Sustainable Energy Technologies and Assessments 53 (2022) 102613

Contents lists available at ScienceDirect

Sustainable Energy Technologies and Assessments

journal homepage: www.elsevier.com/locate/seta

Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy

Salil Bharany [a], Sumit Badotra [b], Sandeep Sharma [a], Shalli Rani [c], Mamoun Alazab [d], Rutvij H. Jhaveri [e], Thippa Reddy Gadekallu [f,*]

[a] Department of Computer Engineering & Technology, Guru Nanak Dev University, Amritsar, India
[b] Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab, India
[c] Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India
[d] College of Engineering, IT and Environment at Charles Darwin University, Australia
[e] Department of Computer Science and Engineering, School of Technology, Pandit Deendayal Energy University, India
[f] School of Information Technology and Engineering Vellore Institute of Technology, India

ARTICLE INFO

Keywords:
Cloud computing
Energy efficiency
Fault-tolerance
Carbon emission
Virtualization

ABSTRACT

Cloud computing has brought the accessibility of several software platforms under a single roof. It has transformed resources into scalable services on demand and provides the only solution to the high resource requirements. All cloud service providers usually offer all types of services in the cloud computing environment, even though they also handle security-related challenges like reliability, availability, and throughput. One of the most decisive challenges in the cloud is handling faults. High fault tolerance in the cloud is a must to attain high performance, and the defects must be investigated and examined for future guidance. The principal target of this paper is to gain insight into the fault tolerance techniques that are available to us and the challenges that are required to be overcome. We concluded that there is always a relation between faults and energy consumption during our survey. If there is a high potential to tolerate a fault, there will be a need for more infrastructure and devices to fix those faults, which further leads to more power consumption. In this paper, 129 Research papers published through February 2022 were considered and further classified. This paper critically reviews techniques to tolerate faults in cloud computing systems and discusses the taxonomy of errors, faults, and failures. Furthermore, this paper aims to investigate several critical research topics and advanced techniques, such as artificial intelligence, deep learning, the Internet of Things, and machine learning, that may be employed as an intelligent fault tolerance strategy in the cloud environment.

Introduction

The Cloud computing model offers a crucial research area and transforms the IT industry while changing the scenario for purchasing hardware and software such as networks, servers, and storage. Nowadays, data can also be accessed using cloud computing applications anywhere globally. Cloud virtualization allows users to access assets they user does not even own. Fault tolerance is an approach that ensures the dependability and availability of resource applications for fault tolerance. Different fault tolerance methods are deployed to limit the faults in the framework and while application execution [1]. The essential instrument for fault tolerance is replication, repetition, and checkpointing. After hybridization, research suggests that trends are towards the federated Cloud as seamless resources of Cloud are vanishing fast due to growing demands of users. The hybridization of the Cloud is compulsory while keeping in mind that the end goal is to give a practical, elite, and reliable solution for resource accessibility to the clients. This combination of clouds forms a federated cloud that allows public and private resources to scale up their resource pool [2]. However, the problem is that cost is greatly enhanced by making such enormous resources up at all times. A federation of Cloud is formed only at peak times to tackle the situation. Otherwise, topology breakage increases the power and cost factors associated with resources.

Moreover, it will ensure practically infinite resources, computing power, storage, and ease of access. Thus, the prime reason for shifting pools at peak times from Cloud (single provider) to federated Cloud

* Corresponding author.
E-mail address: thippareddy@ieee.org (T. Reddy Gadekallu).

https://doi.org/10.1016/j.seta.2022.102613
Received 7 March 2022; Received in revised form 25 July 2022; Accepted 28 July 2022

(multiple providers) is that the limited resources could cause starvation [3] which further leads to a deadlock situation. Companies with innovative ideas no longer require investing in expensive hardware and software; instead, they can use the facilities provided through Cloud to build the desired application. By using Cloud computing PaaS, users can register critical and necessary services and use that resource without setting up any hardware and software. This feature allows businesses to grow since setting up costs is erased, and core components can be focused upon for the organization's growth. Clouds can be analyzed according to various views [4]. According to the computational view, Cloud is nothing without data centers as one or more data centers collaborated to form Cloud. It is considered a low-cost, efficient, robust, fault-tolerant, and energy-efficient mechanism in future computing. According to a sociological point of view, cloud resources are available only when required [1]. In general, Cloud can accommodate infinite computing resources on-demand to the user. With the advancement of cloud technologies, there is no need to build the infrastructure or set up costs as high-end resources are required for the user on a pay-per-use basis for a short period of demand [2]. Performance and energy consumption have a strong relationship as performance degrades mostly when increasing in fault and failure occurrence.

The following is the structure of our systematic survey: Section 2: a discussion on the previous surveys that have been conducted in this domain. Section 3 discusses the survey technique and the study's overall plan. Section 4 provides background information on cloud computing and fault tolerance (FT). A detailed discussion of FT, including a taxonomy of faults, error & failure, cloud fault-tolerance model, problems of fault tolerance, and the implications of FT in cloud computing, are covered in Section 5. Section 6: Classification of faults can be seen, and Section 7 describes the failure types. Fault tolerance schemes and techniques are relegated under section 8. Section 9: This section goes into great detail on FT techniques and classifies them in the cloud. Section 10: Demonstrate and Evaluate several Frameworks for FT in the cloud, and also compare various FT frameworks grounded on specific criteria. Section 11: Discussion about the analysis can be seen. Section 12: Future Research Directions. Section 13 of this review document ends up as a conclusion.

Need of fault tolerance

FT is the power of a system to preserve the execution of its expected functions despite any failure. In other terms, FT is associated with dependability, operational success, and failure. The FT-based system should investigate the defects in individual software or hardware components. Fault tolerance ensures that the system will provide services in one or many failures of the infrastructural components. Fault tolerance is related to the availability and reliability of system resources that are not affected by the event when any previous component or executable fails. Tolerance for most faults is implemented as a crucial system with high availability. However, providing a fault-tolerant design [5] is not practical for each component. The associated redundancy and over-provisioning bring several parasitic penalties like growth in weight, price, power, size, consumption, and time to create, validate, and test the service before it is delivered. Typically, a system, device, or other component is over-provisioned or under-provisioned to ensure that, even if application performance is impacted during downtime, the system may continue to run at a low level rather than falling below evident and acceptable limits.

Motivation for conducting the survey

In the cloud computing environment, various faults may contribute to failures that further degrade the performance of a system. Failures lead to system collapse or shutdown of a system, and sometimes, defects lead to performance eroding instead of a complete shutdown. Different faults can happen in cloud computing. Grounded on the type of fault like Network faults, Physical faults, and process faults, different fault tolerance strategies are available to tackle them. Fault tolerance cannot be accomplished without understanding the occurrence of the fault within the system and the harm caused by the fault to the system. Cloud consists of layers, and each layer grabs the services from the layer beneath it. Faults at any individual layer can corrupt the layer directly above it since services provided by the lower layer could be impacted due to faults. Therefore, errors should be effectively addressed by the correct fault tolerance mechanism for high-performance computers.

The motivation behind our research can be said in broader terms as follows:

- Requirements for comprehending existing Energy-Efficient Fault tolerance techniques in a cloud computing environment.
- The role of the fault tolerance technique in amending the performance of a system has been discussed. The necessity of a fault tolerance technique, its merits, and its demerits has also been studied.
- The Issues related to cloud computing are referred to explicitly with fault tolerance techniques and evaluation of the best possible fault tolerance strategies used in future endeavors.

Article origination

This paper discusses the categorization of faults, the recognition of fault tolerance techniques, methodologies, the specification of fault tolerance frameworks, and the future research direction in this domain. The survey approach has broken down into five stages:

Stage 1 (Article Selection): During the first step, reliable sources had consulted for a significant count of research publications (including surveys). The papers were vetted and analyzed grounded on their titles, abstracts, and research contributions. The research contributions are evaluated for their novelty and quality. Including articles on faults and tolerance methodology ensures that the reader learns every FT technique's essential experimentation and potential modifications/customizations.

Stage 2 (Cloud Fault Classifications): In the second stage, the gathered papers have been thoroughly analyzed to discover a variety of cloud faults. It takes much effort to classify the defects that have been detected. This phase also provides a quick overview of various failure categories, causes, and classifications.

Stage 3 (Cloud Fault-Tolerance Methodologies): The gathered papers are further analyzed in this phase to find various cloud fault tolerance approaches. Stage 3 of the survey lists and describes the identified methods. This phase also presents fault tolerance strategies based on the recognized methodologies in a hierarchical order.

Stage 4 (Cloud Fault Tolerance Frameworks): An overview of several fault tolerance models in the literature is provided in this survey phase. The objective is to build an ever-expanding knowledge base that encompasses the research contributions of each fault-tolerance technique. Fault applicability, assessment methodology, and well-known fault tolerance frameworks are addressed in this subsection.

Stage 5 (Results, Discussion, and Recommendations): The studied fault tolerance frameworks have been carefully examined in the final step. This phase also includes the study's objectives and survey findings. We quantify how often a specific fault category is addressed in the study and how much a specific fault-tolerance approach has been employed in the research. Based on current research challenges, prospective research pathways for cloud fault tolerance were indicated.

Our contribution

In this Survey, a complete study of the fault tolerance in cloud computing systems was done, and also this Survey discussed the taxonomy of errors, faults, failure, and their causes. Different system-centric parameters are also considered, including fault tolerance

degree, overhead, execution time, etc. Our principal contributions to the research are:

- To better understand the energy-efficient Fault Tolerance approach, we have explained a cloud system model that combines a data center, cloud, and service models.
- A comprehensive cloud Fault taxonomy and failure classification are posed.
- Fault tolerance and overall energy consumption in cloud environments are explained using an analytical study.
- Based on the type of cloud system fault, several energy-efficient approaches are compared.
- The analysis of the literature reveals specific research gaps that were previously ignored.

Related surveys

Although substantial studies in cloud fault tolerance have been done, few surveys [2–10] have been brought out. While valuable to the discipline, these surveys are not comprehensive. These surveys seem to limit their focus to one or two explanations only. Cheraghlou et al. [3] handed a general review of fault tolerance approaches, ignoring failure types. The survey also discusses a few frameworks, limiting its reach [34]. This survey does not classify any fault-tolerance models in the cloud. Agarwal and Sharma [4] presented faults, errors, and failures in taxonomies but no explanations. The authors did not include any current fault-tolerance frameworks in the survey to improve the discussion. In their survey, Atallah et al. [6] described numerous fault tolerance criteria but not defect types. These few frameworks are deficient in defining the state-of-the-art. Saikia and Devi [7] briefly outline several defects and fault tolerance approaches. No-fault tolerance approaches for defects were categorized in any of these surveys. The survey also only contains a few fault tolerance frameworks with no comparison. Amin et al. [5] also included a list of fault tolerance metrics and a brief description of fault detection. Again, minimal fault tolerance frameworks were formulated with no reflection on the fault tolerance methodology used in the framework. None of those, as mentioned above, surveys confront the entire structure of fault tolerance in cloud computing. Gayathri and Prabakaran and Patra [16] highlighted the taxonomy of faults and the requirement for fault tolerance, describing many methods for applying fault tolerance. Based on Metrics for fault tolerance in the cloud, various fault-tolerance models are regarded and contrasted. There are many fault tolerance models from Nuygen et al. [18] to ensure stability. An essential difficulty in fault tolerance is identifying the defective components inside a system. F-Chain, a fault localization technology developed by Black Box, can pinpoint the problematic components in a system as soon as an abnormality is identified [19,20]. The author had to refer to various sources to properly understand the fault-tolerance structure in this survey [144]. Consequently, we were motivated to write a comprehensive and systematized research study on cloud fault tolerance (a) by describing its entire structure, including descriptions of different fault forms, reasons, and categorization; (b) fault tolerance approaches, techniques, and framework analysis. Using the following qualities, Table 1 summarizes and compares the existing surveys referenced in this research with the current survey: Fault taxonomy, Failure taxonomy, Fault tolerance methodologies, comparative analysis, and graphical depiction.

Survey technique

In this study, we used to follow the instructions of Kitchenham et al. [29,144] methodology to conduct a survey and focus on research related to fault tolerance. We have developed a methodology for review and an investigation method to perform a systematic survey. This section discusses the sources of information for research articles, selection criteria, and quality evaluation.

Source of information

We extensively search for research articles and conference papers at Scopus, Web of Science, Google Scholar, books, and magazines as a data source, as shown below. In Fig. 2, we represent the percentage of research articles from different databases which we have used in our survey. The number of articles included in this survey year-wise data is shown in Fig. 3.

The following databases were used in our search to retrieve articles related to our domain:

- IEEE Xplore (https://www.ieeexplore.ieee.org
- Elsevier (https://www.www.elsevier.com)
- Springer (https://link.springer.com)
- Google Scholar (https://scholar.google.co.in)
- Scopus (https://www.scopus.com)
- ACM Digital Library (https://www.acm.org/digital-library)
- Sematic scholar (https://www.semanticscholar.org)
- ResearchGate (https://www.researchgate.net)
- Taylor and Francis https://taylorandfrancis.com

Selection criteria

Multiple articles were discovered using the final search strings after the string was generated, as shown in section 3.4. However, many papers were pertinent to our survey, as shown in the figure below. In Fig. 4, the technique is shown by which a total of 1002 papers were identified, of which 129 were permitted in this survey based on the selection criteria and some further analysis. The following explanations are as follows:

Some research publications were excluded due to their focus on

Table 1
Comparison with other cited surveys.

Survey paper	Fault tolerance approaches	Fault Taxonomy	Failure Taxonomy	Comparative analysis of Fault tolerance Framework	Visual representation of Results
Talwar and Bharany[2]	×	✓	×	×	×
Cheraghlou et al[3]	✓	×	×	✓	×
Agarwal and sharma[4]	✓	✓	×	×	×
Amin et al[5]	✓	×	×	×	×
Ataallah et al[6]	✓	×	×	✓	×
Saikia and devi[7]	×	✓	×	×	×
Tchana et al.[8]	✓	×	×	×	✓
Gayathri and Prabakaran	×	✓	×	×	×
Patra[16]	✓	✓	×	✓	×
Nuygen et al. (2013)	×	×	×	×	×
Kumari [9]	✓	✓	×	×	×
Present survey	✓	✓	✓	✓	✓

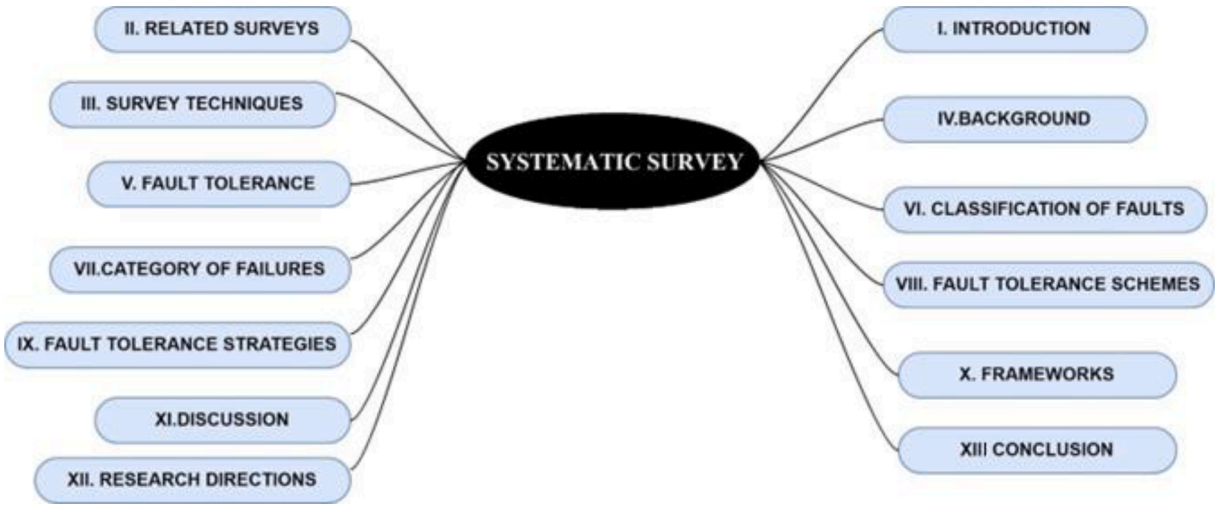

Fig. 1. Structure of the paper and topic classification.

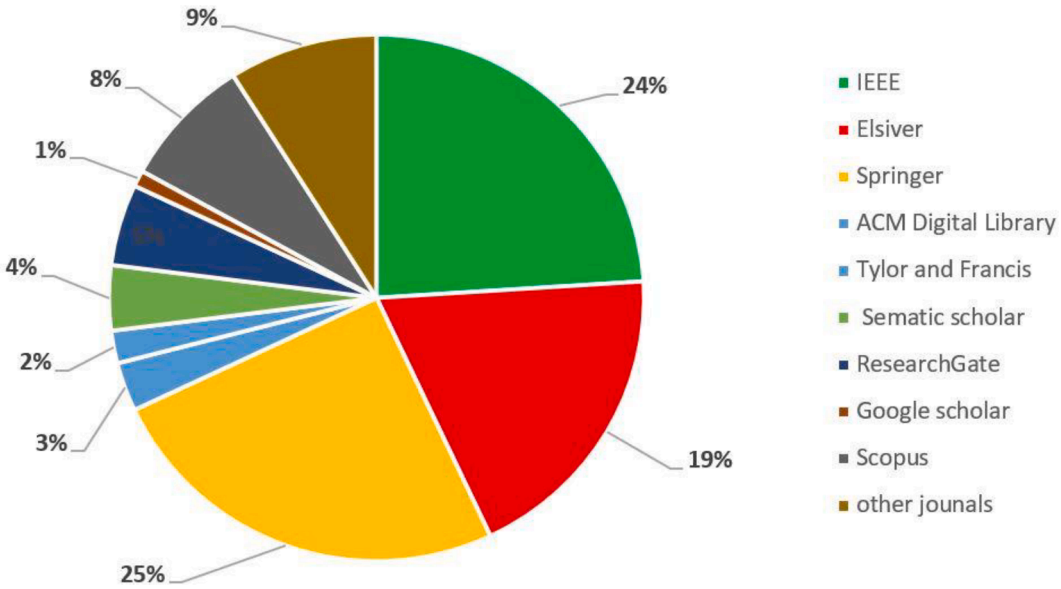

Fig. 2. Illustrating the percentage of research papers comprehend from several sources.

mechanical systems, thermal energy, mechanical equipment, etc, as our surveys are confined to cloud computing fault tolerance.

We include articles from January 2010 to December 2021 in our research. We delimited the keywords to search in the databases as mentioned above. The keywords "fault tolerance" and "cloud" were demanded in the abstract of every search. Moreover, we used to search by different synonyms and pertained keywords that matched our results, including "fault," failure management, "energy efficient," "distributed," "workload," "carbon-emission."

Quality assessment

This paper used a quality evaluation approach followed by inclusion and exclusion criteria, as mentioned in Table 2. We identified 1002 papers from various publications from multiple sources, including IEEE Xplore, Springer, Science Direct, and ACM, based on primary keywords. We rejected specific papers based on titles that were irrelevant to our research domain. After reciting the abstracts, we eliminated certain publications since they did not match our requirements. We thoroughly reviewed the remaining publications and discovered 129 research papers included in our review. Standards for inclusion and exclusion are mentioned in SM Table 1.Table 3.

The following criteria for inclusion in research publications:

- A primary focus on Fault Tolerance in cloud computing.
- Second, select research articles that depict fault tolerance challenges and parameters reliant on fault tolerance frameworks.
- •Thirdly, try to include and explain the articles with energy consumption concerns integrated into fault tolerance.
- Finally, the approaches, procedures, and various fault tolerance frameworks are described.

Search string

We followed the Kitchenham and Charters[29] guidelines when creating our search string. The steps below were used to prepare the preliminary string.

First, the key search terms were derived from the literature we studied.

The descriptors, synonyms, and alternate orthography lists for the

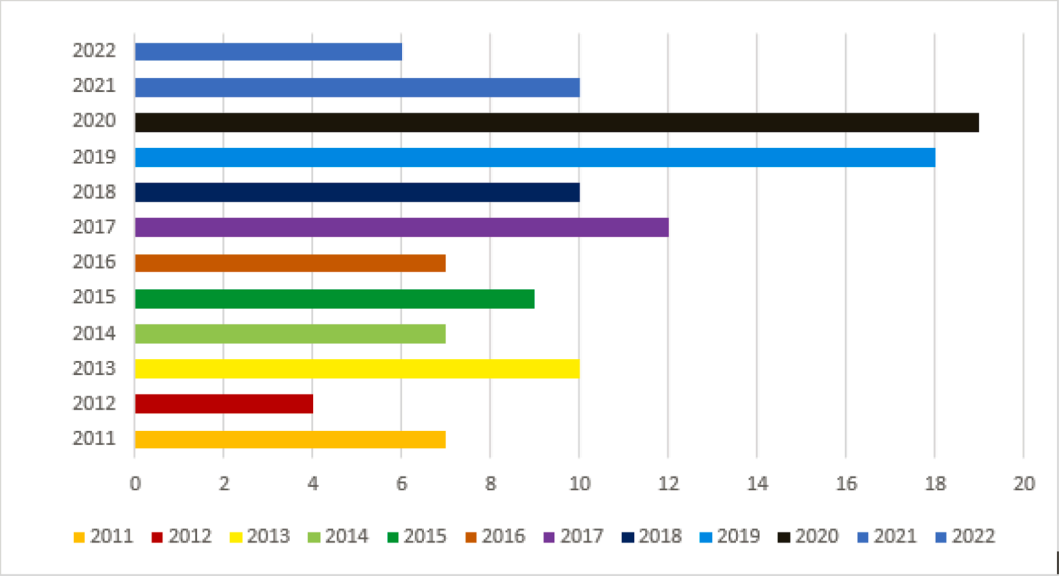

Fig. 3. From 2011 to 2022, the number of articles included in this survey is shown.

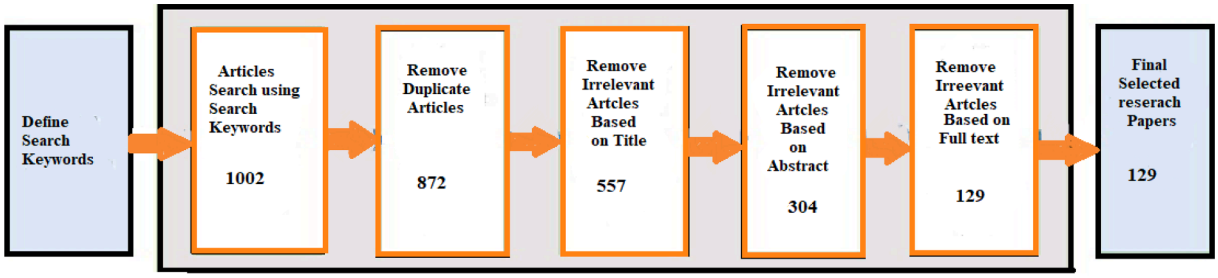

Fig. 4. Selection methodology used in this review paper.

principal search words are shown in the table below.

Finally, relevant search keywords were combined using Boolean AND operators.

The resulting preliminary string is: (Fault **OR** Failure **OR** Cloud computing **OR** energy **OR** power **OR** Fault-Tolerance **OR** Data center **OR** Fault & failures **OR** Network topology **OR** Fault tolerance frameworks **OR** Proactive **OR** Reliability **OR** High availability **OR** Reactive **OR** Error **OR** Bugs **OR** Fault-management **OR** mitigations **OR** Migration **OR** power consumption **OR** methods **OR** techniques Methods **OR** Schemes & techniques **OR** "' energy efficiency") **AND** (empirical* **OR** "case study" **OR** "case studies" **OR** experiment* **OR** Review **OR** survey **OR** Systematic survey) **AND** (Fault Tolerance * **OR** Techniques *).

We carefully chose 29 publications to check the search keywords with the initial search phrase (Fault tolerance and Management). The search strings will look for the Title, abstracts, and article keywords in each electronic database mentioned in Section 3.1. As a result, we double-checked the search terms for the title, abstract, and keywords from the 29 articles that were nominated. We anticipate that this assessment will ensure that all relevant material published in designated digital sources is fully covered. However, only 18 of the 29 selected articles were acquired by our original search string, allowing us to broaden our search strings for better article distribution.

We had defined more search strings for each digital library even though various digital libraries require different search formation/syntax guidelines, as provided in Table 2. Before transitioning to a data-gathering strategy based on constructed search phrases, we conducted a pilot investigation in selected areas' digital libraries. The test sequence set was then compared to 29 manually created reference papers. The search string yielded 26 of 29 results indicating that our search string was successful.

Background

Cloud deployment models

Cloud deployment models are a particular cloud environment segmented based on ownership, size, and accessibility.

i. **Private Cloud**: The private cloud is employed within a particular enterprise. The NIST definition is: "Cloud infrastructure is designed for exclusive use by an organization involving a large number of users (i.e., business units)" [29–31].
ii. **Public Cloud:** The NIST definition is: "Cloud infrastructure is intended for use by the public. It can be managed, run, and owned by a business, an academic, or government organization, or any combination thereof ". It allows participants to use an enterprise provider's systems and services [29]. Because of this, multi-tenancy is commonly employed and enables flexibility, scalability, and location independence at a meagre cost.
iii. **Hybrid Cloud:** Private, community, and public cloud infrastructures are monolithic [29] but connect with standard or proprietary technologies to offer data and application convenience [32].
iv. **Community Cloud**: Multiple enterprise organizations use this approach simultaneously to assist a specific community or society with communal involvements (such as security procedures,

Table 2
Search strings determined for digital libraries.

Digital library	Search string
ScienceDirect	((cloud OR Fault OR Failure OR Fault-tolerance power OR computation OR energy OR cloud computing Network OR Error OR Bug OR mitigation OR migration {power-consumption} OR {Fault tolerance }) AND (Failure * OR {Fault management } OR {Fault tolerance-techniques} OR {Proactive techniques} OR {Reactive } OR {Failure management} OR {applications} Methods OR Frameworks))
IEEE Xplore	Cloud OR Fault OR Failure OR Fault-tolerance power OR high computation OR energy OR cloud computing Network OR Error OR Bug OR mitigation OR migration {power-consumption} OR {Fault tolerance}) AND (energy * OR Fault tolerance * Or Reliability Or availability)
SpringerLink	((Cloud OR Fault OR Failure OR Fault-tolerance power OR computation Data centers OR energy OR cloud computing OR Network OR power-consumption OR Error OR Bug OR mitigation OR migration AND (Fault tolerance techniques * OR "Failure management" OR "energy-management" OR "energy-efficient fault tolerance" OR "reliability" OR Reactive Or Proactive))
Scopus	title-abs-key ((Cloud OR Fault OR Failure OR Fault-tolerance power OR computation Data centers OR energy OR {cloud computing} Or {Network} OR {power-consumption} OR {Error} OR {Bug } OR {mitigation } OR {migration } AND ({energy-efficient techniques} * OR "Failure management OR {energy-management} OR {energy efficient fault tolerance} OR { reliability } OR { Reactive } Or Proactive))
ACM Digital Library	(Title: ((Cloud OR Fault OR Failure OR Fault-tolerance power OR computation Data centers OR energy OR (cloud computing) Or (Network) } OR {Error} OR {Bug } } OR {migration } AND ({energy-efficient techniques} * OR "Failure management OR {energy-management} } OR { reliability } OR { Reactive } Or Proactive)) Keywords:("power-efficiency") AND (energy-efficient Failure management s * OR "power-reduction" OR "fault-mitigation" OR Fault tolerance" OR "Failure management")))

Table 3
Various types of faults.

Type of Faults	Description
Network Faults:	A fault that occurs on a network due to network consequences, such as connection failure, location collapse, packet contamination, packet loss [30,32].
Physical Faults:	Faults caused by hardware resources such as the CPU, memory, storage, and power failure, among several others [30].
Process Faults:	Any issue which is caused by a lack of resources, a software error, or other software related factors [30–32].
Service Expiry Faults	When a program tries to use an asset and the asset's duration expires [32].
The Parametric Faults	These occur due to an inexplicable variation in parameter values [32].
Contributor Faults	These emerge from a disagreement amongst cloud users such as consumers, providers, and administrators [31].
Resource Contention Faults	A disagreement occurs when a shared resource is used [31].
Time Restriction Faults	This is a situation that causes an application to be considered incomplete after the specified deadline [32–35].
Media Faults	A fault will arise due to media head collisions in clouds [10] and a shortage of communication media [33].
Processor Faults:	A processor error encountered due to an operating system problem [34].
Ordering Faults	This happens when the system's component ordering is interrupted [35].
Restrictions Faults	This occurs when a fault situation happens, and the responsible agent fails to notice it [35].

vision, compliance issues, and so on) [32]. It may be owned, operated, and managed by a third-party organization or any combination above.

Service models

Cloud Computing introduces the concept of services that includes everything, mainly referred to as "XaaS", where X can be software, infrastructure, hardware, platform, data, business etc. The three services which NIST defines are shown in SM Fig. 1:

i. **Software-as-a-Service (SaaS)** that provides complete applications or software to the user on-demand [8](e.g., Dropbox, Google Docs).
ii. **Platform-as-a-Service (PaaS)** provides an environment to the user to construct their application that runs on a service provider base, e.g., Amazon Web Services, Salesforce.com, MS Windows Azure).
iii. **Infrastructure-as-a-Service (IaaS)** that provides services of fundamental storage and computing capability [2] (e.g., Amazon EC2, Go-Grid).

Issues and challenges in cloud computing

Fault tolerance

Fault tolerance is the potency of a computer system or a network to prevent any failure or a fault. While execution deals with the practical step to prevent this error or failure. One of the critical concepts of fault tolerance is that it must be sure about the availability and reliability of a system. Fault tolerance is a method in which an error or bug that occurs will not affect its current execution.

These faults can be of different kinds like.

· Transient, intermittent, or changeless equipment faults.
· Software bugs and planning mistakes.
· Operator faults.
· Externally actuated faults and failure.

Load balancing

Cloud load balancing is a method by which the distribution of workload and resources is done in cloud computing to satisfy proper load balancing and availability.

VM migration

Virtualization allows cloud administrators to move virtual machines (VMs) across physical devices and manage resources that users can access. VM migration is an important event in current cloud installations. It helps perform administrative tasks (such as removing VMs from inactive physical nodes) and provides advanced resource management policies such as load balancing. VM migration is carried out either by direct migration or by discontinuation. Starting a migration misses the adeptness to react to unexpected workload alterations and detect workload hotspots. The optimization technique used to migrate the current VM to another healthy one is known as Live VM migration, without turning off any machine. Physical and virtual machines are worked together; therefore, it is known as live VM migration. Several techniques are available to be used under Live VM Migration. These techniques are discussed as under:

Pre-copy migration. The entire VM contents (i.e., storage and memory) are relocated in this approach. In a pre-copy approach, the pages from the source machine are copied to the destination machine iteratively. The iterative nature of the pre-copy approach is due to dirty pages. This approach is functional when a modification to the pages is limited. In case of heavy changes in pages, this approach considerably slows down the migration process; hence downtime and migration time is substantially increased. Such an approach is used in Xen VM Migration.

Post copy. In the post copy approach, the processing or execution is first

transferred and then the memory. This means that execution immediately starts after memory resources are transferred to the destination machine. Downtime is considerably reduced in this case. Migration time depends upon the length of memory resources and the number of CPUs required to execute the job.

Hybrid approach. This approach is optimal among pre and post-copy approaches. This approach contains processor state information and lots of helpful information about the state, a working set.

Interoperability

Interoperability has been defined as measuring the degree to which various components or systems work simultaneously and effectively. According to IEEE and ISO definitions, two or more systems or applications can communicate information and mutually benefit from the information transmitted. Interoperability is needed for the following reasons [26] to protect users' capital invested for developments, Expansion of the cloud market and ecosystem, and Utilization of comprehensive benefits of the "pay-per-use" and elasticity concept. Interoperability requirements for three cloud service models (Iaas, Paas, SaaS) are discussed below.

IaaS interoperability. Interoperability in IaaS is the capability of the end-user to get infrastructural resources from distinct service providers by using a standard mechanism. Some essential issues like access mechanisms, storage, network, security, and Service Level Agreement (SLA) must be addressed.

PaaS interoperability. Interoperability in PaaS is the ability to get tools, libraries, or APIs for application development through distinct service providers. Interoperability of applications deployed on PaaS is limited because the precise application was developed using a distinct programming language, and hence no standards are developed for interoperability.

SaaS interoperability. SaaS allows specific applications to reciprocate information. Enterprises maintain and store data in prevailing enterprise applications, ensuring that every SaaS application is interoperable.

Server consolidation. Server Consolidation is used in cloud computing to reduce energy usage and increase resource sharing while hiding server resource specifics from consumers. It is independent of program performance.

Performance unpredictability. Sharing, the input/output tool, is a complex task in cloud computing, while a couple of VM can easily divide CPU and main memory. Virtualization of I/O interrupt is one of the possible methods to enhance the operating system's performance, and architecture can also be improved.

Quirks of cloud computing

Various characteristics of cloud computing are summarized as follows SM Table 2 and Faults are broadly classified on the basis of characteristics that are partially as shown below in SM Fig. 3:

Results and discussion

Fig. 5 depicts the pertinence statistics of fault tolerance strategies, demonstrating the dominance of reactive procedures (with 82 per cent applicability) over proactive approaches. Continual system control is critical for aggressive techniques as they focus heavily on learning and prediction while employing probability theory and artificial intelligence. Reactive strategies are primarily reflected by the replication method (30% applicability), restarting checkpoints (32% applicability), and job migration (13% applicability), Self-healing 6%, System Rejuvenation 9 %, as shown in Fig. 6. The chart shows how frequently a given fault type (crash or byzantine) is targeted in the literature. It is worth mentioning that 44% of frameworks target crash faults, 25% target byzantine issues, and 31% target both, as shown in Fig. 7. It demonstrates that researchers are slightly more interested in crash fault tolerance. It might be because the identification and tolerance of crash faults appear to be easier to replicate than those of byzantine defects. However, there might be some additional factors also [34]. In Table 7, all the frameworks are being evaluated on specific parameters. In this Fig. 8, it can be seen what percentage of faults are being studied by the researcher in the literature.

Research directions

Nowadays, the cloud is being adopted by researchers in industry and organizations. Fault tolerance is a fundamental concept in cloud computing as it improves the quality of service in the cloud environment. The service provider's main problem in cloud computing is task scheduling, the prediction of virtual workload, resource provisioning, security issues, and denigrating energy consumption and cost. Each of these problems can be solved efficiently by the deep learning approach. Deep learning provides better performance accuracy, so we can find many ways to integrate fault tolerance with deep learning techniques. Cloud computing is an enormous resort for business, commerce, education, and information technology professionals. Its consumer base has continuously risen throughout the years. Service dependability is one of the essential aspects of cloud computing, and as a consequence, an

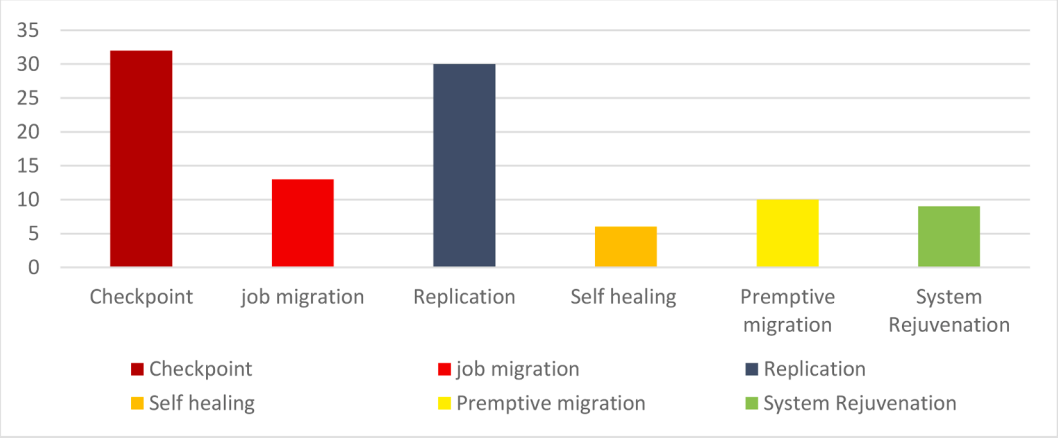

Fig. 5. Applicability count of different techniques.

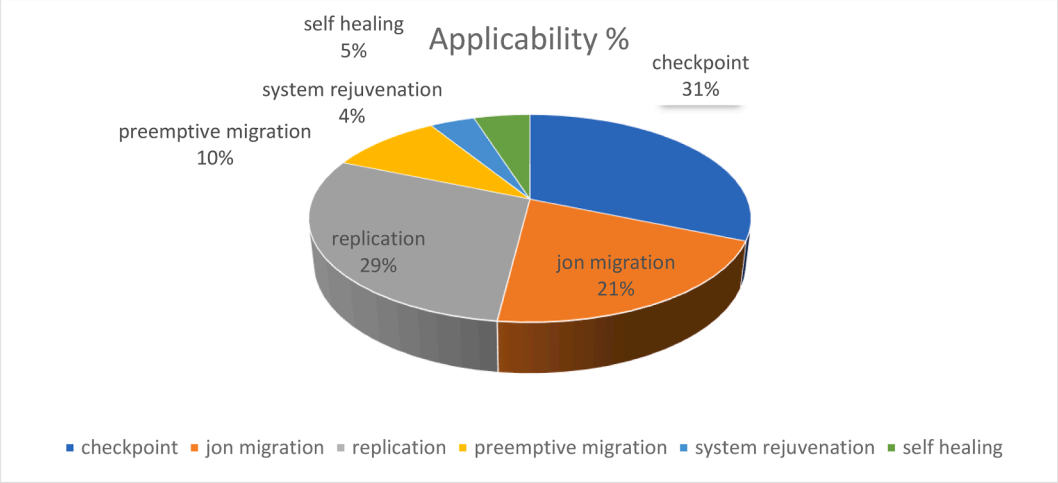

Fig. 6. Applicability % of FT methods.

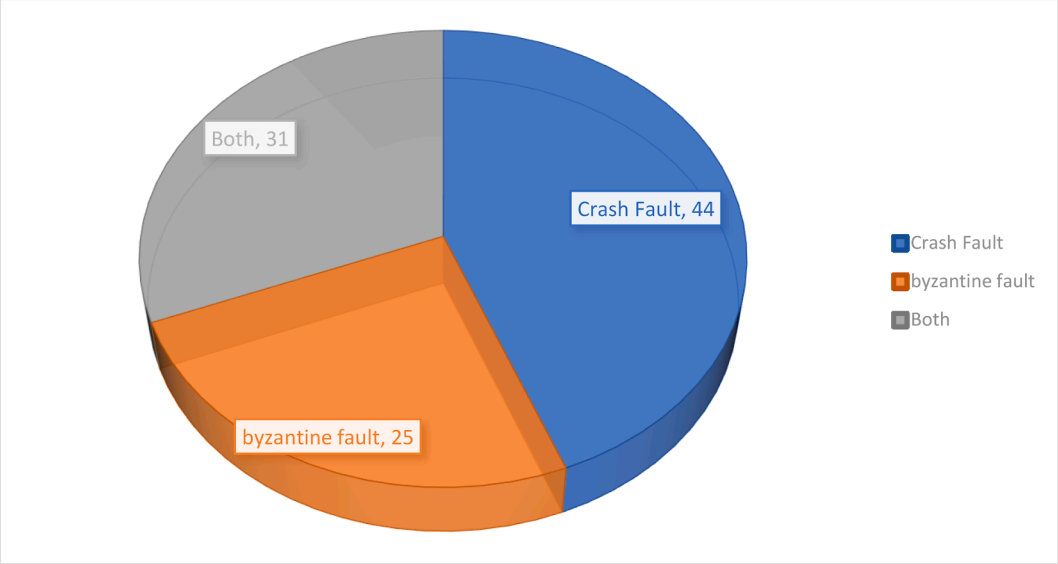

Fig. 7. Percentage of Faults research paper are published.

excellent system completion rate is achieved [34]. According to the research papers in this study, it is apparent that there are currently a substantial number of fault tolerance solutions that are mainly focused on reactive and proactive approaches. In the context of cloud systems, the efficacy, adaptability, and robustness of these techniques must be demonstrated. Existing methods are not adaptable and need some manual intervention for proper operation and specification; as a result, we recommend that automation be contingent on the future of fault tolerance in the cloud [1]. By aiming to achieve some of these challenges, we will witness the growth of agent-based cloud computing as a solution to overcome automated cloud concerns.

However, we suggest that agent-based techniques, such as reinforcement learning, are the developing avenues in the cloud for automated fault tolerance [12]. An agent is a fully functional computational entity that can make independent decisions and communicate with other agents through cooperation, communication, and negotiation. Many essential cloud functions, such as resource management, work scheduling, and fault tolerance, are automated using agent-based processes [1]. New machine learning approaches like deep learning can be used to support agent-based automation processes [2327 34]. Some More suggestions for further research have been discussed below:

a) Offer some more cloud failure tolerance techniques that are resilient/hybrid [1].
b) Grouping and load balancing can be used to determine fault tolerance [36].
c) Cloud failure tolerance may be adjusted based on cost [2]. Providing failure prediction and detection technologies that are accurate [34].
d) Increasing cloud job fulfilment efficiency [34] and improving cloud task completion rate.
e) Furthermore, optimizing constructive fault tolerance techniques over-head [3436]. There must be an Optimization of the costs associated with proactive fault tolerance solutions.
f) Preparation of fault tolerance in the cloud to battle any defects and Work on enhancing the reliability of real-time systems
g) There is a necessity to develop a framework or model with a proactive fault tolerance method to predict and find the reason for the fault.
h) Advice to construct a robust algorithm for scheduling the task with the elastic resource provisioning

Though reactive fault tolerance methods have been popular among academics up to this point, as machine learning advances, gadgets are

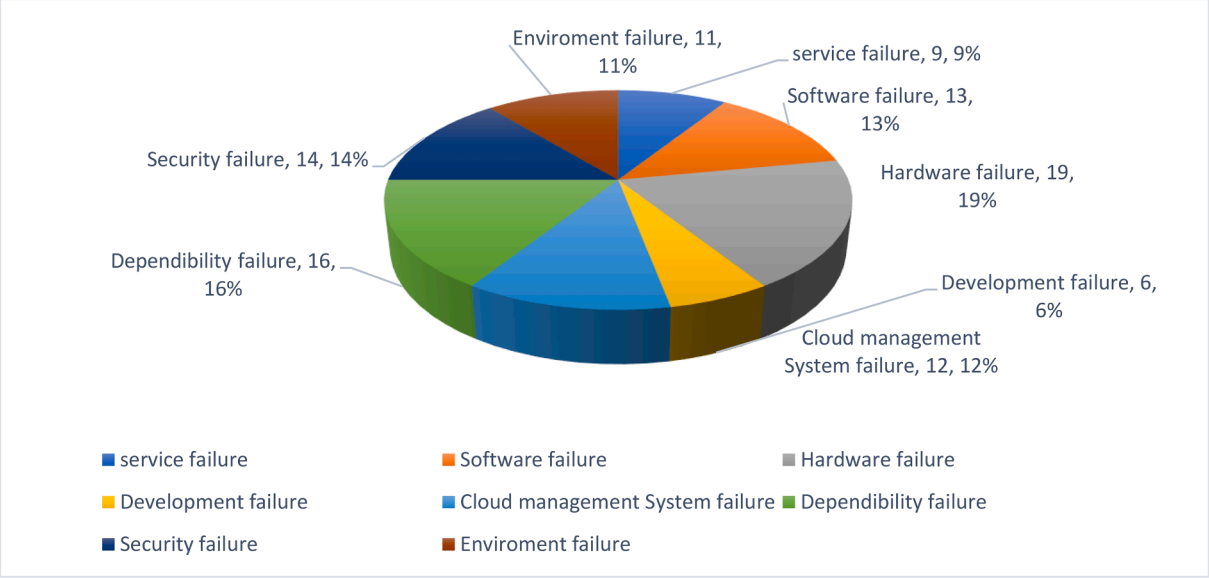

Fig. 8. Percentage of framework target what types of faults.

getting more innovative and more artificially intelligent, expanding the area of study into proactive fault tolerance solutions (specifically self-healing). Overhead Optimization is also one of the unresolved research issues in cloud computing. It can significantly increase the application of proactive approaches amongst both researchers and professionals. Cloud applications need a high level of usability and dependability. As a result, designing a fault tolerance architecture (using a specific fault tolerance mechanism) which is optimal and efficient is the need of the hour. In the literature, several frameworks (both proactive and reactive) used checkpoint restart as an additional technique to trigger fault tolerance. However, it has been discovered that preemptive migration may be a good choice as an auxiliary. To achieve "ZERO" downtime, replication is commonly employed in storage applications. Passive repetition is used in computational applications with checkpoint restart to decrease computational downtime; however, it is tough to accomplish zero computational downtime in this scenario.

However, employing proactive migration rather than checkpoint restarting with passive replication may suffice. Furthermore, existing frameworks only manage a few problem kinds, and to the best of our knowledge, no framework has been deemed sufficient fault-tolerant for handling all fault types. From a research standpoint, innovative fault tolerance frameworks (clever enough to apply multiple techniques in different scenarios) are predicted to oppose all distinct problem kinds. Moreover, the current fault tolerance approaches suggested in the literature mainly focus on service providers. The consumer is given ready-made fault-tolerance measures that have been established at the service provider's end. Cloud computing is a pay-per-use service architecture in which clients only pay for the services they utilize. When a client may choose the needed assistance from the cloud using the pay-per-use service model, they should also have the ability to select the service level guarantee in terms of fault tolerance. As a result, a cloud service orchestration system that encompasses service selection and price-oriented flexible fault tolerance is required. Fault tolerance combines proactive fault predictions, detection, and recovery (reactive). Most literature-based models focus on the first aspect, i.e., fault prevention and recovery. Cloud data centres have thousands of physical hosts with varying reliability [25]. It highlights how difficult it is to predict and detect errors in a vast network. For this reason, complete fault tolerance (forecast/detection prevention/recovery) frameworks are greatly desired.

Conclusion

This paper provided a review of various methods for implementing fault-tolerance in distributed cloud systems and a comprehensive survey on the mechanism used in the cloud to enhance the reliability of the cloud., understanding faults and their rectification procedures thus become critical as cloud services are commonly hampered by performance degradation due to severe faults. This study article extensively covered all types of failure and faults (with all of their impacts) and numerous fault tolerance approaches in cloud computing. The fault tolerance techniques are classified into reactive, proactive, and resilient. The reactive and proactive approaches mainly focus on the traditional fault-tolerance methodology, such as replication, checkpointing, recovery, tracking, and preemptive migration, as such conventional procedures have limitations. Initially, they are known as determined principles and handle problems differently, as described by their deployment. As a result, they overlook the ability to control new defects that may arise in the future.

Moreover, such implementations only consider the device's fundamental properties when addressing failures. External or environmental factors that might affect device output (such as temperature, electricity, and weather) are relatively minor. The basic Fault handling strategy/techniques, fault applicability, and significant characteristics of prominent fault tolerance frameworks have also been evaluated. The following findings have been reached:

Researchers are more interested in resolving crash defects than byzantine problems.

Reactive fault tolerance techniques are more commonly utilized than proactive fault tolerance techniques.

The authors' reluctance to employ proactive techniques is attributed to higher overheads and rugged design.

Fault tolerance techniques are widely used for replication, checkpoint, restart, and job relocation.

Checkpoint restart and task relocation are supplementary replication mechanisms in various approaches.

We suggest that a proactive approach could be a need of the hour and can be used in future along with a job partitioning algorithm to enhance the reliability of operation. Fault tolerance in the cloud may allow the application of machine learning techniques as part of their fault tolerance solution. Few solutions have previously relied heavily on machine learning to predict outcomes based on characteristics. Machine learning has been employed in several applications when dealing with hardware

issues.

Moreover, hardware devices are fixed and not dynamic enough to handle future and unforeseen problems. The paper contains a comparative examination of several fault tolerance techniques and frameworks, which will help researchers to choose the framework and develop new designs according to their requirements. Moreover, this study identifies several obstacles associated with deploying fault tolerance in the cloud and fingers the limits of current fault tolerance approaches and breakthroughs. The applications deployed in the cloud to provide services to the users encompass many interconnected dependent cloud components [149-Ŕ].

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Appendix A

Tables 4-6.

Appendix B

FAULT TOLERANCE STRATEGIES

As technology grows, the size of the CPU keeps on shrinking, and memory capacity and energy demands keep on increasing. Soft faults, which are difficult to detect, exist due to this advancement. These faults eventually cause the entire system to shut down. Fault tolerance strategies hence are essential, and the taxonomy of the Fault tolerance technique is shown in SM Fig. 6.

Proactive Fault Tolerance

A proactive approach allows faults to be corrected before they occur within the system. This approach is suitable in an environment with technological advancement, as measures are taken to avoid defects by predicting the system's behavior at regular time intervals, and decisions are made based on predictions only. A proactive approach ensures continuous and reliable service availability as zero downtime exists since the decision is taken before failure. Along with fault prediction, energy consumption can also be a critical parameter. The proactive approach has the following techniques associated with it.

Preemptive Migration

The feedback loop control method is wont to accomplish preemptive migration in which Applications are detected and examined regularly. Rakesh [50] proposed a proactive migration strategy in which the app's essential assets, e.g., RAM and CPU utilization, are continuously monitored. Before the process is rolled out on existing hardware with the acquisition, any apps demanding more assists than the same app are relocated to the system's least loaded hardware. If the hardware temperature accumulates, all applications must be resettled appropriately to the network's lowest charged hardware. Thanh et al. [83] proposed controlling and reviewing applications using a management program. Based on that information, the software would evaluate whether a problem occurs. If any, then move the app to a safer place. This process is known as proactive migration.

Software Rejuvenation

Programming Rejuvenation is the issue that emerges because of maturing of the product. At first, it is crucial to group the faults

Table 4
Reactive frameworks.

Framework	Techniques	Application scenario	Key features	Limitations
AFTRC [34]	Reactive	Configuration and parametric	Precise, Low capacity overhead and available in line time	Limited experimentation and exploration restricted
BlobCR [34]		Hardware faults	Low storage capacity/ overhead	Unsuitable for couplings which are unstable.
BlobCR 2013 [34]		Hardware faults	Live snapshooting in incremental manner	Very low resource utilization
BFTCloud		Byzantine Faults	Strong trust and more reliable	Inefficient asset utilization
AASIF [34]		Resource contention	High Usefulness of resources	Slow fault tolerance
DAFT [34]		Crash faults	Adaptive, Overhead configured	Programs affect adaptability.
FTCloud 1 [25]		Software faults	Incredibly Accurate	Challenging execution
FTCloud 2 [25]		Software faults	High accuracy	
CAMAS [25]		Resource contention faults	High reliable and very low execution cost	Tolerate particular types of flaws only
FTM [25]		Crash faults	As a service, fault tolerance is used	High cost and difficult deployment
FTWS [25]		Parametric, resource contention, stochastic, and participant faults	Brilliant asset utilization	Usefulness is limited.
FESTAL [25]		Host failures	Excellent asset utilization	Inapplicable if the primary and backup systems both fail at the same time.
FWSS[34]		Parametric, resource contention, stochastic, and participant faults	High resources consumption	Limited applications
MVRS[24]		General crash and byzantine faults, malicious attacks	Secure, mission-critical computing assistance	Mellow overhead
K-OUT-OF-N[25]		Hardware and network faults	Energy-saving	Low resources utilizations
CACS[34]		VM crashes, unhealthy applications	Allows cloud migration	There will be no optimization for overhead.
AFRCE [34]		General faults	Adaptive	Throughput is low
IRW[34]		VM crashes	Low production time, high job completion rate	Applications with tight deadlines should not use this product.
SAFTP [34]		Software faults	Adaptive	Experimentation with a limited scope.
IVFS[34]				

(continued on next page)

Table 4 (*continued*)

Framework	Techniques	Application scenario	Key features	Limitations
		Host failures and software faults	Checkpoint overhead is minimal.	Experimentation with a limited scope.
FTESW [34]		Host failures	Brilliant utilization of resources	Inapplicable if both the main and backup fail at the same time.

Table 5
Proactive and hybrid frameworks.

Framework	Techniques	Application scenario	Key features	Limitations
SHelp [25]	Proactive	Software faults	Service that is faster and more efficient.	Tolerate interstitial types of faults
PFHC [25]		Hardware faults	Execute at a lower cost than PFHX	Complex execution
WSRC [25]		Software faults	Superb quality with a regulated overhead.	Only accurate on the network level.
SRFSC [25]		Software aging and crashes	Virtual machines may be rejuvenated in several ways due to their high availability.	Restricted forms of disaster
FTDG [25]		Parametric faults	Only a short time to respond	Low availability of a defect
ASSURE [34]		software faults	No changes required in Os kernel	Sometime works very slow
PFHC[24]		Hardware faults	Lower execution time	Implementation process is very complex
EFTT [34]	hybrid	General faults	Adaptive technique	Slow in working
FTESW [24]		Host failure	High resources consumption	No solution if primary and backup fails parallelly

happening inside the product framework. The expository methodology is connected with the end goal of deciding the ideal number of times restoration is required. Kimia Rezaei Kalantari et al. [73] proposed a fuzzy framework for software rejuvenation in the form of a proactive fault tolerance method. To solve the issue of aging in electronics, software rejuvenation has been introduced. Nonetheless, software rejuvenation never addresses the fundamental difficulty with out-of-date equipment. As a result, software rejuvenation occurs unpredictably over a predetermined or planned time to maintain the software system's firmness. Tamilvizhi and B. Parvathavarthini [74] proposed a software rejuvenation approach that focuses on length and forecasting to avoid application ageing. This software rejuvenation method based on size and forecasts improved server data performance while also lowering leisure time expenses. However, this has contributed to gadget overload, the most severe flaw. According to Philippe Marcotte et al. [81], software rejuvenation is the "concept to continually and preemptively reboot an app after every period of rejuvenation at a pristine internal state."

Auto Recovery

When several occurrences of an application are running on different

Table 6
Framework descriptions.

Framework	Technique	Methods
AFTRC [34]	Reactive	Replication, checkpoint Restart
BlobCR [34]		Checkpoint restart
BlobCR 2013 [34]		
BFTCloud[34]		Replication
AASIF [34]		Job migration, replication
DAFT [34]		Checkpoint restart
FTCloud [25]		Replication
FTCloud 2 [25]		
CAMAS [25]		Checkpoint restart
FTM [25]		All strategies are reactive
FTWS [25]		Checkpoint restart, Job Migration
FESTAL [25]		Replication
FWSS [34]		Checkpoint, restart and job migration
MVRS[25]		Checkpoint, restart, replication
K-OUT-OF-N[24]		replication
CACS[34]		Checkpoint restart, job migration
AFRCE[34]		Checkpoint restart, replication
IRW[34]		Job scheduling
SAFTP[34 34]		Replication
IVFS[34]		Checkpoint restart, replication
FTESW[34]		Job migration, replication
SHelp [25]	Proactive	Self-healing, checkpoint, restart
PFHC [25]		Preemptive migration
WSRC [25]		System Rejuvenation
SRFSC [25]		System Rejuvenation
FTDG [25]		Preemptive
ASSURE[34]		Self-healing, checkpoint, restart
EFTT [25]	Hybrid	Checkpoint restart, replication
FTESW [34]		Job migration, replication

virtual machines, it handles the failure of use cases. According to Cheraghlou et al. [75], only the self-healing mechanism should be employed in a fault-tolerance cloud. The components mentioned above conduct the fault recovery methods after the fault tolerance system has identified the problem to ensure that the final results are error-free. According to Kasem Khalil et al. [76], self-healing is increasingly feasible for designing secure digital systems. It allows a device to detect and rectify faults or vulnerabilities by healing or mending them. Self-healing mechanisms in digital systems are designed to compensate for flaws. According to Sona Ghahremani and Holger Giese [82], a self-healing method (SHS) may discover and diagnose operational disturbances such as faults and respond to them efficiently by modifying and restructuring the system. Limitations of self-healing systems are that adjustment is only necessary when faults happen.

Prediction

According to Mohammed Amoon et al. [84], proactive approaches exceed cost-effective reactive procedures yet allow for high fault diagnostic precision of at least 80%. According to Bashir Mohammed et al. [77], model prediction requires a fundamental understanding of the characteristics of real system faults. Some number of defects might help designers assess and create a fault-tolerance framework that is both safe and efficient.

Monitoring

When it comes to offering high-end resources like management, expansion, and migration, Sudheer Kumar Battula et al. [78] believe that monitoring services are essential in which CPU, RAM, bandwidth, and power are examples of platform assets that can be monitored and reported to the platform's user or operator. As part of a monitoring tolerance approach for large datasets, Yuling Fang et al. [85] suggested a monitoring model for GPU cluster energy consumption, a framework for

Table 7
Framework comparison based on specific parameters. ⇑ = **high,** ⇔ = **average, low** = ⇓.

Frameworks	Category	Execution Efficiency	Latency	Measurability	Throughput	Reliability	Accessibility	Usability	Processing overhead	Cost-efficient
AFTRC [25]	Reactive	⇑	⇔	⇑	⇑	⇑	⇑	⇑	⇔	⇔
BlobCR [25]		⇓	⇔	⇑	⇔	⇑	⇑	⇑	⇓	⇑
BFTCloud [25]		⇑	⇓	⇑	⇑	⇑	⇑	⇓	⇔	A⇔
AASIF [25]		⇓	⇓	⇔	⇔	⇓	⇑	⇑	⇔	⇑
DAFT [25]		⇑	⇑	⇔	⇔	⇑	⇓	⇔	⇑	⇔
FTCloud [25]		⇑	⇔	⇑	⇔	⇑	⇔	⇑	⇑	⇑
CAMAS [25]		⇑	⇔	⇑	⇔	⇑	⇔	⇑	⇓	⇓
FTM [25]		⇔	⇔	⇓	⇔	⇔	⇑	⇔	⇓	⇓
FTWS [25]		⇔	⇔	⇓	⇓	⇔	⇔	⇔	⇑	⇑
FESTAL [25]		⇑	⇔	⇓	⇑	⇑	⇑	⇑	⇔	⇓
FWSS[34]		⇓	⇔	⇑	⇔	⇑	⇑	⇑	⇓	⇑
MVRS[34]		⇑	⇔	⇔	⇑	⇓	⇔	⇑	⇓	⇔
K-OUT-OF-N [34]		⇑	⇓	⇑	⇑	⇑	⇑	⇓	⇔	⇔
CACS[34]		⇓	⇓	⇔	⇔	⇓	⇑	⇑	⇔	⇑
AFRCE[34]		⇑	⇑	⇔	⇔	⇑	⇓	⇔	⇑	⇔
IRW[34]		⇑	⇔	⇑	⇔	⇑	⇔	⇑	⇑	⇑
SAFTP [34]		⇑	⇔	⇔	⇑	⇓	⇔	⇑	⇑	⇔
IVFS[34]		⇑	⇓	⇑	⇑	⇑	⇑	⇓	⇑	⇓
FTESW[34]		⇓	⇓	⇔	⇔	⇓	⇑	⇑	⇑	⇓
SHelp [25]	Proactive	⇑	⇔	⇓	⇑	⇑	⇑	⇑	⇔	⇔
PFHC [25]		⇑	⇔	⇑	⇑	⇔	⇑	⇑	⇔	⇓
WSRC [25]		⇑	⇑	⇓	⇑	⇑	⇑	⇔	⇑	⇔
SRFSC [25]		⇔	⇔	⇑	⇑	⇑	⇑	⇔	⇔	⇔
FTDG [25]		⇑	⇓	⇓	⇔	⇔	L	⇔	⇑	⇔
ASSURE[34]		⇑	⇔	⇑	⇑	⇑	⇑	⇑	⇔	⇓
PFHC[34]		⇑	⇑	⇓	⇑	⇑	⇑	⇔	⇑	⇔

fault tolerance that relies on energy use, and dynamic task migration.

Reactive Fault Tolerance

Reactive fault tolerance mechanisms tackle the faults when they appear within the system. This means these techniques allow the imperfections to occur within the system, and then a rectification mechanism is implemented to ensure a fault-free system. The downtime prevails for a specific time interval due to failure in reactive approaches. It requires continuous monitoring, during which a massive amount of energy is consumed. The reactive approach has the following techniques associated with it.

Checkpointing

In this, if a failure occurs during the execution of the task, then rather than starting from the initial point, it restarted from a recently checked point. It is the most common strategy used to tackle faults. Ghosh et al. [117] propose a mechanism for conserving energy within a high-performance cloud using the checkpointing technique. Data funnelling and core power capping mechanisms are used to achieve power efficiency using a checkpointing strategy. Wang and dung [118] proposed an energy-efficient fault-tolerant checkpointing approach in which the workload is divided into a set of chucks. In case of failure, it is identified which chucks to re-execute. Hence, it is not required to execute all the chucks in case of loss, thereby conserving energy and fault tolerance. Xia and Chen[119] proposed a message logging and checkpointing mechanism for fault tolerance within the distributed system. A message logging mechanism is used to maintain the information about the system's stability. The system's stability in case of message delivery failure is at stake. This leads to the initiation of a checkpointing mechanism. However, in incremental checkpointing, the entire running states of a system are included in the first checkpoint and succeeding checkpoints only retain pages that have been updated since the last checkpoint. Coordinated checkpointing mechanisms generally depend upon a collaboration of operating system or user level runtime library support, whereas uncoordinated checkpointing tools depend upon logging messages only. Check-pointing can be Disk & Diskless, which is done with the help of MPI. Disk-based stores data on global disk storage and is used for a node or network failure, while diskless stores data on local storage and is used for process or application failure. Mohammed and Bashir [79] proposed an effective fault tolerance technique for long-term tasks and HPC tools. If a system failure happens, the system is set up to restart from the last safe, active checkpoint location rather than from the beginning. Edemo [121], proposes a strategy that is an effective reactive method, especially for long-running services. Checkpointing refers to retaining the app status after any successful completion or saving the preliminary failure-free state. This strategy is implemented at the device, app, and library levels. A checkpoint is an appropriate value stored in the program to which the function may be rolled back if the system fails, according to Soma Prathiba and S. Sowvarnica [91]. It is more effective than starting at the beginning of the cycle. This means Restarting from the most recent checkpoint reduces computation and energy consumption, according to Nivitha and P. Pabitha [93].

Duplication or Replicated Shadow

It means dummy or copy. The basic concept of shadow replication is to associate with the primary process of each made a shadow within a large-scale distributed system. The Shadow replication mechanism is described through the following steps.

(i) A process is assigned an exclusive core processor to accomplish a job.
(ii) The procedure has several Shadows.
(iii) Every Shadow has the same performance and memory allocation as the original process.
(iv) In case of failure, the primary process proceeds to the Shadow, which becomes the primary processor.

Replication is an approach used to enhance reliability and achieve fault tolerance. Sun and wang [120] proposed a dynamic replication mechanism for improving availability within the high-performance cloud environment. Critical information is maintained at distinct nodes. In case of failure, information can be retrieved from other nodes.

The main problem with this approach is the amount of space utilized; hence more energy is consumed, and a performance evaluation of replication strategies is conducted. In the first approach, multiple copies of data are maintained, and in case of failure, data is retrieved from these copies. Data and error handling information are kept at cores in the second approach. The fault tolerance and monitoring are achieved with the machine without conserving energy. According to Yanchao Zhu et al. [93], replication is frequently employed to assess computational correctness; however, it necessitates replacing damaged instruments with redundant materials for node failure defects. Capability to be placed and tolerance to flaws are two aspects [94] that are inextricably intertwined. According to Setlur et al. [95], replication makes an exact clone of a mission. They both have the same obligations and, as a result, can support suitable parallel systems to operate both in parallel, reducing operational costs. The quantity of replication is computed from accurately recorded information supplied to each task, depending on how much they affect the application period (had they been repeated). Based on the number of replicas, Gorbenko et al. [96] describe that it defines a significant trade-off between fault tolerance and energy utilization. Replicated nodes also help network efficiency by coordinating user requests. Guedes et al. [97] proposed that one of a job's duplicates might finish before the earliest one, shortening the duration.

Retry

According to Ledmi et al. [98], This is the most successful error restoration strategy. We expect that the causes of the errors will be discovered in subsequent retries. According to Kumari and Kaur [99], a project is carried out indefinitely until it is done. The same approach is used to retry the failed or incompetent mission. In such systems, retry was employed to manage problems. Mukwevho and Celik [1] suggested that the retry approach works by repeatedly receiving a failed query on the same asset. The device's independence provides a base for the Retry capability. Independence can be defined as a feature that assures that, in the absence of repeated requests and errors, the results received from one app are mainly identical to those obtained from other applications. Alshayeji [100] suggested that the unsuccessful process is retried on the cloud resource itself, and Retry has significant overhead, poor performance, and is inefficient. It is appropriate for rebuilding one channel again, according to Madani and Jamali [33]. Retry is executed in a virtual machine environment, with network host failure as the fault detection technique.

Job Resubmission

In the case of a failed job, job resubmission entails resubmitting the failed task to a different resource or the same resource. If a structure detects a failed errand, the unsuccessful endeavour is resubmitted to another. According to Kochhar et al. [101], the technique for task resubmission is widely used in fault tolerance. When a mission fails, we resend the task to the same or a different asset. According to Bukhari et al. [102], a high accuracy rate minimizes the chance of operation errors and shortens the time required to complete a task. On the other hand, a lousy performance rate will increase the likelihood of operational faults, influencing the time allocated to the work resubmission section to finish the defined job. According to Kumari and Ghamdan [103], employment might still be terminated if a failure to a task is discovered. According to Essa [104], the work is republished to the same or alternative operation asset at compilation time.

User-Defined Exceptions

In this approach, the user decides treatment or necessary actions if any failure arises. Mukwevho and Celik [1] proposed that a custom exception is handling several cases when software engineers embed scripts inside the program to manage specific issues throughout the operating period. The requirements for fault tolerance of modular web applications are often derived from the demands of the enterprise. Methods in which programmers inject code into the app to address various issues during debugging.

Rescue workflow

Almezeini and Hafez [114] assert that this technique disregards faulty tasks. It permits the process to continue unabated, without regard to errors. Sarmila et al. [51] argued that this method allows the process flow to continue even if the node fails until it becomes impossible to continue without catering to the failed job. According to Poola et al. [105], the recovery process strategy disregards unsuccessful activities and continues the work process indefinitely. This strategy may result in a maximum load on the client application, according to Pandita et al. [106].

Load balancing

According to Jaimeel Shah et al. [115], the primary purpose of load balancing is to distribute the load over several cloud nodes evenly. In cloud computing, the effectiveness of load balancing is measured using multiple metrics. A load balancer distributes a load over network connections on various devices or system clusters to maximize asset efficiency and reduce total response time [107]. It reduces device wait time while eliminating asset replication. This approach divides queries across servers to speed up data distribution and processing. Load balancing distributes device load to improve system performance. Load balancing needs to spread the maximum load over various computer systems and clusters, such as servers, network connections, discs, CPUs, etc. Shahbaz Afzal and G. Kavitha [109] described it as splitting workload in a dispersed network, such as cloud computing. Load balancing tries to improve time-sensitive metrics, including response time, execution time, and system stability; as a result, cloud performance improves. According to Syeda Gauhar Fatima et al. [110], load balancing is a mechanism employed in any device to distribute the strain among different resources. As a result, in a cloud-based design, the load must be distributed across the assets so that each support does roughly the same amount of work at any one moment. The fundamental requirement is to supply specific tactics for controlling requests to respond promptly to complaints. Cloud Load Balancers manage internet traffic by dynamically dispersing workloads and assets across several servers. They improve output, reduce reaction time, and avoid overflow. According to V. Arulkumar and N. Bhalaji [111], the primary goal of any load-balancing algorithm is load balancing. Several measures, such as turnaround time and reaction time, must be considered while constructing any algorithm.

N-Version & recovery block

According to Chinnaiah and Niranjan [116], N-version programming might benefit from data diversity. Using these techniques for data re-expression, all software editions work together to improve the efficiency. A recovery block is a typical technique for fault tolerance in the software world. According to Suruchi Talwani and Inderveer Chana [112], recovery techniques should only be used when re-creating a function or measurement at a particular time. This mechanism may self-heal using more spare components to undertake additional backup tasks. Consistency and safety are critical for the N-version programming, as according to Colman and Meixner et al. [113]. Reducing the consequences of an undesirable event after it has occurred is described as a "recovery block" (as opposed to safety, which pre-deployed unnecessary capability). With resilient techniques, a process can fulfil customer expectations even with problems, and the process can improve quickly and within a realistic period. This might be due to hardware damage, a power outage, or even the destruction of the device.

Grid induction

According to Kuan Wang et al. [127], an intelligent grid operator would provide a power supply quickly using certain decentralized systems based on the game theory. It was suggested that the multi-source power management approach might be applied practically on a hybrid energy system, as indicated by Saeed Abapour et al. [128]. According to the research, hybrid power systems increase efficiency, scalability, fault tolerance, and reliability. Some of the most well-known companies globally, such as Google and Amazon, are working to improve their fault tolerance by implementing ideas provided by Muhammad Asim Shahid et al.[58] GameDay, a computer software, was used to make this possible. GameDay is software designed to expose substantial failures to approaches for identifying defects and inter-component dependencies. Employees from all company levels must work together to solve the issue in a GameDay scenario. If the test is repeated and everything works flawlessly, the GameDay activity has been deemed a success.

Fault tolerance using Machine learning

An artificial intelligence technique known as machine learning is used in this case in which a system works as a human automatically without any human involvement. Machine learning techniques can even be used for fault tolerance to enhance a system's reliability. Machine learning focuses on developing software that can access data and then even learn from that data. Normally, machine learning techniques are deployed in proactive approaches where failure prediction is made before the occurrence based on previous system data. According to Mehmet Demirci [122], the reinforcement learning technique examines the health of virtual machines in a cloud computing environment. Each virtual machine acts as an independent participant in the learning activity by employing reinforcement learning. Fault tolerance in a parallel machine learning system is done by regularly monitoring the input variables in the parameter server, as suggested by Kuo Zhang et al. [123]. After a glitch, the system is reset to the latest checkpoint. Despite a high volume of recalls and network activity, the parallel machine learning system does not perform checkpoints at each version. A similar machine learning system does not take a checkpoint at each performance, owing to the massive recollection and network strain that checkpointing causes. The checkpointing time window can be changed. According to Nagdev Amruthnath and Tarun Gupta [124], defect detection is one of the well-known concepts in fore-casting faults. Rapid error detection has the potential to avert systemic system breakdowns. *This operation is divided into numerous processes and one of the most current research studies, such as quantitative model-based approaches, qualitative model-based methods, and process history-based methods. According to Bahrudin Hrnjica and Ali Danandeh Mehr* [126]*, unsupervised learning is a way of detecting patterns in data without a predefined output (unmarked data sets). The proper outcome cannot be estimated because no goal function is defined in unsupervised learning. Algorithms have opted to rely on their skills to extract as much information as possible from the data. Dongqiu Xing* [125] *proposed deep learning algorithms for quickly discovering multi-criteria faults in analysing complex industrial facilities. Some of the AI-related approaches can be used in fault tolerance.*

A neural network is a supervised approach in which a computer is modelled to work as a human with the help of nonlinear processing elements.

KNN (K -Nearest Neighbor) in this technique, the cases are stored, and new possibilities are categorized based on similarities known as a distance function. Using this supervised learning technique, we can solve regression and classification predictive problems as it needs less calculation time and predictive power.

Reinforcement learning provides the facility to determine the excellent behavior in a specific situation to maximize the performance. It learns from the environment as it has two principal components: agent and background.

Clustering is used in parallel computing as well as in cloud computing. The tightly or loosely connected computer works as a single unit are called a cluster. An increase in the number of nodes in parallel computing increases the probability of failure, so various clustering approaches become dependable and resilient using fault tolerance strategies.

References

[1] Mukwevho MA, Celik T. Toward a smart cloud: A review of fault- tolerance methods in cloud systems. IEEE Trans Serv Comput 2018. https://doi.org/10.1109/TSC.2018.2816644.

[2] B. Talwar, S. Bharany, A. Arora, Proactive Detection of Deteriorating Node Based Migration For Energy-Aware Fault Tolerance, 22, 25.

[3] Cheraghlou MN, Khadem-Zadeh A, Haghparast M. A survey of fault tolerance architecture in cloud computing. J Netw Comput Appl 2016;61:81–92. https://doi.org/10.1016/j.jnca.2015.10.004.

[4] H. Agarwal, A. Sharma, A comprehensive survey of fault tolerance techniques in cloud Computing, 2015 Intl. Conference on Computing and Network Communications (CoCoNet'15) (2015) 408–413, 10.1109/CoCoNet.2015.7411218.

[5] Amin Z, Sethi N, Singh H. Review on fault tolerance techniques in cloud computing. Int J Comput Appl 2015;116:11–7.

[6] Ataallah SMA, Nassar SM, Hemayed EE. Fault tolerance in cloud computing – Survey, 11th International Computer. Engineering Conference 2015:241–5.

[7] Saikia LP, Devi YL. Fault tolererance techniques and algorithms in cloud system. Int J Comput Sci Commun Netw 2014;4:1–8.

[8] Tchana A, Broto L, Hagimont D. Fault tolerant approaches in cloud computing infrastructures. The Eighth International Conference on Autonomic and Autonomous Systems 2012:42–8.

[9] Kumari, P., & Kaur, P. (2018). A survey of fault tolerance in cloud computing. In Journal of King Saud University – Computer and Information Sciences. Elsevier BV. 10.1016/j.jksuci.2018.09.02.

[10] Nimisha Singla, Seema Bawa "Priority Scheduling Algorithm with Fault Tolerance in Cloud Computing", International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE),3(12), December – 2013, pp. 645-652.

[12] V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, W.Ramirez, and S. Sanchez-Lopez. "Proactive vs reactive failure recovery assessment in combined Fog-to-Cloud (F2C) systems." In 2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2017, pp. 1–5.

[16] Joshi SC, Sivalingam KM. Fault tolerance mechanisms for virtual data center architectures. Photon Netw Commun 2014;28(2):154–64.

[18] Liang Luo, Wenjun Wu and Dichen Di,Fei Zhang,Yizhou Yan,Yaokuan Mao, 'A Resource Scheduling algorithm of Cloud Computing base d on Energy Efficient Optimization Methods', IEEE 978-1-4673-2154-9, Vol. 12 (2012).

[19] E. AbdElfattah, M. Elkawkagy, and A. El-Sisi, "A reactive fault tolerance approach for cloud computing," in 2017 13th International Computer Engineering Conference (ICENCO), 2017, pp. 190– 194.

[20] Yanbing Liu, Shasha Yang, Qingguo Lin, and Gyoung-Bae Kim, 'Loyalty-Based Resource Allocation Mechanism in Cloud Computing', Recent Advances in CSIE 2011, LNEE 125, PP: 233e-238.

[23] Zhongyuan Lee, Ying Wang, Wen Zhou, ' A dynamic priority scheduling algorithm on service request scheduling in cloud computing', 2011 International Conference on Electronic & Mechanical Engineering and Information Technology, IEEE 978 -1-61284-088-8, Vol. 11 (2011), PP: 4665-4669.

[24] Padmakumari P, Umamakeswari A, Akshaya M. Hybrid Fault Tolerant Scheme to Manage VM Failure in the Cloud. Indian J Sci Technol ISSN 2016;948:974–6846.

[25] Gokhroo MK, Govil MC, Pilli ES. Detecting and mitigating faults in cloud computing environment. 3rd IEEE Int Conf 2017.

[26] Win Win Naing "Fault-tolerant Management for Private Cloud System", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 1, Issue 1, May-June 2012 ISSN 2278-6856.

[27] O. Subasi, T. Martsinkevich, F. Zyulkyarov, O. Unsal, J. Labarta, and F. Cappello, "Unified fault-tolerance framework for hybrid task-parallel message-passing applications," The International Journal of High Performance Computing Applications, vol. 32, no. 5. SAGE Publications, pp. 641–657, Sep. 26, 2016. doi: 10.1177/1094342016669416.

[29] P. Jain, A dynamic process for fault tolerance techniques in cloud computing (DPFT), 21, 2019, p. 10.

[30] A.P. Pandian, R. Palanisamy, K. Ntalianis (Eds.), Proceeding of the Inter- national Conference on Computer Networks, Big Data and IoT (ICCBI – 2019), Springer International Publishing, Cham, 2020, 10.1007/978-3-030-43192-1.

[31] P. Gupta, P.K. Gupta, Trust & Fault in Multi Layered Cloud Computing Architecture, Springer International Publishing, Cham, 2020,http://dx.doi. org/10.1007/978-3-030-37319-1.

[32] S.S. Madani, S. Jamali, A comparative study of fault tolerance techniques in cloud computing, 2018, p. 9.

[33] Hasan M, Goraya MS. Fault tolerance in cloud computing environment: A systematic survey. Comput Ind 2018;99:156–72. https://doi.org/10.1016/j.compind.2018.03.027.

[34] E. AbdElfattah, M. Elkawkagy, A. El-Sisi, A reactive fault tolerance approach for cloud computing, in: 2017 13th International Computer Engineering Conference (ICENCO), IEEE, Cairo, 2017, pp. 190–194.

[35] Khaldi M, Rebbah M, Meftah B, Smail O. Fault tolerance for a scientific workflow system in a cloud computing environment. Int J Comput Appl 2020;42:705–14. https://doi.org/10.1080/1206212X.2019.1647651.

[36] S.M. Hosseini, M.G. Arani, Fault-tolerance techniques in cloud storage: A survey, IJDTA 8 (2015) 183–190,10.14257/ijdta.2015.8.4.19.
[50] K. Kotecha, V. Piuri, H.N. Shah, R. Patel (Eds.), Data Science and Intelligent Applications: Proceedings of ICDSIA 2020, Springer Singapore, Singapore, 2021,10.1007/978-981-15-4474-3.
[51] What is the difference between crash faults and byzantine faults? – Quora, 2020, https://www.quora.com/What-is-the-difference-between- Crash-Faults-and-Byzantine-Faults. (Accessed 10 September 2020).
[58] Vishwanath KV, Nagappan N. (2010). Characterizing cloud computing hardware reliability. In Proceedings of the 1st ACM symposium on cloud computing. ACM, Indianapolis. 193–204.
[73] New Fuzzy-Based Fault Tolerance Evaluation Framework for Cloud Com- puting Request PDF, ResearchGate. 10.1007/s10922-019- 09491-2.
[74] K. Khalil, O. Eldash, A. Kumar, M. Bayoumi, Self-healing hardware sys- tems: A review, Microelectron. J. 93 (2019) 104620,10. 1016/j.mejo.2019.104620.
[75] Mohammed B, Awan I, Ugail H, Younas M. Failure prediction using ma- chine learning in a virtualised HPC system and application. Clust Comput 2019;22: 471–85. https://doi.org/10.1007/s10586-019-02917-1.
[76] Battula SK, Garg S, Montgomery J, Kang B. An efficient resource monitoring service for fog computing environments. IEEE Trans Serv Comput 2020;13:709–22. https://doi.org/10.1109/TSC.2019.2962682.
[77] B. Mohammed, A framework for efficient management of fault tolerance in cloud data centres and high- performance computing systems, 192.
[78] Kumar S, Kushwaha DAS. Future of fault tolerance in cloud computing 2019;22:6.
[79] P. Marcotte, F. Gregoire, F. Petrillo, Multiple fault-tolerance mechanisms in cloud systems: A systematic review, in: 2019 IEEE International Sym- posium on Software Reliability Engineering Workshops (ISSREW), IEEE, Berlin, Germany, 2019, pp. 414–421,10.1109/ISSREW. 2019.00104.
[81] K.B. Thanh, L.P. Dieu, S.D.T. Hong, T.V. Pham, H.T. Cong, A proactive fault tolerance approach for cloud computing based on takagi-sugeno fuzzy system and simulated annealing algorithm. 12.
[82] Amoon M, El-Bahnasawy N, Sadi S, Wagdi M. On the design of reactive approach with flexible checkpoint interval to tolerate faults in cloud computing systems. J Amb Intell Hum Comput 2019;10:4567–77. https://doi.org/10.1007/s12652-018-1139-y.
[83] Fang Y, Chen Q, Xiong N. A multi-factor monitoring fault tolerance model based on a GPU cluster for big data processing. Inform Sci 2019;496:300–16. https://doi.org/10.1016/j.ins.2018.04.053.
[84] M. Nazari Cheraghlou, A. Khademzadeh, M. Haghparast, New fuzzy- based fault tolerance evaluation framework for cloud computing, J. Netw. Syst. Manage. 27 (2019) 930–948,10.1007/s10922-019-09491-2.
[85] Prathiba S, Sowvarnica S. In: Survey of failures and fault tolerance in cloud. Chennai, India: IEEE; 2017. p. 169–72. https://doi.org/10.1109/ICCCT2.2017.7972271.
[91] A. Ledmi, H. Bendjenna, S.M. Hemam, Fault tolerance in distributed systems: A survey, in: 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS), IEEE, Tebessa, 2018, pp. 1–5, 10.1109/PAIS.2018.8598484.
[93] D.K. Baruah, L. Saikia, 2015. A Review on Fault Tolerance Techniques and Algorithms in Cloud Computing Environment, Undefined. (2015)./paper/A-Review-on-Fault-Tolerance-Techniques-and-in-Cloud-Baruah- Saikia/0f00259437c7182320e0299c1f19911e2f40f5e0 (accessed March 26, 2021).
[94] D. Kochhar, A. Kumar, J. Hilda, An approach for fault tolerance in cloud computing using machine learning technique, 8.
[95] S. Bukhari, K.R. Ku-Mahamud, H. Morino, Dynamic ACO-based fault tolerance in grid computing, IJGDC 10 (2017) 117–124, 10.14257/ijgdc.2017.10.12.11.
[96] Kumari DJM. A study on fault tolerance solution. Int J Eng Res 2016;4:5.
[97] Y.M., A survey of cloud computing fault tolerance: Techniques and implementation, IJCA 138 (2016) 34–38,10.5120/ijca2016909055.
[98] D. Poola, M.A. Salehi, K. Ramamohanarao, R. Buyya, A taxonomy and survey of fault-tolerant workflow management systems in cloud and distributed computing environments, in: Software Architecture for Big Data and the Cloud, Elsevier, 2017, pp. 285–320,10. 1016/B978-0-12-805467-3.00015-6.
[99] A. Pandita, P.K. Upadhyay, N. Joshi, Fault tolerance based compara- tive analysis of scheduling algorithms in cloud computing, in: 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), IEEE, Kottayam, India, 2018, pp. 1–6,http://dx.doi. org/10.1109/ICCSDET.2018.8821216.
[100] Kumar P, Kumar R. Issues and challenges of load balancing techniques in cloud computing: A survey. ACM Comput Surv 2019;51:1–35. https://doi.org/10.1145/3281010.
[101] I.J. of S.M., E.R. Ijsmer, A review: Load balancing algorithm using cloud analyst environment, 2020,https://www.academia.edu/30460499/A_ Review_Load_ balancing_Algorithm_Using_Cloud_Analyst_Environment. (Accessed 10 September 2020).
[102] Afzal S, Kavitha G. Load balancing in cloud computing – A hierarchical taxonomical classification. J Cloud Comput 2019;8:22. https://doi.org/10.1186/s13677-019-0146-7.
[103] S.G. Fatima, S.K. Fatima, S.A. Sattar, N.A. Khan, S. Adil, Cloud computing and load balancing, IJARET 10 (2019)https://doi.org/10.34218/IJARET.10.2.2019.019.
[104] Arulkumar V, Bhalaji N. Performance analysis of nature inspired load balancing algorithm in cloud environment. J Amb Intell Hum Comput 2020. https://doi.org/10.1007/s12652-019-01655-x.
[105] S. Talwani, I. Chana, Fault Tolerance Techniques for Scientific Applications in Cloud, 2017, p. 5.
[106] C. Colman-Meixner, C. Develder, M. Tornatore, B. Mukherjee, A survey on resiliency techniques in cloud computing infrastructures and applications, IEEE Commun. Surv. Tutor. 18 (2016) 2244–2281,10. 1109/COMST.2016.2531104.
[107] Almezeini N, Hafez A. In: An enhanced workflow scheduling algorithm in cloud computing, in. Rome, Italy: Science and and Technology Publications; 2016. p. 67–73. https://doi.org/10.5220/0005908300670073.
[109] Chinnaiah MR, Niranjan N. Fault tolerant software systems using soft- ware configurations for cloud computing. J Cloud Comput 2018;7:3. https://doi.org/10.1186/s13677-018-0104-9.
[110] Rahul Ghosh, Francesco Longo, Vijay K. Naik, and Kishor S. Trivedi. 2013. Modeling and performance analysis of large scaleIaaS clouds. Future Generation Computer Systems 29, 5 (2013), 1216–1234. DOI: 10.1016/j.future.2012.06.
[111] Wang J, Luo W, Liang W, Liu X, Dong X. Locally Minimum Storage Regenerating Codes in Distributed Cloud Storage Systems. IEEE Access 2017;no. November: 82–91.
[112] Xia Y, Tsugawa M, Fortes JAB, Chen S. Large-Scale VM placement with disk anti-colocation constraints using hierarchical decomposition and mixed integer programming. IEEE Trans Parallel Distrib Syst 2017;28(5):1361–74.
[113] Sun D, Chang G, Miao C, Wang X. Analyzing, modeling and evaluating dynamic adaptive fault tolerance strategies in cloud computing environments. The Journal of Supercomputing Mar. 2013;66(1):193–228.
[114] M.K. Edemo, Developing fault tolerance architecture for real-time systems of cloud computing, 94.
[115] Demirci M. In: A survey of machine learning applications for energy-efficient resource management in cloud computing environments. Miami, FL, USA: IEEE; 2015. p. 1185–90. https://doi.org/10.1109/ICMLA.2015.205.
[116] K. Zhang, S. Alqahtani, M. Demirbas, A comparison of distributed machine learning platforms, in: 2017 26th International Conference on Computer Communication and Networks (ICCCN), IEEE, Vancouver, BC, Canada, 2017, pp. 1–9,10.1109/ICCCN.2017.8038464.
[117] N. Amruthnath, T. Gupta, A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance, in: 2018 5th International Conference on Industrial Engineering and Applications (ICIEA), IEEE, Singapore, 2018, pp. 355–361,http://dx.doi. org/10.1109/IEA.2018.8387124.
[118] Xing D, Chen R, Qi L, Zhao J, Wang Y. Multi-source fault identification based on combined deep learning. MATEC Web Conf 2020;309:03037. https://doi.org/10.1051/matecconf/202030903037.
[119] F. Al-Turjman (Ed.), Smart Cities Performability, Cognition, & Security, Springer International Publishing, Cham, 2020,10.1007/978-3-030-14718-1.
[120] K. Wang, J. Wu, X. Zheng, A. Jolfaei, J. Li, D. Yu, Leveraging energy function virtualization with game theory for fault-tolerant smart grid, IEEE Trans. Ind. Inf. (2020) 1,10.1109/TII.2020.2971584.
[121] Abapour S, Nazari-Heris M, Mohammadi-Ivatloo B, Tarafdar M. Hagh, Game theory approaches for the solution of power system problems: A comprehensive review. Arch Comput Methods Eng 2020;27:81–103. https://doi.org/10.1007/s11831-018-9299-7.
[122] S. Bharany et al., "Energy-Efficient Clustering Scheme for Flying Ad-Hoc Networks Using an Optimized LEACH Protocol," Energies, vol. 14, no. 19. MDPI AG, p. 6016, Sep. 22, 2021. doi: 10.3390/en1419.
[123] Safara F, Souri A, Baker T, Al Ridhawi I, Aloqaily M. PriNergy: A priority-based energy-efficient routing method for IoT systems. J Supercomput 2020;76: 8609–26.
[124] Asadi, A.N.; Azgomi, M.A.; Entezari-Maleki, R. Analytical evaluation of resource allocation algorithms and process migration methods in virtualized systems. Sustain. Comput. Inform. Syst. 2020, 25, 100370.
[125] Welsh T, Benkhelifa E. On Resilience in Cloud Computing: A survey of techniques across the Cloud Domain. ACM Comput Surv (CSUR) 2020;53:1–36.
[126] Yuan, H.; Liu, H.; Bi, J.; Zhou, M. Revenue and energy cost-optimized biobjective task scheduling for green cloud data centers. IEEE Trans. Autom. Sci. Eng. 2020, 18, 817–830.
[127] Haseeb-Ur-Rehman RMA, Liaqat M, Aman AHM, Hamid SHA, Ali RL, Shuja J, et al. Sensor Cloud Frameworks: State-of-the-Art, Taxonomy, and Research Issues. IEEE Sens J 2021;21:22347–70.
[128] Alaei, M.; Khorsand, R.; Ramezanpour, M. An adaptive fault detector strategy for scientific workflow scheduling based on improved differential evolution algorithm in cloud. Appl. Soft Comput. 2021, 99, 106895.
[144] Bharany S, Sharma S, Khalaf OI, Abdulsahib GM, Al Humaimeedy AS, Aldhyani THH, et al. A Systematic Survey on Energy-Efficient Techniques in Sustainable Cloud Computing. Sustainability 2022;14:6256. https://doi.org/10.3390/su14106256.