Sharma [70] | × | × | ✓ | × | × | ✓ | × | × | × |
| Kumar and Sharma [71] | × | × | ✓ | ✓ | × | ✓ | × | × | × |
| Sadia et al. [72] | ✓ | × | ✓ | × | × | ✓ | × | × | × |
| Kaur and Ghumman [73] | × | ✓ | ✓ | × | × | ✓ | × | × | × |
| Cheng et al. [74] | × | × | ✓ | × | × | ✓ | × | × | × |
| Kaushik et al. [75] | × | ✓ | × | × | × | ✓ | × | × | × |
| BEN ALLA et al. [76] | × | ✓ | × | × | × | ✓ | × | × | ✓ |
| Vanitha and Marikkannu [77] | × | × | × | × | × | ✓ | ✓ | × | × |
| Dam et al. [78] | × | ✓ | × | × | × | ✓ | × | × | × |
| Singh et al. [79] | × | ✓ | × | × | × | ✓ | × | × | × |
| Panwar and Mallick [80] | × | ✓ | × | × | × | ✓ | × | × | × |
| vasiel et al. [81] | × | ✓ | × | ✓ | × | ✓ | × | × | × |
| Peng et al. [82] | × | ✓ | × | × | × | × | × | × | × |

reduce the interval of clouds. Finally, it uses a slot exploiting technique, task forward technique and task remodel technique to ensure resource utilization.

Chinnathambi *et al.* [87] proposed a combined load balancing and fault tolerance algorithm. First, a scheduling algorithm called (WSSS) for load balance by keeping track of server performance and all virtual cluster elements to assign the optimal performing server to the mission-important application. thus arranging the servers based on a counter that monitors each virtual machine for performance failures. Secondly, a checkpoint optimization algorithm called (TCC) for fault tolerance to tolerate and completely remove Byzantine faults. It uses monitoring delay difference to understand the potential Byzantine error-prone region to set up new virtual machine with previous checkpointing. besides that it will stretch the interval of the state for performing arts and error-free virtual machine in control to reduce the area, time, and value overheads caused by checkpointing. Keshk *et al.* [88] introduces the issues related fault tolerance and to load balancing in cloud. Load balancing depicts the mechanism of distributing the load between numerous virtual machines of the cloud to prevent resource wastage when a multiple resources are heavily loaded, although there are equivalent resources that are passive. The fault tolerance is to meet the needs of the provision and reliability resulting from the user's services being processed remotely. There are possibilities of errors and missing management over cloud virtual machines and missing management over cloud virtual machines. Abohamama *et al.* [89] proposed a complete framework that combines different fault tolerance techniques to enhance cloud reliability while achieving reliability. Also, a fault-tolerant period scheduling algorithm was implemented

**TABLE 8.** Comparative analysis of current frameworks.

| Author Name | Type Of Proposed | Technique Type | Method | Application | Advantages | limitations |
|---|---|---|---|---|---|---|
| SUN et al [84] | model | hybrid | QoS task scheduling/primary backup task | Python | Better performance | performance based on: Guarantee Ratio Average QoS Level Reliability Cost |
| Xu et al [85] | method | hybrid | genetic algorithm/job migration | Real | optimal resource provisioning | Makespan Load balance Average resource utilization |
| Han et al [86] | algorithm | hybrid | PB (Primary/Backup) model and Checkpoint/task scheduling | Cloudsim | improve the resource utilization of cloud while guaranteeing fault tolerance. | performance based on: Guarantee Ratio Task Complete Time Host Consume |
| Chinnathambi et al [87] | algorithm | hybrid | Checkpoint/task scheduling | Cloudsim | decreasing the overload and preventing the Byzantine risks completely. The use of scheduling algorithm for selecting the optimal servers to begin backup VNs in case of failures Normal distribution of time | Normal distribution for execution time |
| Keshk et al [88] | algorithm | hybrid | hill-climbing/ AFTRC model | Cloudsim | decreasing the unbalancing between available virtual machines, enhance the performance. | calculate the Makespan Standard deviation for each process |
| Abohamama et al. [89] | framework | hybrid | preemptive migration, checkpoint/restart, and replication. task scheduling | C # | decrease the Harvest Time and Missing Rate and enhance Imbalance Degree | makespan |
| Guo et al. [90] | algorithm | hybrid | PB (Primary/Backup) model and Checkpoint/task scheduling | | good energy efficiency and deadline guarantee ratio. | Performance impact of task count Performance impact of task length Performance impact of task deadline |
| Arabnejad et al. [91] | Famework | hybrid | Proactive fault tolerance/ Fuzzy weighted job distributor | OpenStack | Decrease faults | CPU utilization Failure rate |
| Guo and Xue [92] | Algorithm | hybrid | Primary/backup (P/B)/ Particle swarm optimization | | low cost and high deadline guarantee ratio | Performance impact of task count Performance impact of task length Performance impact of deadline |
| Satpathy et al [93] | Model | hybrid | Queueing model scheduling/ multi-objective virtual machine placement/ migration | CloudSim | Increase system performance and resource utilization. Decrease resources wastage. | Migration and Down Time |
| Tamilvizhi and Parvathavarthini [94] | Framework | hybrid | Proactive and reactive fault tolerance/ elastic cloud balancing (ECB) technique combined with Job Shop Scheduling | JAVA | Low Execution time Less Network congestion Cost Reduced Increased Data availability | Migration Exution time Network congestion Cost |
| Soniya et al. [95] | Mechanism | hybrid | Primary/backup (P/B)/scheduling | Cloudsim | improves the system resource utilization | performance based on: Increase the ratio of Hosts Active Time |

to reduce deadlines, makespan time, and load imbalance. Guo *et al.* [90] proposed a hybrid algorithm based on particle swarm optimization (EFTP) called energy-efficient fault-tolerant static scheduling for period tasks in clouds. For fault tolerance, the primary backup (PB) model is used. And the overlapping backup technique is deployed to reduce the overhead caused by job replication. for load balancing to select compliance resources. The particle swarm optimization (PSO) approach is used for task allocation. Due to the complexity of fault tolerance management for users. The authors Arabnejad *et al.* [91] proposed a fuzzy distribution. There goals to redeuce the likelihood of fault occurrences by distributing user task requests across resources. And the system will manage abnormal events that may cause failure by distributing the incoming task request to advocated by the reliability of process nodes. The reliability of virtual machines may be a variable parameter and changes throughout its period. For cloud cycle jobs, a combination of Particle swarm optimization (PSO) to handle the task assignment and primary/backup (P/B) technique is implemented to achieve fault tolerance for jobs failure during hardware failure. also, rescheduling techniques were proposed to meet the needs cycle job point in time constraints. was proposed by Guo and Xue [92] and called a cost-effective fault-tolerant scheduling algorithm (CEFT) Satpathy *et al.* [93] proposed a queuing structure to manage and schedule a larger number of virtual machines. also, they are implementing a multi-objective virtual machine placement formula that is crow search-based, mostly VM placement (CSAVMP), to reduce resource wastage and power consumption in

cloud data centres. Tamilvizhi and Parvathavarthini [94] proposed work introduces advanced prospective on adopting fault-tolerant techniques for reducing network congestion and fault detection based on migration techniques even when fault happens. Soniya *et al.* [95] proposed dynamic resource allocating techniques include fault tolerance to support resource utilization. They will be disposed to combine an overlapping backup technique and VM migration technique to find a novel dynamic Fault Tolerant proposed technique for real-time jobs in the cloud. Their proposed aims to achieve fault tolerance and optimal resource utilization.

Hybrid cloud fault tolerance and load-balancing frameworks are techniques used to achieve and provide high availability services on cloud computing. This review finds that many researchers are working in this field. In this review, researchers working on reactive fault tolerance with dynamic load balancing algorithm were covered, i.e., when the requests come and how to deal with it and even in failure mode. An analytical comparison was made between all the proposals among researchers by explaining the proposal in general, the type of proposal, its advantages, the technology that was used, the environment in which it was applied, and the metrics used in the evaluation process.

## VI. FUTURE DIRECTIONS AND CONCLUSION

Computational cloud is a technology that transfers the computational process from clients' end nodes to the cloud providers on the internet. Therefore, there are several challenges facing service providers and consumers, such as multiple security services, in-cluding the need for availability. Consequently, it is necessary to use load balancing and fault tolerance techniques. This paper introduced a broad systematic literature review of various reactive fault tolerance, dynamic load balancing, and integrated reactive fault tol-erance with load balancing in their basic approaches, types, proposed research frame-works, comparative analysis, and future direction.

The answers to the research questions were completed based on the comparative analysis between researchers. As a result, we found that reactive fault tolerance and dynamic load balancing can work in an integrated way to achieve high availability and reliability. As the existing frameworks focus only on a few fault types, new reactive fault tolerance frameworks are expected to solve more fault types due to advancements in machine learning methods. Researchers can use deep learning algorithms to detect defects and build integrated models that combine multiple reactive fault tolerance techniques to handle fault detection and resource wastage when faults occur.

Hybrid fault tolerance techniques such as replication if a task has failed at the beginning of execution or checkpointing resume the task from where failure had happened. Furthermore, migration methods can be integrated to move the job from one node with failure to another. Most of the approaches proposed in the literature for cloud fault-tolerance focus on reactive fault tolerance techniques. Cloud providers comprise thousands of physical nodes with different reliabilities, resulting in a complex way to predict and discover faults and failures in large networks. Therefore, cloud reactive fault tolerance researchers recommend frameworks that provide complete reactive fault tolerance with prediction, detection, prevention, and recovery.

On the other hand, the existing dynamic load balancing frameworks do not focus on load imbalance between multiple nodes, leading to a denial of service. Instead, using deep learning algorithms in case of an imbalance of nodes achieves good load balance and better performance. As hybrid approaches of reactive fault tolerance and dynamic load balancing, reactive fault tolerance generally work in hardware infrastructure. However, hardware infrastructure failure can occur much more than software infrastructure failure. Unbalanced load allocation can be a fundamental reason that causes hosts to overload and, accordingly, the fault. Reactive fault tolerance can be accomplished across efficient load distribution based on dynamic load balancing and clustering. Clustering is a procedure implemented to shape groups of similar entities. A cluster of jobs and nodes is applied in a distributed network to achieve efficient job-node mapping.

## ACKNOWLEDGMENT

The authors extend their appreciation to the deputyship for research and innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (NU/IFC/ENT/01/013) under the institutional funding committee at Najran University, Kingdom of Saudi Arabia.

## REFERENCES

[1] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud Computing: Principles and Paradigms*. Hoboken, NJ, USA: Wiley, 2010.
[2] R. Kaur and S. Kinger, "Analysis of job scheduling algorithms in cloud computing," *Int. J. Comput. Trends Technol.*, vol. 9, no. 7, pp. 379–386, Mar. 2014.
[3] V. N. Volkova, L. V. Chemenkaya, E. N. Desyatirikova, M. Hajali, A. Khodar, and A. Osama, "Load balancing in cloud computing," in *Proc. IEEE Conf. Russian Young Res. Electr. Electron. Eng. (EIConRus)*. Jan./Feb. 2018, pp. 387–390.
[4] S. Bansal, S. Sharma, and I. Trivedi, "A detailed review of fault-tolerance techniques in distributed system," *Int. J. Internet Distrib. Comput. Syst.*, vol. 1, no. 1, pp. 1–7, 2011.
[5] V. A. D. Santos, A. Manacero, R. S. Lobato, R. Spolon, and M. A. Cavenaghi, "A systematic review of fault tolerance solutions for communication errors in open source cloud computing," in *Proc. 15th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2020, pp. 1–6.
[6] P. Marcotte, F. Grégoire, and F. Petrillo, "Multiple fault-tolerance mechanisms in cloud systems: A systematic review," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2019, pp. 414–421.
[7] M. Hasan and M. S. Goraya, "Fault tolerance in cloud computing environment: A systematic survey," *Comput. Ind.*, vol. 99, pp. 156–172, Aug. 2018.
[8] 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.
[9] Z. Chaczko, V. Mahadevan, S. Aslanzadeh, and C. Mcdermid, "Availability and load balancing in cloud computing," in *Proc. Int. Conf. Comput. Softw. Modeling*, vol. 14. Singapore: IACSIT Press, 2011, pp. 134–140.
[10] M. H. Alshayeji, M. Al-Rousan, E. Yossef, and H. Ellethy, "A study on fault tolerance mechanisms in cloud computing," *Int. J. Comput. Electr. Eng.*, vol. 10, no. 1, pp. 62–71, 2018.

[11] T. Mohmmed and N. Abdalrahman, "A load balancing with fault tolerance algorithm for cloud computing," in *Proc. Int. Conf. Comput., Control, Electr., Electron. Eng. (ICCCEEE)*, Feb. 2021, pp. 1–6.

[12] M. Dhingra and N. Gupta, "Comparative analysis of fault tolerance models and their challenges in cloud computing," *Int. J. Eng. Technol.*, vol. 6, no. 2, pp. 36–40, 2017.

[13] A. Bala and I. Chana, "Fault tolerance-challenges, techniques and implementation in cloud computing," *Int. J. Comput. Sci. Issues (IJCSI)*, vol. 9, no. 1, p. 288, 2012.

[14] D. P. Chandrashekar, "Robust and fault-tolerant scheduling for scientific workflows in cloud computing environments," Ph.D. dissertation, Univ. Melbourne, Parkville, VIC, Australia, 2015.

[15] W. Zhao, P. M. Melliar-Smith, and L. E. Moser, "Fault tolerance middleware for cloud computing," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, Jul. 2010, pp. 67–74.

[16] S. Malik and F. Huet, "Adaptive fault tolerance in real time cloud computing," in *Proc. IEEE World Congr. Services*, Jul. 2011, pp. 280–287.

[17] C. O. Kumar and P. R. K. S. Bhama, "Fuzzy based energy efficient workload management system for flash crowd," *Comput. Commun.*, vol. 147, pp. 225–234, Nov. 2019.

[18] 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. Hum. Comput.*, vol. 10, no. 11, pp. 4567–4577, Nov. 2019.

[19] V. M. Sivagami and K. S. Easwarakumar, "An improved dynamic fault tolerant management algorithm during VM migration in cloud data center," *Future Gener. Comput. Syst.*, vol. 98, pp. 35–43, Sep. 2019.

[20] B. Mohammed, M. Kiran, I.-U. Awan, and K. M. Maiyama, "Optimising fault tolerance in real-time cloud computing IaaS environment," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2016, pp. 363–370.

[21] G. Yao, Q. Ren, X. Li, S. Zhao, and R. Ruiz, "A hybrid fault-tolerant scheduling for deadline-constrained tasks in cloud systems," *IEEE Trans. Services Comput.*, vol. 15, no. 3, pp. 1371–1384, May 2022.

[22] E. AbdElfattah, M. Elkawkagy, and A. El-Sisi, "A reactive fault tolerance approach for cloud computing," in *Proc. 13th Int. Comput. Eng. Conf. (ICENCO)*, Dec. 2017, pp. 190–194.

[23] M. Amoon, "Adaptive framework for reliable cloud computing environment," *IEEE Access*, vol. 4, pp. 9469–9478, 2016.

[24] M. Hasan and M. S. Goraya, "Flexible fault tolerance in cloud through replicated cooperative resource group," *Comput. Commun.*, vol. 145, pp. 176–192, Sep. 2019.

[25] M. K. Beheshti and F. Safi-Esfahani, "BFPF-cloud: Applying SVM for Byzantine failure prediction to increase availability and failure tolerance in cloud computing," *Social Netw. Comput. Sci.*, vol. 1, no. 5, pp. 1–31, Sep. 2020.

[26] M. Alaei, R. Khorsand, and M. Ramezanpour, "An adaptive fault detector strategy for scientific workflow scheduling based on improved differential evolution algorithm in cloud," *Appl. Soft Comput.*, vol. 99, Feb. 2021, Art. no. 106895.

[27] B. Mohammed, M. Kiran, I.-U. Awan, and K. M. Maiyama, "An integrated virtualized strategy for fault tolerance in cloud computing environment," in *Proc. Int. IEEE Conf. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr. (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 542–549.

[28] A. B. Singh, S. Bhat, R. Raju, and R. D'Souza, "Survey on various load balancing techniques in cloud computing," *Adv. Comput.*, vol. 7, no. 2, pp. 28–34, 2017.

[29] 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, p. 64, 2016.

[30] M. Xu, W. Tian, and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," *Concurrency Comput., Pract. Exp.*, vol. 29, no. 12, Jun. 2017, Art. no. e4123.

[31] 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.

[32] N. Pasha, A. Agarwal, and R. Rastogi, "Round Robin approach for VM load balancing algorithm in cloud computing environment," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 4, no. 5, pp. 34–39, 2014.

[33] T. Kokilavani and D. G. Amalarethinam, "Load balanced min-min algorithm for static meta-task scheduling in grid computing," *Int. J. Comput. Appl.*, vol. 20, no. 2, pp. 43–49, 2011.

[34] O. M. Elzeki, M. Z. Reshad, and M. A. Elsoud, "Improved max-min algorithm in cloud computing," *Int. J. Comput. Appl.*, vol. 50, no. 12, pp. 22–27, Jul. 2012.

[35] J. M. Galloway, K. L. Smith, and S. S. Vrbsky, "Power aware load balancing for cloud computing," in *Proc. World Congr. Eng. Comput. Sci.*, vol. 1, 2011, pp. 19–21.

[36] S. Sethi, A. Sahu, and S. K. Jena, "Efficient load balancing in cloud computing using fuzzy logic," *IOSR J. Eng.*, vol. 2, no. 7, pp. 65–71, Jul. 2012.

[37] W. Hashem, H. Nashaat, and R. Rizk, "Honey bee based load balancing in cloud computing," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 12, pp. 5694–5711, 2017.

[38] N. Er-raji and F. Benabbou, "Priority task scheduling strategy for heterogeneous multi-datacenters in cloud computing," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 2, pp. 272–277, 2017.

[39] 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*. Singapore: Springer, 2018, pp. 181–195.

[40] J. H. Zhong, D. L. Cui, Z. P. Peng, Q. R. Li, and J. G. He, "Multi workflow fair scheduling scheme research based on reinforcement learning," *Proc. Comput. Sci.*, vol. 154, pp. 117–123, Jan. 2019, doi: 10.1016/J.PROCS.2019.06.018.

[41] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, and M. Rida, "FACO: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing," *J. Ambient Intell. Hum. Comput.*, vol. 11, no. 10, pp. 3975–3987, Oct. 2020.

[42] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep *Q*-learning," *Inf. Sci.*, vol. 512, pp. 1170–1191, Feb. 2020.

[43] M. Hamdani, Y. Aklouf, and H. A. Bouarara, "Improved fuzzy load-balancing algorithm for cloud computing system," in *Proc. 9th Int. Conf. Inf. Syst. Technol.*, Mar. 2019, pp. 1–4.

[44] D. Chaudhary and B. Kumar, "Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing," *Appl. Soft Comput.*, vol. 83, Oct. 2019, Art. no. 105627.

[45] S. M. G. Kashikolaei, A. A. R. Hosseinabadi, B. Saemi, M. B. Shareh, A. K. Sangaiah, and G. B. Bian, "An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm," *J. Supercomput.*, vol. 76, no. 8, pp. 6302–6329, 2020.

[46] Y. Li, S. Wang, X. Hong, and Y. Li, "Multi-objective task scheduling optimization in cloud computing based on genetic algorithm and differential evolution algorithm," in *Proc. 37th Chin. Control Conf. (CCC)*, Jul. 2018, pp. 4489–4494.

[47] V. Priya, C. S. Kumar, and R. Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning," *Appl. Soft Comput.*, vol. 76, pp. 416–424, Mar. 2019.

[48] N. Kumar and D. Shukla, "Load balancing mechanism using fuzzy row penalty method in cloud computing environment," in *Information and Communication Technology for Sustainable Development*. Singapore: Springer, 2018, pp. 365–373.

[49] A. Khodar, H. A. F. Al-Afare, and I. Alkhayat, "New scheduling approach for virtual machine resources in cloud computing based on genetic algorithm," in *Proc. Int. Russian Autom. Conf. (RusAutoCon)*, Sep. 2019, pp. 1–5.

[50] M. Adhikari and T. Amgoth, "An enhanced dynamic load balancing mechanism for task deployment in IaaS cloud," in *Proc. Int. Conf. Comput., Power Commun. Technol. (GUCON)*, Sep. 2018, pp. 451–456.

[51] M. Padmavathi and S. M. Basha, "Dynamic and elasticity ACO load balancing algorithm for cloud computing," in *Proc. Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, Jun. 2017, pp. 77–81.

[52] S. Acharya and D. A. D'Mello, "Enhanced dynamic load balancing algorithm for resource provisioning in cloud," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Aug. 2016, pp. 1–5.

[53] S. Garg, D. D. V. Gupta, and R. K. Dwivedi, "Enhanced active monitoring load balancing algorithm for virtual machines in cloud computing," in *Proc. Int. Conf. Syst. Modeling Adv. Res. Trends (SMART)*, 2016, pp. 339–344.

[54] K. R. Babu and P. Samuel, "Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud," in *Innovations in Bio-Inspired Computing and Applications*. Cham, Switzerland: Springer, 2016, pp. 67–78.

[55] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *J. King Saud Univ., Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2332–2342, Jun. 2022.

[56] N. Chitgar, H. Jazayeriy, and M. Rabiei, "DSCTS: Dynamic stochastic cloud task scheduling," in *Proc. 5th Iranian Conf. Signal Process. Intell. Syst. (ICSPIS)*, Dec. 2019, pp. 1–5.
[57] K. D. Patel and T. M. Bhalodia, "An efficient dynamic load balancing algorithm for virtual machine in cloud computing," in *Proc. Int. Conf. Intell. Comput. Control Syst. (ICCS)*, May 2019, pp. 145–150.
[58] S. S. Rajput and V. S. Kushwah, "A genetic based improved load balanced min-min task scheduling algorithm for load balancing in cloud computing," in *Proc. 8th Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Dec. 2016, pp. 677–681.
[59] N. S. Ghumman and R. Kaur, "Dynamic combination of improved max-min and ant colony algorithm for load balancing in cloud system," in *Proc. 6th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2015, pp. 1–5.
[60] M. Seddigh, H. Taheri, and S. Sharifian, "Dynamic prediction scheduling for virtual machine placement via ant colony optimization," in *Proc. Signal Process. Intell. Syst. Conf. (SPIS)*, Dec. 2015, pp. 104–108.
[61] W. Sun, Z. Ji, J. Sun, N. Zhang, and Y. Hu, "SAACO: A self adaptive ant colony optimization in cloud computing," in *Proc. IEEE 5th Int. Conf. Big Data Cloud Comput.*, Aug. 2015, pp. 148–153.
[62] 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.
[63] A. Sarkar, K. Pant, and S. Chattopadhyay, "DRSQ—A dynamic resource service quality based load balancing algorithm," in *Proc. Int. Conf. Comput. Intell., Commun., Bus. Anal.* Singapore: Springer, 2018, pp. 97–108.
[64] T. Azmat, D. Kumar, and V. K. Dwivedi, "A novel approach towards an efficient load balancing algorithm in cloud computing," in *Proc. 4th Int. Conf. Inf. Syst. Comput. Netw. (ISCON)*, Nov. 2019, pp. 572–577.
[65] H. Alazzam, A. Alsmady, W. Mardini, and A. Enizat, "Load balancing in cloud computing using water flow-like algorithm," in *Proc. 2nd Int. Conf. Data Sci., E-Learn. Inf. Syst.*, 2019, pp. 1–6.
[66] 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.
[67] S. B. Kshama and K. R. Shobha, "A novel load balancing algorithm based on the capacity of the virtual machines," in *Proc. Int. Conf. Adv. Comput. Data Sci.* Singapore: Springer, 2018, pp. 185–195.
[68] S. Mohapatra, I. Aryendu, A. Panda, and A. K. Padhi, "A modern approach for load balancing using forest optimization algorithm," in *Proc. 2nd Int. Conf. Comput. Methodol. Commun. (ICCMC)*, Feb. 2018, pp. 85–90.
[69] M. Kumar, K. Dubey, and S. C. Sharma, "Job scheduling algorithm in cloud environment considering the priority and cost of job," in *Proc. 6th Int. Conf. Soft Comput. Problem Solving*. India: Springer, 2017, pp. 313–320.
[70] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," *Proc. Comput. Sci.*, vol. 115, pp. 322–329, Jan. 2017.
[71] M. Kumar and S. C. Sharma, "Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment," *Comput. Electr. Eng.*, vol. 69, pp. 395–411, Jul. 2018.
[72] F. Sadia, N. Jahan, L. Rawshan, M. T. Jeba, and T. Bhuiyan, "A priority based dynamic resource mapping algorithm for load balancing in cloud," in *Proc. 4th Int. Conf. Adv. Electr. Eng. (ICAEE)*, Sep. 2017, pp. 176–180.
[73] R. Kaur and N. S. Ghumman, "A load balancing algorithm based on processing capacities of VMs in cloud computing," in *Big Data Analytics*. Singapore: Springer, 2018, pp. 63–69.
[74] D. Cheng, J. Rao, Y. Guo, C. Jiang, and X. Zhou, "Improving performance of heterogeneous mapreduce clusters with adaptive task tuning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 774–786, Mar. 2017.
[75] Y. Kaushik, A. Bhola, and C. K. Jha, "Proposed SKYMAX load balancing algorithm," in *Proc. Int. Conf. Adv. Inf. Commun. Technol. Comput.*, 2016, pp. 1–5.
[76] H. B. Alla, S. Ben Alla, and A. Ezzati, "A novel architecture for task scheduling based on dynamic queues and particle swarm optimization in cloud computing," in *Proc. 2nd Int. Conf. Cloud Comput. Technol. Appl. (CloudTech)*, May 2016, pp. 108–114.
[77] M. Vanitha and P. Marikkannu, "Effective resource utilization in cloud environment through a dynamic well-organized load balancing algorithm for virtual machines," *Comput. Electr. Eng.*, vol. 57, pp. 199–208, Jan. 2017.
[78] S. Dam, G. Mandal, K. Dasgupta, and P. Dutta, "Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing," in *Proc. 3rd Int. Conf. Comput., Commun., Control Inf. Technol. (C3IT)*, Feb. 2015, pp. 1–7.
[79] A. Singh, D. Juneja, and M. Malhotra, "Autonomous agent based load balancing algorithm in cloud computing," *Proc. Comput. Sci.*, vol. 45, pp. 832–841, Jan. 2015.
[80] R. Panwar and B. Mallick, "Load balancing in cloud computing using dynamic load management algorithm," in *Proc. Int. Conf. Green Comput. Internet Things (ICGCIoT)*, Oct. 2015, pp. 773–778.
[81] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, and J. Kołodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing," *Future Gener. Comput. Syst.*, vol. 51, pp. 61–71, Oct. 2015.
[82] H. Peng, W. Han, J. Yao, and C. Fu, "The realization of load balancing algorithm in cloud computing," in *Proc. 2nd Int. Conf. Comput. Sci. Appl. Eng. (CSAE)*, 2018, pp. 1–5.
[83] A. S. Nassar, A. H. Montasser, and N. Abdelbaki, "A survey on smart cities' IoT," in *Proc. Int. Conf. Adv. Intell. Syst. Inform.* Cham, Switzerland: Springer, 2017, pp. 855–864.
[84] H. Sun, H. Yu, G. Fan, and L. Chen, "QoS-aware task placement with fault-tolerance in the edge-cloud," *IEEE Access*, vol. 8, pp. 77987–78003, 2020.
[85] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6172–6181, Sep. 2020.
[86] H. Han, W. Bao, X. Zhu, X. Feng, and W. Zhou, "Fault-tolerant scheduling for hybrid real-time tasks based on CPB model in cloud," *IEEE Access*, vol. 6, pp. 18616–18629, 2018.
[87] S. Chinnathambi, A. Santhanam, J. Rajarathinam, and M. Senthilkumar, "Scheduling and checkpointing optimization algorithm for Byzantine fault tolerance in cloud clusters," *Cluster Comput.*, vol. 22, no. S6, pp. 14637–14650, Nov. 2019.
[88] A. E. Keshk, R. A. I. Alsini, and M. A. Tawfeek, "Adaptive fault tolerance for online tasks scheduling in cloud computing," in *Proc. 1st Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, Apr. 2018, pp. 1–6.
[89] A. S. Abohamama, M. F. Alrahmawy, and M. A. Elsoud, "Improving the dependability of cloud environment for hosting real time applications," *Ain Shams Eng. J.*, vol. 9, no. 4, pp. 3335–3346, Dec. 2018.
[90] P. Guo, M. Liu, and Z. Xue, "A PSO-based energy-efficient fault-tolerant static scheduling algorithm for real-time tasks in clouds," in *Proc. IEEE 4th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2018, pp. 2537–2541.
[91] H. Arabnejad, C. Pahl, G. Estrada, A. Samir, and F. Fowley, "A fuzzy load balancer for adaptive fault tolerance management in cloud platforms," in *Proc. Eur. Conf. Service-Oriented Cloud Comput.* Cham, Switzerland: Springer, 2017, pp. 109–124.
[92] P. Guo and Z. Xue, "Cost-effective fault-tolerant scheduling algorithm for real-time tasks in cloud systems," in *Proc. IEEE 17th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2017, pp. 1942–1946.
[93] A. Satpathy, S. K. Addya, A. K. Turuk, B. Majhi, and G. Sahoo, "Crow search based virtual machine placement strategy in cloud data centers with live migration," *Comput. Electr. Eng.*, vol. 69, pp. 334–350, Jul. 2018.
[94] 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.
[95] J. Soniya, J. A. J. Sujana, and T. Revathi, "Dynamic fault tolerant scheduling mechanism for real time tasks in cloud computing," in *Proc. Int. Conf. Electr., Electron., Optim. Techn. (ICEEOT)*, Mar. 2016, pp. 124–129.

**TAWFEEG MOHMMED TAWFEEG** received the bachelor's degree in information for communication and technology and the master's degree in information security from the University of Science and Technology, Sudan, in 2016 and 2018, respectively. He is currently working as a Lecturer at the University of Science and Technology. His research interests include cloud computing security, network security, and data science.

**ADIL YOUSIF** received the B.Sc. and M.Sc. degrees from the University of Khartoum, Sudan, and the Ph.D. degree from the University of Technology in Malaysia (UTM). He is currently an Associate Professor at the College of Arts and Sciences Sharourah, Najran University, Saudi Arabia. He is also the principal investigator of several research projects in artificial intelligence and emerging technologies. His research interests include computer networks, cloud computing, artificial intelligence, and optimization techniques.

**ALZUBAIR HASSAN** received the B.Sc. degree in computer science from the University of Kassala, in 2010, the M.Sc. degree in mathematical science from the University of Khartoum, in 2013, and the Ph.D. degree in computer science and technology from the University of Electronic Science and Technology of China, in 2018. He was a Postdoctoral Researcher at the School of Computer Science and Cyber Engineering, Guangzhou University. Furthermore, he was a Research Scientist at the School of Computer Science, University College Dublin. He was also a Researcher with the Lero-the Irish Software Research Centre. He is currently an Assistant Professor at the School of Computer Science, University College Dublin. His research interests include cryptography, network security, privacy-preserving in machine learning, and adaptive security.

**SAMAR M. ALQHTANI** received the Ph.D. degree in information technology from the University of Newcastle, Australia. She is currently an Assistant Professor at Najran University, Saudi Arabia. She has lectured and developed curricula for computer science and information system courses. She has recently led and worked on various projects, including event detection applications in social media, medical applications, and applying artificial intelligence and machine learning algorithms to emerging technologies. Her research interests include information technology and multimedia, including artificial intelligence, machine learning, deep learning, health informatics, data mining, image processing, computer vision, text processing, and the IoT.

**RAFIK HAMZA** received the M.Sc. and Ph.D. degrees in cryptography and security from Batna 2 University, Algeria, in 2014 and 2017, respectively. In 2018, he joined DC Research and Development Sonatrach, where he worked as a Principal Engineer focused on data reliability and AI-based prediction for industrial systems. In March 2019, he moved to Guangzhou University, where he worked as a Postdoctoral Researcher. His research activities focused on developing secure and efficient machine learning solutions for industry 4.0 applications while maintaining data privacy. In February 2020, he started working as a Researcher at the Big Data Integration Research Center, NICT, Tokyo. The NICT team focuses on developing data collection and analysis technologies that aim to leverage real-world information for better social life and enable cross-domain combinations (https://www.xdata.nict.jp). In April 2022, he became an associate professor at Institute for International Strategy, Tokyo International University. His research interests include privacy-preserving machine learning for real-world industrial technologies and applied cryptography for big data applications.

**MOHAMMED BAKRI BASHIR** received the B.Sc. degree from SUST University, Sudan, and the Ph.D. degree from the University of Technology Malaysia (UTM), Malaysia, in 2014. He is currently an Associate Professor at the Faculty of Computer Science and Information Technology, Shendi University, Sudan. He is also an Associate Professor at Turubah University College, Taif University, Saudi Arabia. His research interests include computer network and data processing, grid computing, distributed computing, NoSQL databases, and AI applications.

**AWAD ALI** received the Ph.D. degree in computer science from the University of Technology Malaysia (UTM), in 2016. He is currently an Assistant Professor with the Computer Science Department, Najran University. His research interests include component-based systems, software quality, software modeling and measurement, and machine learning techniques.