Received May 5, 2022, accepted June 1, 2022, date of publication June 10, 2022, date of current version June 20, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3182211

Fault-Tolerance in the Scope of Cloud Computing

A. U. REHMAN, RUI L. AGUIAR, (Senior Member, IEEE), AND JOÃO PAULO BARRACA
Instituto de Telecomunicações, 3810-193 Aveiro, Portugal
Departamento de Eletrónica, Telecomunicações e Informática (DETI), Universidade de Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal
Corresponding author: A. U. Rehman (asad.rehman@av.it.pt)

This work was supported in part by the Fundação para a Ciência e a Tecnologia/Ministério da Educação e Ciência (FCT/MEC) through national funds and when applicable co-funded by Fundo Europeu de Desenvolvimento Regional (FEDER)-PT2020 Partnership Agreement under Project UIDB/50008/2020-UIDP/50008/2020, and in part by FCT under Grant PD/BD/113822/2015.

ABSTRACT Fault-tolerance methods are required to ensure high availability and high reliability in cloud computing environments. In this survey, we address fault-tolerance in the scope of cloud computing. Recently, cloud computing-based environments have presented new challenges to support fault-tolerance and opened new paths to develop novel strategies, architectures, and standards. We provide a detailed background of cloud computing to establish a comprehensive understanding of the subject, from basic to advanced. We then highlight fault-tolerance components and system-level metrics and identify the needs and applications of fault-tolerance in cloud computing. Furthermore, we discuss state-of-the-art proactive and reactive approaches to cloud computing fault-tolerance. We further structure and discuss current research efforts on cloud computing fault-tolerance architectures and frameworks. Finally, we conclude by enumerating future research directions specific to cloud computing fault-tolerance development.

INDEX TERMS Cloud computing, fault-tolerance, system-level metrics, component-level metrics, fault-tolerance frameworks, fog computing, 5G networks, edge computing, emerging cloud technologies.

I. INTRODUCTION

Cloud computing can dramatically simplify resource sharing and the cost of computation. The National Institute of Standards and Technology (a physical sciences laboratory and non-regulatory agency of the United States Department of Commerce) define cloud computing as follows: “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [1].

Cloud computing offers multiple benefits and features for enterprises to develop their cloud platform according to their business models and customer needs. Cloud computing supports multiple service models such as Infrastructure as a Service (IaaS), Application as a Service (AaaS) [2]. These services and Platform as a Service (PaaS) can be deployed as public, private, hybrid, and community clouds, with strategies that evolved over the past decade in order to generate new revenue streams. The main salient features and advantages [3]–[6] are summarized below.

The associate editor coordinating the review of this manuscript and approving it for publication was Ali Kashif Bashir.

- Multi-tenancy: Cloud computing is designed to support a multi-tenant model in which multiple end-users can share the same application over the same infrastructure while retaining their own privacy and security [7], [8].
- Resource Pooling: Computing resources are pooled to serve multiple end-users using a multi-tenant model. Resources are assigned and reassigned according to multiple end-users demands.
- Dynamic Resource Provisioning: Computing resources are created and terminated on the fly considering the current demand of end-users instead of provisioning resources using traditional peak-load demand. Allocating resources through dynamic provisioning can lower the operating cost.
- On-demand Self-service: End-users can provision their cloud computing resources without any human interaction through a web-based self-service portal.
- Elasticity and Scalability: Resources are provisioned and released on-demand, in some cases automatically to ensure that the application has exact capacity at any given point in time to scale in and scale out as per the end-users demand.

These features provide with a well-known set of advantages as follows:

A. ADVANTAGES OF CLOUD COMPUTING

1) COST SAVING

Cloud computing reduces cost for organizations by providing infrastructural resources using pay as you go pricing models [9]. End users/organizations do not need to pay upfront costs for infrastructure, and start gaining benefits from cloud computing by simply renting the resource from cloud providers according to their own needs.

2) DISASTER RECOVERY

Usually, in cloud computing, service providers implement their Information and Communications Technology (ICT) infrastructure spanning multiple geographical locations in order to effectively recover from natural or human-induced disasters and ensure the continuity of the end-users services and provide quick data recovery.

3) SUSTAINABILITY

Hosting applications on the cloud is more environmentally friendly than hosting them on-premises. Recent studies show that the adoption of cloud and other virtual data options reduces carbon footprint and improves energy efficiency.

4) EASY BACKUP AND DATA RESTORATION

Generally, on-premises data storage capacity is low and storing data can be a time taking process while in cloud computing data storage capacity with high data stored on the cloud is easy to back up and recover.

5) AUTOMATIC SOFTWARE INTEGRATION

In the cloud, applications are updated and software is integrated automatically, instead of on-premises time-consuming manual updates.

As a consequence cloud computing offers compelling features and provides considerable opportunities [10], [11] for information technology-based businesses and cloud infrastructure owners (cloud service/solution providers). However, the development of fault-tolerance in the scope of cloud computing is still in its infancy and must be addressed thoroughly.

Cloud computing architectures are complex and consist of multiple interconnected servers in data centers [12]–[14]. Designing fault avoidance and prevention approaches [15] are necessary to resist changes that occur due to hardware and software failures. We call fault-tolerance the ability of a system to provide services even in the presence of faults [16].

Cloud computing systems are implemented in a centralized, decentralized, or distributed fashion (typical complex computing infrastructure). Due to their inherent complexity, cloud computing systems are prone to three major issues: Failures, Errors, and Faults.

"A failure happens if a system is unable to implement the specified function appropriately. An error is caused because one or more of the sequences of system states deviate from the specified sequence, and can cause service disruption. A fault is the hypothesized cause of an error, for instance, a software bug, human-made error, or hardware power failure" [17]. Faults can cause errors and lead to single or multiple failures [18]. Relationship between fault, error, and failure [19] is depicted in Fig. 1 [20].

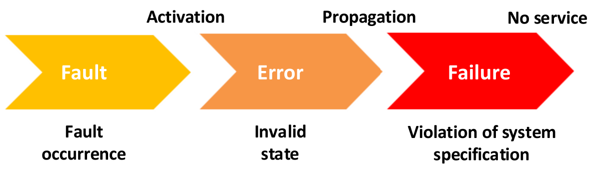

FIGURE 1. Relationship: Fault, error, and failure.

Multiple system-level tools are being used for implementing fault-tolerance techniques in environments, such as HAProxy, SHelp, Assure, Hadoop, and Amazon Elastic Compute Cloud (EC2) [21], [22].

The rest of the paper is structured as follows. Section II discusses previous studies on fault-tolerance in cloud computing, and identifies differentiating factors of this work. Section III provides detailed background on cloud computing and its related concepts. Section IV discusses fault-tolerance concepts in cloud computing, and an analysis of reactive and proactive fault-tolerance concepts. Section V discusses existing proactive and reactive fault-tolerance architectures and state-of-the-art of cloud computing research. Section VI discusses respective cloud computing fault-tolerance challenges and future research directions. Section VII summarizes an overall discussion and analysis on cloud computing fault-tolerance. Section VIII concludes the paper.

II. RELATED WORK

Some previous surveys have explored fault-tolerance in cloud computing [23]–[25]. We can summarize these efforts as follows.

Cheraghlou *et al.* [23] pointed out the methods of creating the capacity of fault-tolerance in cloud computing and briefly discuss architectures for the production of such capacity. They compared architectures already employed in terms of their methods of failure detection and recovery and implemented policies. They also briefly discussed general cloud computing models, their challenges, and provide an overview of fault-tolerance techniques. They further described existing architecture that provides fault-tolerance in cloud computing and broadly divided them into proactive and reactive groups. Finally, they evaluated and discussed existing cloud fault-tolerance architectures.

Hasan and Goraya [24] presented a systematic study of fault-tolerance in a cloud computing environment. They classify various types of faults in the cloud and briefly discuss proactive and reactive fault-tolerance approaches. Next, they present an in-depth analysis of fault-tolerance frameworks. Furthermore, they provide their own graphical representation of a quantified view of targeted fault categories and fault-tolerance methods utilized in the previous studies. Finally, they outline future research directions concerning cloud fault-tolerance.

TABLE 1. Scope and technical contributions of previous and this work.

Related Work	Scope of the Work	Main Technical Contributions
A survey of fault-tolerance architecture in cloud computing [23].	Provide a comprehensive review of existing fault-tolerance architectures in cloud computing and discuss briefly cloud computing models and fault-tolerance techniques.	Presented comparative analysis of existing fault-tolerance architecture.
Fault-tolerance in cloud computing environment: A systematic survey [24].	Provide a systematic literature review of fault-tolerance in cloud computing by analyzing existing proactive and reactive fault-tolerance approaches, fault classification in clouds, and fault-tolerance frameworks.	Presented an in-depth analysis of fault-tolerance frameworks and outline future research directions based on proactive and reactive fault-tolerance approaches.
A survey of fault-tolerance in cloud computing [25].	Provide an overview of cloud computing and other miscellaneous fault-tolerance approaches and integration of fault-tolerance approaches in cloud-based applications.	Studied traditional fault-tolerance approaches and analyze their connections with distributed and cloud computing environments. Outlined and discussed future research direction in terms of block-chain, distributed system, and emphasizing issues related to performance.
Fault-tolerance in the scope of cloud computing (this work).	Differentiate by discussing needs and applications for fault-tolerant computing, basic topics, components, and system-level metrics in the context of fault-tolerance support in cloud. Furthermore, previous studies focus more on reactive fault-tolerance frameworks and only briefly touched the proactive fault-tolerance frameworks. However, in this work, we discuss research efforts focusing on proactive as well as reactive frameworks in cloud computing.	Organized a concise, and complete approach to cloud computing fault-tolerance and discussed state-of-the-art research efforts addressing cloud computing fault-tolerance. Highlighted a comparison of fault-tolerance and tools used to overcome such faults and outlined important future research directions specific to cloud computing fault-tolerance developments.

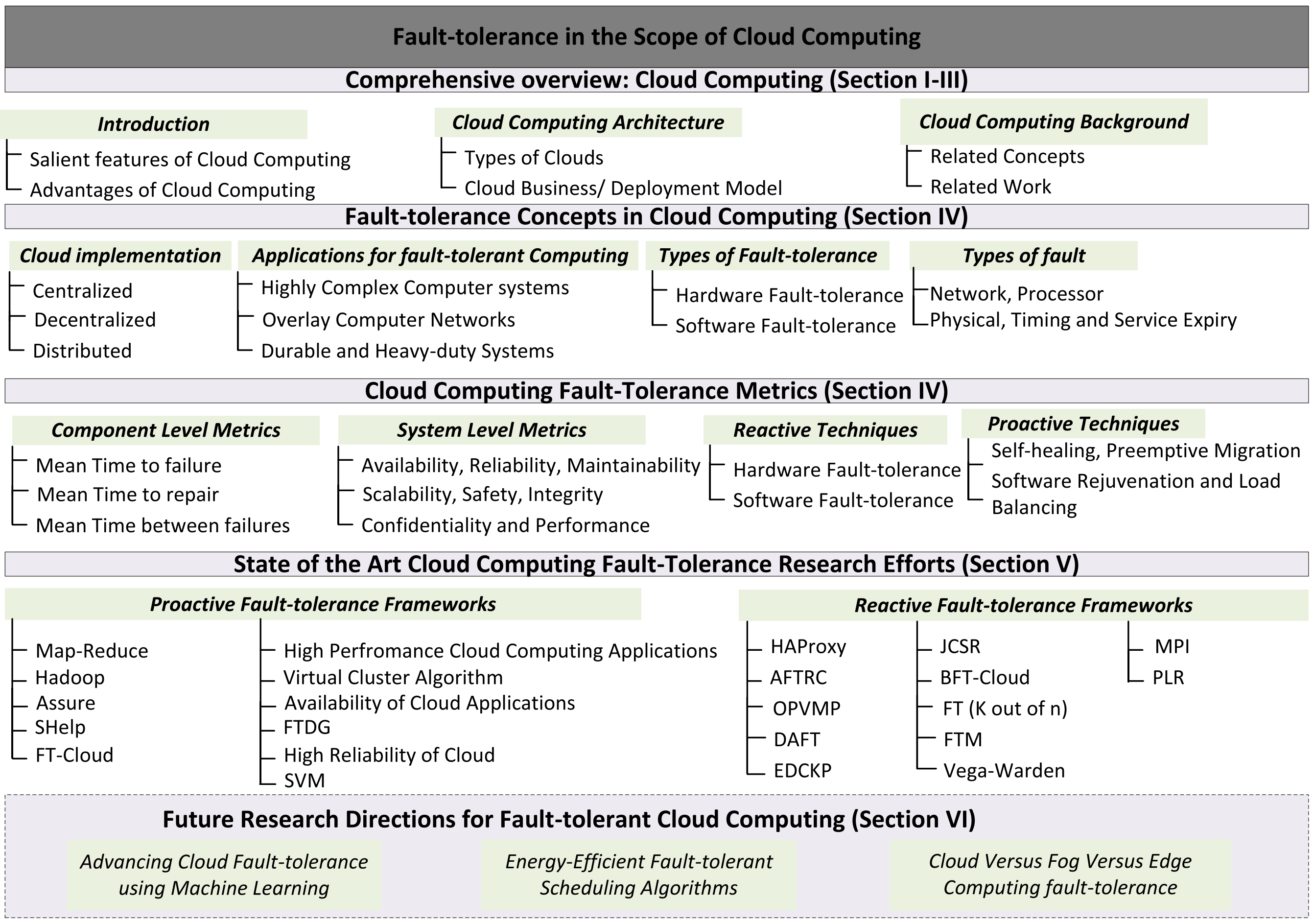
Fault-tolerance in the Scope of Cloud Computing
Comprehensive overview: Cloud Computing (Section I-III)
Introduction
Salient features of Cloud Computing
Advantages of Cloud Computing
Cloud Computing Architecture
Types of Clouds
Cloud Business/ Deployment Model
Cloud Computing Background
Related Concepts
Related Work
Fault-tolerance Concepts in Cloud Computing (Section IV)
Cloud implementation
Centralized
Decentralized
Distributed
Applications for fault-tolerant Computing
Highly Complex Computer systems
Overlay Computer Networks
Durable and Heavy-duty Systems
Types of Fault-tolerance
Hardware Fault-tolerance
Software Fault-tolerance
Types of fault
Network, Processor
Physical, Timing and Service Expiry
Cloud Computing Fault-Tolerance Metrics (Section IV)
Component Level Metrics
Mean Time to failure
Mean Time to repair
Mean Time between failures
System Level Metrics
Availability, Reliability, Maintainability
Scalability, Safety, Integrity
Confidentiality and Performance
Reactive Techniques
Hardware Fault-tolerance
Software Fault-tolerance
Proactive Techniques
Self-healing, Preemptive Migration
Software Rejuvenation and Load Balancing
State of the Art Cloud Computing Fault-Tolerance Research Efforts (Section V)
Proactive Fault-tolerance Frameworks
Map-Reduce
Hadoop
Assure
SHelp
FT-Cloud
High Perfromance Cloud Computing Applications
Virtual Cluster Algorithm
Availability of Cloud Applications
FTDG
High Reliability of Cloud
SVM
Reactive Fault-tolerance Frameworks
HAProxy
AFTRC
OPVMP
DAFT
EDCKP
JCSR
BFT-Cloud
FT (K out of n)
FTM
Vega-Warden
MPI
PLR
Future Research Directions for Fault-tolerant Cloud Computing (Section VI)
Advancing Cloud Fault-tolerance using Machine Learning
Energy-Efficient Fault-tolerant Scheduling Algorithms
Cloud Versus Fog Versus Edge Computing fault-tolerance

FIGURE 2. Condensed structure of this survey.

Kumari and Kaur [25] also addressed fault-tolerance in cloud computing and provided detailed backgrounds of cloud computing in terms of cloud computing infrastructure, data-center system model, deployment, and service models, and analyzed their connections in distributed computing environments. Furthermore, they briefly discussed existing models and frameworks based on proactive and reactive fault-tolerance approaches and touched on security aspects in fault-tolerance. Finally, discuss future research direction in terms of block-chain, distributed systems, and performance issues.

A. CONTRIBUTION AND SCOPE OF THIS SURVEY

However, these previous studies have not discussed the needs and applications for fault-tolerant computing, basic topics, components, and system-level metrics in the context of fault-tolerance in the cloud. This survey concisely addresses fault-tolerance specific to cloud computing. Furthermore, Table 1 locates the present work in the context of other technical contributions. We can distinguish our contribution in this paper in comparison to the other related work as follows:

- We organize state-of-the-art research efforts addressing cloud computing fault-tolerance based on reactive and proactive approaches used to ensure cloud computing fault-tolerance.
- We linked and discuss components and system-level metrics in the context of fault-tolerance in the cloud comprehensively, which was not addressed before.
- We structure and compare state-of-the-art research efforts addressing fault-tolerance of existing cloud computing architecture/frameworks based on reactive and proactive techniques.
- We outline important future research directions from the perspective of cloud versus fog versus edge computing, advancing machine learning tools, and energy-efficient fault-tolerant programming.

The condensed structure of this survey is summarized in Fig. 2. In this paper, we presented an organized survey of fault-tolerance in the scope of cloud computing. We overview fault-tolerance techniques and discuss the taxonomy of faults. Furthermore, we reviewed over 100 recent state-of-the-art studies to cover fundamental as well as advanced topics from literature related to fault-tolerance in the cloud to clarify, classify, analyzes, and discuss fault-tolerance in cloud computing.

III. CLOUD COMPUTING: BACKGROUND AND RELATED CONCEPTS

In this section, we provide a brief background on the concepts of cloud computing. Furthermore, we discuss centralized, decentralized, and distributed systems in the context of implementing a cloud architecture on top of these systems. Generic research challenges of cloud computing are discussed in [26].

A. CLOUD COMPUTING OVERVIEW

Cloud computing aims to provide Internet-based computing services from the pool of shared resources and enable on-demand seamless data processing and resource sharing to the computer and other devices with greater cost reduction, flexibility, and elasticity. Typically, cloud computing consists of four main elements, classified as client, data center, distributed servers, and Virtualization/Virtual Machines.

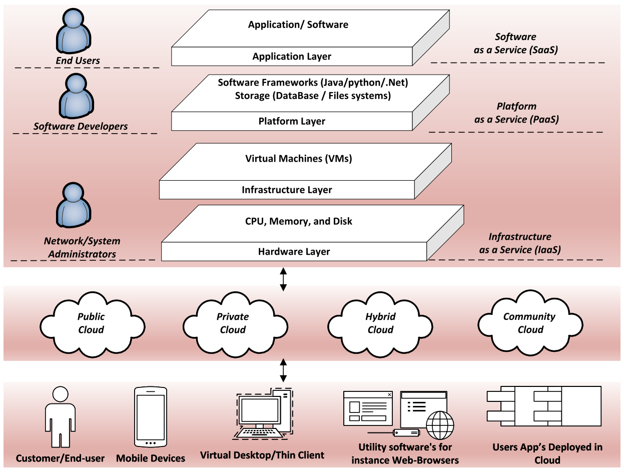
End Users
Application/ Software
Application Layer
Software as a Service (SaaS)
Software Developers
Software Frameworks (Java/python/.Net)
Storage (DataBase / Files systems)
Platform Layer
Platform as a Service (PaaS)
Virtual Machines (VMs)
Infrastructure Layer
Network/System Administrators
CPU, Memory, and Disk
Hardware Layer
Infrastructure as a Service (IaaS)
Public Cloud
Private Cloud
Hybrid Cloud
Community Cloud
Customer/End-user
Mobile Devices
Virtual Desktop/Thin Client
Utility software's for instance Web-Browsers
Users App's Deployed in Cloud

FIGURE 3. **Cloud computing architecture.**

- Clients: End-user devices such as a computer, laptop, mobile phone, and tablet are being used for exchanging information on clouds.
- Data Center: Collection of servers (ICT infrastructure) where cloud applications and services are hosted by the service provider.
- Distributed Servers: Service provider Servers located in different geographical locations in order to provide resilient, secure, and high availability to end-users.
- Virtualization/Virtual Machines: Virtualization is the technology that provides an abstracted view of physical resources such as a server, storage, and networking.

A simplified cloud computing architecture is illustrated in Fig. 3. A generic cloud computing architecture can be divided into four layers: the hardware layer, the infrastructure layer, the platform, and the application layer. Furthermore, the cloud offers services that can be grouped into three business models, as discussed, IaaS, PaaS, and SaaS. In addition to that, these services are deployed using different types of clouds namely: Public, Private, Hybrid, and Community clouds. We describe briefly each of them as follows:

1) HARDWARE LAYER

This layer is the combination of physical hardware, including servers, routers, switches, power, and cooling system. These physical resources of the cloud, (i.e., hardware layer) are typically deployed in data centers, interconnection systems.

2) INFRASTRUCTURE LAYER

This layer is an abstracted view of the hardware layer. Generally, a hypervisor provides an abstraction to create a virtual environment over the underlying infrastructure.

3) PLATFORM LAYER

This layer is built on top of the infrastructure layer. Software frameworks, such as Java, Python, and .Net, provide application programmable interfaces (APIs) support to quickly create and implement databases and storage of different web applications.

4) APPLICATION LAYER

This layer is responsible for hosting and managing actual cloud applications over the Internet, on-demand and on a subscription basis.

Cloud computing architectures are very modular and each layer is loosely coupled (isolated), which means that a wide variety of applications can be supported and better managed in the cloud compared to traditional services hosted on dedicated server farms.

B. CLOUD BUSINESS/SERVICE DEPLOYMENT MODELS

Cloud service providers model their computing services based on three primary business/service strategies (also known as cloud computing stack) namely: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

1) INFRASTRUCTURE AS A SERVICE

IaaS is a basic model of cloud computing services that allows renting infrastructural resources in terms of virtual machines or servers from the cloud service provider. Amazon web services, GoGrid, and Flexi-scale are examples of IaaS.

2) PLATFORM AS A SERVICE

In this model, an on-demand environment for developing, testing, managing, and delivering software applications using software development frameworks (Java, Python, and .Net) is provided to built web applications quickly without worrying about the configuration of the underlying infrastructure. Google app engine, Microsoft Azure, and Force.com are examples of PaaS.

3) SOFTWARE AS A SERVICE

In this model, the cloud service provider delivers on-demand software applications to the end-users. Google, Salesforce cloud-based software solutions [27], and Zoho are examples of SaaS.

C. TYPES OF CLOUD

In terms of different levels of security and management requirements, there are four different types of cloud: Public, Private, Hybrid, and Community clouds [28]. The details of these different types of clouds are as follows:

1) PUBLIC CLOUDS

The entire computing infrastructure is located on the premises of the cloud service provider. The security and data control level is lowest while using a public cloud. There is no upfront cost for hosting applications on public clouds. Examples include AWS/EC2 Amazon, Azure Microsoft, and Google cloud platform.

2) PRIVATE CLOUDS

The entire computing infrastructure is typically located or used exclusively by a single organization. The security and data control level is highest while using a private cloud. There is an upfront cost for hosting applications on private clouds. Examples include Eucalyptus Systems, OpenNebula, and OpenStack.

3) HYBRID CLOUDS

A hybrid cloud is a combination of public and private clouds, depending on the purpose and sensitivity of applications to be hosted on the cloud. Most important applications are hosted on a private cloud while other applications can be hosted on a public cloud, this allows flexibility to use both public and private cloud and some potential cost-saving.

4) COMMUNITY CLOUDS

A community cloud is a collaborative effort in which infrastructure is shared between professional communities (groups of organizations) having mutual goals and common concerns.

The cloud computing architectures, as depicted in Fig. 3, end-user, and network devices can be deployed in different cloud deployment models such as public, private, hybrid, and community, depending on the computing requirements that can support the needs of service requested by end-users. These services can be a method to achieve business goals with reduced costs. However, it is quite challenging to offer resiliency, where fault-tolerance is one of the key enablers for seamless operations in a cloud computing environment to offer a high-quality service experience to end-users [26].

IV. FAULT-TOLERANCE CONCEPTS IN CLOUD COMPUTING

In this section, we describe fault-tolerance in cloud computing and related concepts to establish a basic understanding of the subject.

Faults in the cloud environment can be classified into two main categories [14], [15]: Crash faults and Byzantine/Arbitrary faults. Crash faults can cause system fatal errors (for instance process and machine power-related failures), while Byzantine/Arbitrary faults can cause the system to deviates from normal operation [14]. Typically, in a cloud environment, the faults appear as a failure of resources, such as applications, storage, and hardware faults that affect the end-users (leading to failures and performance degradation) [29]. Faults can be transient (appears once and then disappears), intermittent (appears, disappears, and reappears with no real pattern), and permanent hardware faults (appear and persist until the faulty component is replaced or repaired). They can be of the following type considering the scope of this work:

- Physical Faults (hardware faults): Faults that are related mainly to hardware such as Central Processing Unit (CPU), memory, storage, and power failure.

- Network Faults (link faults): In cloud computing resources are accessed through a network and prone to network-related faults such as packet loss, link failure, etc.
- Processor Faults (node faults): Faults that occur due to bugs in software, resource shortage, and inefficient processing of computing resources.
- Service Expiry Faults: Faults that occur when the service time of an application is expired while the application is using it.
- Timing Faults: Faults that cause an application unable to complete within the specified period.

A. CLOUD IMPLEMENTATION: CENTRALIZED VERSUS DECENTRALIZED VERSUS DISTRIBUTED SYSTEMS

Computer networks are built to enable sharing resources through communications. Moreover, in networking, computers are connected together in different settings and organized into different systems known as Centralized, Decentralized and Distributed system as shown in the Fig. 4 [30]. Clouds are implemented using distributed, centralized and decentralized architectures. For the sake of clarity, we discuss such systems because the meaning of fault-tolerance and resilience of a system can be understood differently depending on the (small or large) scale implementation.

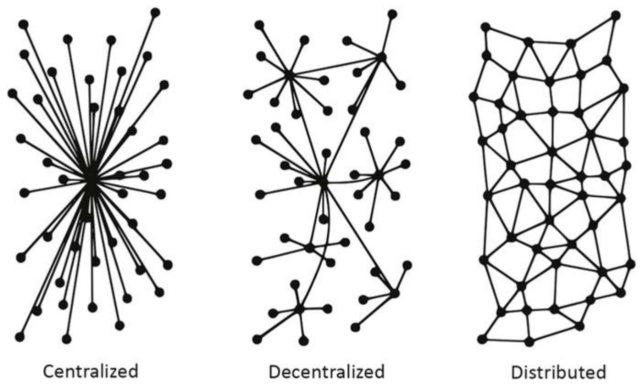

FIGURE 4. Centralized vs decentralized vs distributed systems.

Distributed systems/distributed computing can either be centralized or decentralized. To enable easy access and ensure the availability of computing resources, cloud providers offer data storage facilities in different geographical locations, although the access control is centralized. These concepts are significant in a cloud computing environment, and the principles of centralized, decentralized, and distributed systems remain applicable to emerging cloud technologies such as Blockchain [31]. Public Blockchains, including Bitcoin and Ethereum (cryptocurrencies), are both distributed (independent nodes at different geographical locations) and decentralized (data once stored cannot be altered). In the following, we briefly discuss centralized, decentralized, and distributed systems.

1) CENTRALIZED SYSTEM

Centralized systems are also known as command and control systems. In this type of system, the central command node is the node in which all decisions must be taken and another connected node has to follow strictly the commands. Moreover, a centralized system is easy to maintain. However, due to the single point of failure they can become unstable. The development of such systems is easy due to centralized control but such systems are not known for extreme scalability.

2) DECENTRALIZED SYSTEM

The decentralized system is also known as a partially distributed system. In this type of system, there is no central command node issuing instruction that other connected nodes strictly have to follow. The decision is made independently, involving multiple parties and sub-nodes to achieve system-wide goals. Moreover, the decentralized system is moderate to maintain.

3) DISTRIBUTED SYSTEM

"A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system" [32]. In this type of system, all the connected nodes are involved in the decision to achieve system-wide goals. Distributed systems are complex and hard to maintain. However, they are highly scalable and support large complexities.

B. NEEDS AND APPLICATIONS FOR FAULT-TOLERANCE COMPUTING

Generally, fault-tolerance is an essential part of the design of any communication system. However, designing needs differ across different environments and specific needs related to the environments in which the communication system is operating. For instance, mission-critical computation systems are highly critical systems and require expensive and advanced fault-tolerance support with high accuracy to guarantee the handling of mission-critical applications and situations (such as computer systems in aircraft, medical equipment systems in health and safety, E-commerce systems for financial applications, and space shuttle communications systems). All of these environments are critical and any malfunction can be catastrophic. Therefore, to avoid such damage, these systems are designed in a way that the probability of failures is very low or negligible. Such systems are highly reliable and equipped with advanced fault-tolerance support and are quite expensive systems [33]. Another example includes highly available and reliable systems, in which the availability requirement is set to a maximum of Five 9's reliability [34] which means that a system must not go down (unavailable) for more than "5 min and 15 seconds" per year. Carrier-grade networks have this strict requirement for Five 9's reliability [35].

Research in fault-tolerant cloud computing systems span a wide range of applications, from general-purpose computer systems to highly available computer, space, transportation, and military systems [33], [36], [37]. We list some applications and discussed them briefly in order to illustrate the difference between them. Each application has its fault-tolerant design considerations and challenges.

- Durable and Heavy-duty Systems.
- Highly Complex Computer Systems.
- Overlay Computer Networks Systems.

1) DURABLE AND HEAVY-DUTY SYSTEMS

Durable and heavy-duty systems are designed to operate in a harsh environment in the presence of electromagnetic disturbance and external noise. These systems usually consist of both electrical and mechanical parts. As these systems are designed to operate for a long period, repairing is hard for such systems. Therefore a completely redundant system, i.e., including all parts, are essential for the continuous operation of the system. After a certain time, these systems are replaced with a new system.

2) HIGHLY COMPLEX COMPUTER SYSTEMS

Another example includes highly complex systems, which consist of billions of equipments. Moreover, each equipment connected to these complex systems has a probability of failure. Due to the huge number of devices attached to the system, the total system failure probability can be high. However, different types of hardware, coupled with different software redundancy and replication techniques, minimize the probability of system failures. Distributed systems in computer networks are based on these complex types of systems.

3) OVERLAY COMPUTER NETWORKS SYSTEMS

These network systems use existing hardware infrastructure and map existing hardware infrastructure to form a virtualized infrastructure with the use of technologies such as Software-defined Networking (SDN), Network Functions Virtualization (NFV), and Cloud Computing, concepts often associated with Fifth-generation (5G) networks. Such systems, unlike traditional static distributed systems, are highly dynamic in nature. A vast set of fault-tolerance mechanisms are needed to support and embrace the use of these evolving technologies, high availability and reliability [38] must be guaranteed.

We have discussed above different types of systems and fault-tolerance requirements to clarify the concepts.

C. FAULT-TOLERANCE COMPONENT LEVEL METRICS

In this section, we discuss fault-tolerance traditional metrics that can be explained in cloud computing. Traditional component metrics related to fault-tolerance are [39]:

1) Mean Time to Failure (MTTF): The mean time until a component fails.
2) Mean Time to Repair (MTTR): The mean time required to repair a failed component.

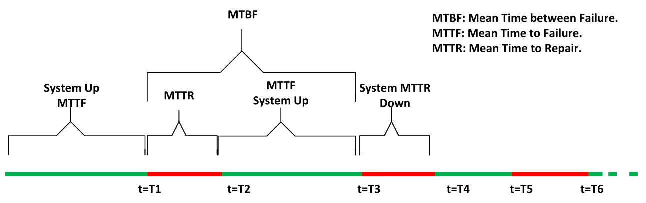
MTBF
MTBF: Mean Time between Failure.
MTTF: Mean Time to Failure.
MTTR: Mean Time to Repair.
System Up
MTTF
MTTR
MTTF
System Up
System MTTR
Down
t=T1
t=T2
t=T3
t=T4
t=T5
t=T6

FIGURE 5. **MTBF, MTTR, MTTF.**

3) Mean Time Between Failures (MTBF): It is the measure of the total time when the system is available and operating.

These three metrics can be differentiated as shown in Fig. 5.

Metrics described above can vary in real-time cloud computing environments. This infers that both software and hardware-based fault-tolerance is a must offer. Further, hardware-based fault-tolerance is critical as it provides an infrastructure to run over it; this is not too complex but costly. Although service abstraction needs software fault-tolerance and it is perceived as more critical in a real-time environment. Improper handling can cause multiple problems and service may deviate and result in failure of achieving cost reduction.

D. FAULT-TOLERANCE SYSTEM LEVEL METRICS

Several system-level metrics, considered in cloud computing to design a fault-tolerance system, are as follows [40], [41]:

1) AVAILABILITY

Availability refers to the access of service without any service deviation within a particular time. This means that resources can be accessed when needed for proper functionality. The availability is the probability of the system running without failure.

2) RELIABILITY

Reliability refers to the service continuity with correct results in a certain period. Reliability is the probability of the system running continuously without failure at the given time. The difference between reliability and availability is one provides a way to access resources without any faults and others maintain the continuity without any fault while the service remains active.

3) MAINTAINABILITY

Maintainability refers to the ease of a system to repairs and adapt modifications.

4) SCALABILITY

Scalability refers to the effectiveness of the fault-tolerant algorithm to manage increasingly more nodes without any degradation in services. A fault-tolerant solution must be scalable.

5) SAFETY

Safety refers to the property of a system that temporarily fails to operate correctly, but still nothing catastrophic happens to the users acquiring the services. A fault-tolerant solution must be safe.

6) INTEGRITY

Integrity refers to the intelligence mechanisms to prevent the system from improper alteration, and also to avoid user crash failure.

7) CONFIDENTIALITY

Confidentiality refers to the mechanisms to protect unauthorized access to information.

8) PERFORMANCE

Performance refers to metering system efficiency by analyzing and evaluating multiple system-level metrics. This evaluation significantly depends on the throughput: the greater the system throughput greater the performance.

E. TYPES OF FAULT-TOLERANCE

There are two main types of techniques used to design a fault-tolerant system. These techniques are known as hardware and software fault-tolerance [42].

1) HARDWARE FAULT-TOLERANCE

In general hardware fault-tolerance system is the designing of hardware components and their implementation such that these components can perform tasks and interacts with another component correctly within the systems. In a computing system, these components can be the CPU, memory, hard disk, and other hardware-based computing devices. Hardware-based fault-tolerance techniques in computing are useful to develop a structured computing system that not only tolerates faults but is also capable of initializing a system recovery itself. The system recovery process normally involves splitting a computing system into modules, where these modules are already backed up with protective redundancy hence providing at some extent automatic recovery in the case of failure [43].

2) SOFTWARE FAULT-TOLERANCE

Recently, the rise of virtual networks (software-defined networks) and the concept of softwarization of telecommunications systems boost software fault-tolerance. Software fault-tolerance became an area of interest for researchers. Similar to hardware-based fault-tolerance, software fault-tolerance is the continuous development process for attaining software that can tolerate software faults (programming errors), both active and passive redundancy techniques are used to design fault-tolerant software.

Fault-tolerance in cloud computing may vary in real-time scenarios, but mostly in cloud computing the data plane, control plane and specialized hardware that provides abstraction must be fault-tolerant with built-in self-recovery, self-healing mechanisms to guarantee the quality of service with scalability. This infers that both software and hardware-based fault-tolerance is a must offer. Further, hardware-based fault-tolerance is critical as it provides an infrastructure to run over it; this is not too complex, but is usually costly. Although, service abstraction needs software fault-tolerance and it is often perceived as more critical in a real-time environment without this a service may deviate and fail in achieving cost reduction.

Broadly, the two main approaches used to implement fault-tolerance are Recovery and Redundancy [44]. i) In Recovery, the system state is restored (rollback) to a predefined checkpoint. ii) In Redundancy, the hardware, software, and computing components are replicated (extra components are added to the system for backup purposes). Both the Recovery and Redundancy approaches can impact reactive and proactive policies.

The next section discusses and compares cloud computing fault tolerance based on reactive and proactive policies.

F. FAULT-TOLERANCE APPROACHES THROUGH REACTIVE TECHNIQUES

Reactive fault-tolerance policies are mainly developed to decrease the failures in the cloud/distributed system after the occurrence of the faults/errors. The Reactive policies used to prevent faults in cloud computing are as follows [21], [41], [45], [46], [47].

1) CHECKPOINT/RESTART

The system recovers from faults using a recent reference checkpoint instead of instantiating the task from the beginning. This policy is effective for a large application, in the case of failure because of uncompleted tasks.

2) REPLICATION

To safeguard the consistency between resources in a different cluster, a copy or replicas of the various task are stored. In the event of a crash, the identical replicas take over and therefore guarantee fault-tolerance.

3) JOB MIGRATION

This approach enables tasks to be migrated to a new machine seamlessly in case execution is not possible with a current machine. This technique is very useful in cloud computing and data center environments.

4) RETRY/TASK RE-SUBMISSION

Task submission refers to the execution of that task that failed to execute. The task is re-implemented repeatedly on the same or different resources to proceed with the execution of the failed tasks. The fault is rectified or it reaches the stage where the fault is unrepairable.

5) S-GUARD

This policy uses the rollback recovery fault-tolerance technique and can be implemented in Hadoop, and Amazon Elastic Compute Cloud. It periodically checkpoints the state of stream processing nodes and restarts failed nodes from

TABLE 2. Classification of fault-tolerance approaches and used tools.

Fault-tolerance Approach	Category	Detected types of fault	System Tools Used
Checkpoint/Restart	Reactive	Application failure	SHelp.
Replication	Reactive	Node and process failure	Amazon EC2.
Job Migration	Reactive	Node and process failure	Ha-Proxy.
Retry/Task Re-submission	Reactive	Network and Application failure	Assure and Amazon EC2.
S-Guard	Reactive	Application failure	Hadoop.
Timing check	Reactive	Application and Node failure	SHelp.
User-defined exceptional handling	Reactive	Application failure	SHelp.
Self-Healing	Proactive	Network failure	Assure.
Preemptive Migration	Proactive	Node and process failure	Ha-Proxy.
Software Rejuvenation	Proactive	Application failure	SHelp.
Load Balancing	Proactive	Application failure	SHelp.

their most recent checkpoints. SGuard is less disruptive to normal stream processing and typically leaves more resources available for normal stream processing.

6) TIMING CHECK

This is based on the idea of continuous monitoring. The state of the execution of tasks through watchdog supervision and in any case of alteration of the time-critical task can be processed to avoid failures.

7) USER-DEFINED EXCEPTIONAL HANDLING

In this, the user specifies the workflow for the treatment of failed tasks.

G. FAULT-TOLERANCE APPROACHES THROUGH PROACTIVE TECHNIQUES

Proactive fault-tolerance policies are mainly developed to predicts the failures in the cloud/distributed system before the occurrence of the faults/errors.

The Proactive policies used to prevent faults in cloud computing are as follows:

1) SELF-HEALING

Tasks are isolated on different virtual machines. This enables easier automatic fault handling. In the case of certain tasks failure, only the specific isolated task on a specific machine recover automatically, instead of affecting the operation of all virtual machines.

2) PREEMPTIVE MIGRATION

This technique checks the system for latent and dormant faults through feedback control.

3) SOFTWARE REJUVENATION

This technique checks and updates system records for any new configuration. The re-initialization depends on the set design assumption of the specific system failure and the time interval for re-initialization.

4) LOAD BALANCING

This technique is used to auto-scale the computing resources (CPU, and the load of the memory) when the certain threshold defined in the auto-scaling policy exceeds. The load balancing algorithms reallocate a load of exceeded computing resources to other computing resources that are not exceeded in order to balanced resources and efficient utilization of cloud infrastructure.

Table 2 presents a classification of reactive, proactive approaches, and system tools used in cloud computing fault-tolerance.

V. EXISTING CLOUD COMPUTING FAULT-TOLERANCE ARCHITECTURAL FRAMEWORKS

In this section, we present existing fault-tolerance architectures developed for cloud computing. The classification of these architectures as per fault-tolerance policy is depicted in Fig. 6. In cloud computing, fault-tolerance architectures are based on proactive and reactive policies and with the combination of more than one fault-tolerance approach to devise error detection and recovery mandatory for end-to-end service delivery. For instance, MapReduce and FT Cloud are proactive policy-based architectures and HAProxy, BFT Cloud are reactive-based architectures.

Previous surveys were mostly focused on addressing reactive fault-tolerance architectures and only briefly touched on the proactive fault-tolerance architectures. Here, we discuss in detail proactive as well as reactive architectures. The simplified taxonomy of fault-tolerance architectures/frameworks is depicted in Fig. 6. Further, we made a comparison of this architecture with fault-tolerance policy and aspects of fault-tolerance covers by these architectures [23]. We provide a brief comparison based on the studied and discussed state-of-the-art research efforts and with the type of fault-tolerance support offered by these architectures.

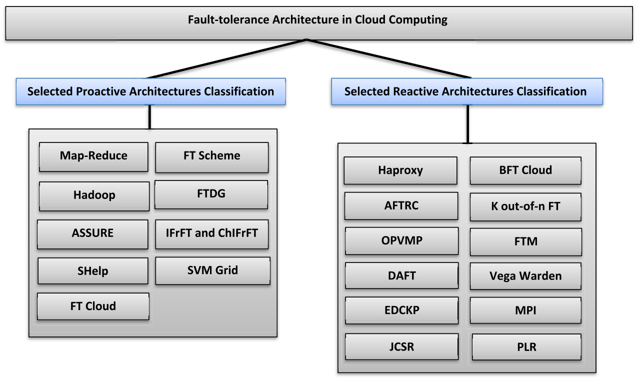

FIGURE 6. Fault-tolerance architectures in cloud computing.

In Table 3, we structure and compare the state-of-the-art research works on fault-tolerance architecture/frameworks in cloud computing based on proactive and reactive techniques.

A. RESEARCH EFFORTS FOCUSING PROACTIVE FAULT-TOLERANT ARCHITECTURES

In this section, we survey state-of-the-art proactive architectures that support fault-tolerance in cloud computing.

In MapReduce, the task is divided into sub-tasks and data is processed based on the concept of parallel computing [48]. In big data processing, high availability is a key requirement [49]. Therefore, as MapReduce architecture aims to provide high availability, a proactive policy is utilized because these techniques meet availability requirements much better than a reactive approach [50]. Fault-tolerant techniques, self-healing, and preemptive migration are exploited in MapReduce architecture.

Hadoop, based on the MapReduce architecture, has been implemented successfully by Yahoo, Amazon, IBM, and Google to process and analyze big data in data center environments [51]. Hadoop is an open-source software framework developed for storage and large-scale data processing in cloud environments [52]. It is designed to detect and handle application layer failures in cloud systems.

Sidiroglou *et al.* [53] have presented Assure, a system tool that offers automatically self-healing software using rescue points that recover systems from unknown failures. It uses lightweight instrumentation mechanisms to monitor and triage the system for fault analysis. When a fault is detected, Assure restores execution to an appropriate rescue point, and provides recovery from software failures in server applications. However, Assure does not assign weights to rescue points, and selecting an appropriate point contributes to overheads which can slow down system recovery.

Chen *et al.* [54] presented SHelp, a tool that can provide automatic self-healing for multiple application instances in a cloud-based system. SHelp followed the Assure framework and improved the earlier proposed work by assigning weights to rescue points. When weights are assigned, searching for and selecting an appropriate rescue point can be done quickly in contrast with Assure, which uses additional overheads to search and assign rescue points. SHelp handles software failures in the framework of virtual machines.

Another proactive framework named FTCloud, proposed by [55], uses a component invocation structure to identify the significant components and select optimal fault-tolerance strategies to automatically confront faults in cloud computing. They propose two algorithms to rank application components: first, considering the invocation frequencies only; and second, considering the characteristics of application components along with the invocation frequencies. They use three replication techniques and compute failure probability, response time, and application cost based on these techniques to select the optimal fault-tolerance strategy. The FTCloud framework detects software faults with high accuracy. However, implementing it is quite complex.

Egwutuoha *et al.* [56] carried out a research study to develop a proactive fault-tolerance system to support high-performance cloud computing applications. Their proposed design consisted of three modules: i) Node monitoring—module monitoring, CPU temperature, and fan speed; ii) Fault- tolerance—the module takes necessary action in case of failure due to any adverse network behavior, and iii) Controller—the module controls the live migration of virtual machines and implements predefined fault-tolerance policies whenever a failure is detected. They argue that their developed fault-tolerance system was able to tolerate hardware faults at a low cost. However, the live migration of virtual machines and predicting node failure are complex phenomena in their developed solution.

Liu *et al.* [57] argued that the existing fault-tolerance schemes do not adequately consider coordination among virtual machines (VMs) that jointly complete a parallel application execution. To address this problem, they proposed using a virtual cluster algorithm to reduce VM resource and energy consumption in the data center. They then modeled CPU temperature to predict physical machine failure and to search for and target optimal physical machines. Their solution uses a proactively coordinated fault-tolerance (PCFT) approach, which is based on particle swarm optimization (PSO) [74], to address the coordinated fault-tolerance problem of a virtual cluster. They evaluated their approach by considering overall transmission overheads, resource consumption, and execution time for a set of parallel applications. They claimed their approach outperforms others by reducing network and resource consumption while guaranteeing cloud service reliability.

Liu and Buyya [58] proposed using a holistic software, rejuvenation-based, fault-tolerance scheme to overcome cloud applications' aging problem. The system architecture is based on four fault-tolerance entities that work together to detect an aging failure and start the rejuvenation process. The fault-tolerance modules are as follows: i) the aging failure detector monitors CPU and memory usage of the service components to detect aging failures; ii) the aging degree evaluator identifies fatal errors that tend to result in crash failure and are queued and rejuvenated, iii) a software rejuvenation manager-control software rejuvenation execution; they used a checkpoint and VM migration-based service component to avoid re-executing of aging service components to guarantee service continuity of running cloud applications; and iv) interim node when VMs are prone to failure, their working state is stored in an interim node and migrated to seamlessly maintain service communication. After the migration process is completed, VMs are rebooted to clear the aging effects. They showed that their fault-tolerance scheme improves the availability of cloud applications.

Sun *et al.* [59] proposed a fault-tolerant framework with a deadline guarantee for stream computing called FTDG. They have implemented their framework on an open-source

TABLE 3. Selected work on cloud computing fault-tolerance architectures/frameworks and their characteristics.

Cloud Computing Fault-tolerance Architectures/Frameworks	Proactive Techniques	Reactive Techniques	Purpose or Used for	Applications Developed for
MapReduce [48], [49], [50]	Self-healing and preemptive migration.	-	Big data processing.	Social networks and E-commerce e.g Facebook, Twitter and Amazon, etc.
Hadoop [51], [52]	Self-healing and preemptive migration.	-	Large-scale data processing in cloud environments.	Social networks and E-commerce e.g Yahoo, Amazon, IBM, and Google, etc.
Assure [53]	Self-healing.	-	Build to address software faults.	A tool to recover the system from unknown failures and offer automatic self-healing of software.
SHelp [54]	Self-healing.	-	Build to address software faults.	Handling failures in the framework of Virtual Machines (VMs).
FT Cloud [55]	Self-Healing.	-	Ranking framework for cloud components to enhance the reliability of cloud applications.	Cloud application designer, to design optimal fault-tolerance strategies.
FT System [56]	Self-healing and preemptive migration.	-	High-performance cloud applications.	Tolerating hardware faults at low cost in high-performance computing environments.
PCFT [57]	Self-healing and preemptive migration.	-	Data center cloud computing environment.	preventing break down of virtual machine running on physical hardware in data center environment.
FT Scheme [58]	Software rejuvenation.	-	Cloud application's aging problem.	A fault-tolerance scheme that improves the availability of cloud applications.
FTDG [59]	preemptive migration.	-	High fault-tolerance for big data stream computing environments.	Designing user applications on demand of data center requirements.
IFrFT and ChIFrFT [60]	Self-healing and preemptive migration.	-	Failure of critical configurations in configurable systems.	Fault-tolerance software configurations for cloud computing.
SVM Grid [61]	Self-healing and preemptive migration.	-	Calibrate fault predictions of the cloud.	Achieving high accuracy and stability of the cloud.
HaProxy [62]	-	Replication and job Migration.	Proxying transmission control protocol and hypertext transfer protocol based applications, offering high availability and load balancing solution.	Applications that require load balancing.
AFTRC [63]	-	Checkpoint, Replication and Job migration.	To support multi-core architectures.	Building real-time high-performance computing systems.
OPVMP [64]	-	Replication.	Improve reliability of server-based cloud applications using replication method.	Server-based cloud applications and services.
DAFT [65]	-	Checkpoint.	Crash failures in cloud.	Fault-tolerence services in cloud computing systems.
EDCKP [66]	-	Checkpoint.	To address edge switch failures.	Designing enhanced service reliability in the cloud computing environments.
JCSR [67]	-	Checkpoint.	To improve resource scheduling in the data center cloud.	Optimizing clients consistency level in the data center.
BFT-Cloud [68]	-	Replication.	Byzantine fault-tolerance problem in voluntary resource cloud.	Ensuring good performance in voluntary resource cloud.
K out-of-n FT [69]	-	Replication.	For data storage and processing in the cloud.	Providing energy-efficient and network fault-tolerance in the cloud.
FTM [70]	-	Replication, Checkpoint, and Job Migration.	Monitoring manager for cloud-based system.	Implementing full policy as per the cloud deployment model.
Vega Warden [71]	-	Replication and job Migration.	Support usability and security in cloud virtualization.	A Unifying management system to support cloud virtualization.
MPI [72]	-	Checkpoint and Job Migration.	High-Level API's development.	Parallel programming systems used in the cloud computing.
PLR [73]	-	Replication, Checkpoint and Job Migration.	To support multi-core architectures.	Designing real-time high-performance computing systems.

distributed computing platform called Strom. Their proposed system architecture is composed of four spaces as follows. i) hardware space-data centers are distributed in different geographical locations for big data stream computing; ii) storm space real-time scheduling, optimization, and fault-tolerance strategy is applied using storm platform; iii) graph space-data stream graphs are created according to the source code and based on a user profile; and iv) user-space-users can design applications and submit their data stream on-demand of data centers. Their results showed that FTDG provides a desirable trade-off between high fault-tolerance and low response time objectives set for big data stream computing environments.

Failure of critical configurations in configurable systems can have a severe impact on system reliability and performance. To address this problem, Chinnaiah *et al.* [60] proposed fault-tolerant software systems using software configurations for cloud computing. They first classified the configuration of a software system into critical and non-critical configurations based on the frequency of configuration interactions (IFrFT) and characteristics and frequency of interactions (ChIFrFT). They argued that identifying critical configurations and then developing fault-tolerance support for each configuration is the way to overcome this problem. They then proposed an algorithm that identifies optimal fault-tolerance for every critical configuration of the software system. They tested the performance of IFrFT and ChIFrFT using the file structure system. They argued that their proposed scheme achieves higher reliability and fault-tolerance software configurations for cloud computing.

Zhang *et al.* [61] argued that traditional Support Vector Machine (SVM) models used for fault-detection provide very low accuracy. To overcome this problem, they presented an online fault-detection model considering SVM Grid. In their proposed approach, the Grid method was used to enhance the input parameter to calibrate fault predictions for achieving high accuracy and stability of the cloud. They experimented with their developed fault-tolerance algorithm using Google2 application cluster data sets and illustrated in their study that their developed fault-detection model using SVM Gird provides high accuracy and reduces time costs compared with traditional approaches such as learning vector quantization based on traditional SVM.

B. RESEARCH EFFORTS FOCUSING REACTIVE FAULT-TOLERANT ARCHITECTURES

In this section, we survey state-of-the-art reactive architectures that support fault-tolerance in cloud computing.

HAProxy is an open-source tool used to build a fault-tolerant cloud system [62]. It provides an efficient and reliable solution offering high availability and load balancing of web-based applications. Job migration and replication fault-tolerance approach is being used with redundancy. HaProxy can be migrated to another server in the presence of system failures. In the event of a primary server failure, the secondary server can backup the operation of the entire system. HAProxy handles server fail-over in the cloud.

Malik and Huet [63] presented an adaptive fault-tolerance in the real-time cloud computing (AFTRC) model for the cloud computing environment. In their model, the incoming tasks are executed in a first come first serve (FCFS) manner. Tasks are executed on virtual machines that are embedded with different algorithms replicated and executed in real-time. For the verification of the result's correctness, the result produced by each algorithm is sent to the acceptance test. After verification, results are moved to the time checker (TC) to check the time-stamps. The task is sent back to the input buffer if the results are not obtained within the time limit. The task states are periodically saved in the recovery cache.

AFTRC model also contains checkpoint mechanisms to serve the client requests on a highly reliable platform. AFTRC is applicable for real-time applications and offers high accuracy. However, when the workload increases it may lead to low resource availability.

Zhou *et al.* [64] proposed an optimal redundant virtual machine placement (OPVMP) model to improve the reliability of Server-based cloud services by using a replication-based fault-tolerance method. The method used in their work employs three algorithms. The first algorithm selects a VM hosting-server potentially from a large set of servers. The second algorithm decides an optimal strategy (optimal VM placement) to place the primary and backup VMs selected by the first algorithm with ''k'' fault-tolerance assurance. Finally, the Third algorithm (using heuristic approach) is formulated for recovery strategy decisions by finding a maximum weight matching. The results obtained from this approach were compared with other models. They demonstrated that the proposed approach utilized fewer network resources in the service recovery stage.

Sun *et al.* [65] studied a dynamic Adaptive Fault-tolerance (DAFT) model using checkpoint technique in order to provide fault-tolerant services in a cloud computing environment. The measures for fault-tolerance are taken from the fault-tolerance space (a main component of the architecture). The checkpoint technique used in the study is not conventional, rather dynamic and adaptive. Thus, checkpointing interval varies corresponding to check-pointing overhead, fault-tolerance overhead, fault overhead, and failure density. Maximum recovery could be obtained. On the one hand, whenever the failure rate increases the system adapts to conditions by dynamically decreasing the checkpoint interval. On the other hand, whenever the failure rate decreases the checkpoint interval increases, due to which the overhead would be optimized. DAFT can be applied to crash failures. However, the adaptiveness of the DAFT fault-tolerance is system-based, not component-based.

Zhou *et al.* [66] proposed a model named Edge switch failure-aware check-pointing (EDCKP). This model is designed to enhance service reliability in the cloud computing system. To address the edge switch failure, a fat-tree network topology and two algorithms were proposed. The first algorithm is used to select a storage server for the checkpoint image. The second algorithm is used for the recovery server. They compared their model with the network topology-aware distributed delta checkpoint-based technique (NDCKP) model which is based on a network topology-aware distributed delta checkpoint-based technique, and with the No checkpoint-based method (NOCKP). They argue that their result illustrated that the EDCKP model performs better in terms of service reliability by reducing the execution time and by consuming fewer network resources.

Zhao *et al.* [67] studied interference-free scheduling to provide flexible reliability optimization in the cloud environment. The Joint Checkpoint Scheduling and Routing (JCSR) model was presented. In their model, they use a peer-to-peer checkpointing method that optimized the client consistency levels based on the evaluation of their requirements and resources available in the data center. In order to solve the joint optimization by dual decomposition. They developed a distributed algorithm and argue that their developed solution can improve resource scheduling in the data center cloud. JCSR, with peer-to-peer check-pointing, provides a solution to the joint checkpoint scheduling problem.

Zhang *et al.* [68] proposed the BFT-Cloud architecture based on replication policy. User requests in the cloud were implemented on different nodes with one node is selected as the primary and the remaining nodes as backup nodes. All applications were locally executed on the machine during request execution. In this architecture, the cloud computing system implements users request on different nodes. If the user requests running on both primary and backup nodes are the same, the cloud computing system considers the output to be correct and verify that it responds to the requesting module. However, in case of discrepancy, the cloud computing system considers the node as the faulty node. In this case, recovery operations are performed by updating this stage. If the faulty node is the primary node then update with the new primary version and if the faulty node is other than the primary node i.e., the backup node then updates the replica with the appropriate nodes. The fault-tolerance capability of this framework is 33% which means this framework can identify up to 33 faulty nodes out of 100 nodes in the cloud.

BFTCloud can tolerate all complicated faults and is a highly reliable fault-tolerance architecture. However, the drawback of BFTCloud is that it is low resource utilization.

Chen *et al.* [69] proposed a K-out-of-n fault-tolerance (FT) framework that provides an energy-efficient and fault-tolerance solution for data storage and processing in clouds. To store data, the AllocateData function is developed, and to process data, the ProcessData function is developed. At the time of data storage, the proposed model provided the facility of data fragmentation by a node. The storage requests and processing requests are separated and transferred to their respective functions. The expected transmission time is computed based on operation failure probability, which is estimated then using the K-out-of-n FT framework; and after that, the resources are allocated. This framework consists of five components as follows: (a) Topology Discovery and Monitoring, the current topology of the ad-hoc network is discovered by this component. (b) Failure Probability Estimation, the node failure probability is estimated here. (c) Expected Transmission Time Computation, generates a matrix of communication costs between two nodes. (d) K-out-of-n allocation uses an erasure code algorithm to partition data into n fragments. (e) K-out-of-n Processing, to minimize energy consumption it creates a job that consists of m tasks and schedules them on n processing nodes.

The K-out-of-n FT framework provides the solution for energy-efficient and network fault-tolerance in the cloud. However, resource utilization is significantly low.

Jhawar *et al.* [70] proposed a Fault-tolerance Management (FTM) framework in cloud computing. In this framework, the fault-tolerance approach is applied at the virtualization layer directly. Faults are detected by a runtime monitoring system that uses a heartbeat protocol to detect faults. Following are the basic components of this architecture: i) Resource Manager: this component is responsible to keep the information of resources in the cloud, ii) FTM Kernel: Central component and decision-maker about the type of fault recovery, iii) Client Interface: This component provides the client interface to the fault-tolerance service provider (to put up the user's respective requirements), iv) Messaging Monitor: This component has four sub-components as follows: (a) Replication Manager, which manages the execution of replication resources, (b) Fault Detection/Prediction Manager manages and predicts fault by applying an optimal fault detection algorithm, (c) Fault Masking Manager, this component masks the fault occurrence to hide them from the users, and (d) The recovery Manager maintains and repairs the faulty nodes in the cloud.

The FTM framework can be applied to all crash faults. However, the fault-tolerance overhead is relatively high.

Lin *et al.* [71] proposed VegaWarden (a uniform user management system for cloud applications) to address two problems on user management: usability and security. These problems arise due to cluster virtualization and infrastructure sharing in a cloud computing environment. The proposed VegaWarden architecture was implemented in a Cloud-oriented infrastructure and a production Grid Computing environment. The authors claimed that their proposed uniform user management system can solve these problems. They addressed these problems by providing global userspace and providing a uniform resource interface (using a resource controller) under a single cloud. Different administrative domains were isolated by separate authentication with authorization and access control. Vegawarden uses decentralized and distributed rules through naming service to construct global-consistent and global-synchronous metadata storage. However, when more than one application service is installed authors assumed that the load balancing mechanisms must be used automatically. VegaWarden architecture is a uniform user management system that supports different virtual infrastructure provider and application service provider models in a cloud computing environment.

Fagg and Dongarraet [72] Message Passing Interface (MPI) architecture is a standard for parallel programming, and it uses checkpoint/restart and job migration techniques. Like most of the other models, fault detection is done outside the node and by another module. It has two layers: the top layer does not depend on any infrastructure communication, and the bottom layer, (which is called SSI) specifies the need for backup of the checkpoint. This architecture uses checkpoint/restart and job migration techniques in such a way that if there is a faulty node then it sends MPI Request to the other nodes. The nodes that have received the faulty node request sends the positive response of the request to the faulty node. The faulty node migrates the program to the healthy node from the last checkpoint.

Egwutuoha *et al.* [73] introduced Process Level Redundant (PLR) architecture which creates redundancy in the processes using three policies of the reactive method (replication, job migration, and checkpoint). Furthermore, MPI architecture is embedded in the PLR architecture. The PLR architecture has five modules. The first one is the fault predictor which is for the prediction of fault. The second module is for MPI monitoring and application which is called PLR Controller Daemon, this module is also responsible for replica redundancy management. The next module is the fault-tolerance policy to select the policy type for dealing with certain types of fault. The fourth module is a fault-tolerance dream which first sends a message of monitoring-based to PLR and then performs live migration and checkpoints are initialized as defined in the checkpoint library. Finally, the fifth module failure points are eliminated through the checkpoint/restart method which stores checkpoints on the neighboring nodes, and resources are released used for checkpoints. The policies used in the PLR architecture increase the fault-tolerance capability as well as increases the response time of the system to eliminates failures.

VI. FUTURE RESEARCH DIRECTION FOR FAULT-TOLERANT CLOUD COMPUTING

Cloud computing models and frameworks have attracted significant attention from both the academic research community and industry [10], [75]. In this section, we outline the future directions for fault-tolerance cloud computing development from the perspective of its role in advancing machine learning tools, energy-aware infrastructure, and finally enabling fault-tolerance in emerging architectures based on cloud related technologies.

A. ADVANCING CLOUD FAULT-TOLERANCE USING MACHINE LEARNING TOOLS

Cloud technology has been proposed to fulfill different demands of future networks. To offer shared resources and infrastructure cloud computing models and frameworks are implemented in the data center. In a data center, virtualization approaches are used to structure physical machines into virtual machines [76] for better resource allocation and utilization. However, the main problems faced by cloud service providers are task scheduling, resource allocation, and virtual machine workload management.

Virtual machines in a data center are deployed using cloud infrastructures comprised of several high-volume servers. Task scheduling is the process of embedding Virtual machines in such a way to compose a service chain that minimizes the total run time of service execution and minimizes energy consumption [77]–[79]. However, task scheduling is complex and should be carried out carefully without performance degradation and affecting the operating cloud. Acquiring resource allocation in the cloud requires efficient algorithms to determine the locations of required virtual machines in high-volume servers located in the data center. This then enables the live migration of virtual machines from one location to another for efficient utilization in a cloud. Further, this flexible placement of virtual machines can offer load balancing and optimization of traffic flow.

The methods developed to address the mentioned problems still consume more power, time, and lack quality and accuracy. Advanced machine learning and artificial intelligence [80] tools/algorithms are necessary to develop in order to provide better resource allocation, reduce power consumption, and mitigate VM workload management issues in a cloud. We envisaged that advancing machine learning and artificial intelligence tools/algorithms [81] can provide solutions to mentioned problems and their integration with fault-tolerant methods can significantly improve cloud performance.

B. ENERGY-EFFICIENT FAULT-TOLERANT SCHEDULING ALGORITHMS

Over the past few years, cloud usage has grown. Shortly there will be a significant increase in the usage of cloud infrastructure which has an associated growing impact on energy consumption [82]. Presently, ICTs societies are globally facing energy efficiency challenges coupled with carbon (CO_2) emission [83]–[85]. Recently, significant research studies focused on the development of energy-aware cloud-based infrastructure an initiative to support a green networking environment (eco-friendly) [86]. Energy efficiency in a cloud-based environment is still a key issue, and there is still a need to develop an energy-efficient [87] and energy-aware ICTs infrastructure (cloud-based ecosystem) that is based on energy-efficient Fault-tolerant scheduling algorithms [77], [88].

The emergence of IoT technology has extended the usage of ICT infrastructure [89]. New IoT enables applications such as smart home control [90], industrial automation [91], and the health sector [92] has evolved significantly over the years. To support IoT applications over ICT infrastructure energy management and fault-tolerance are critical. Energy-efficient and optimal fault-tolerant scheduling [93], [94] algorithms are a key requirement for cloud infrastructure to support the goals of the green networking environment [95], [96].

C. FAULT-TOLERANCE IN EMERGING ARCHITECTURES: CLOUD-RELATED TECHNOLOGIES

Cloud computing offers several advantages and efficient usage of shared resources, but due to its centralized approach, it is often considered inefficient for transferring

TABLE 4. Features of different technological concepts.

Technological Concepts	NFV	SDN	P4	Cloud computing	5G
Motivation	To decouple propriety-based hardware appliances (Network Functions (NFs)) to software-based Virtual Network Functions (VNFs) that run over Commercial off-the-shelf (COTS) equipment.	To provide data and control plane separation, enabling the support for simple network programmability.	To develop a protocol-independent tool that can implement a customized switch pipeline, standardized and program dependent APIs.	To simplify resource sharing and the cost of computation significantly.	To build high-capacity networks to offer fast wireless broadband services over Internet Protocol (IP).
Approach	Functions/Service Abstractions (i.e., Virtualization of network functions).	Network Abstractions.	Flexibility to define and parse new fields and protocols.	Computing Abstraction (i.e. Shared pool of configurable storage resources).	Integrate and combine technological concepts to fulfill the needs of new business verticals.
Supporting Organization	European Telecommunications Standards Institute (ETSI).	Open Networking Foundation (ONF) and Linux Foundation.	Open Networking Foundation.	Distributed Management Task Force (DMTF).	3rd Generation Partnership Project (3GPP) and ETSI.
Advantages	To simplify the expertise and efforts needed to design, deploy, and integrate complex hardware-based appliances.	Advances network management and automation.	Updating P4 software can save the cost of purchasing new switches.	Efficient resource allocation and utilization.	Ultra-low latency, ultra reliability, and more than 90% energy efficiency over Long-term Evolution (LTE).

and processing data for latency-sensitive applications. As discussed and predicted in [6], the number of devices connected to the Internet may exceed 50 billion shortly, and data produced by these devices may exceed 500 zettabytes [7]. Due to the growing amount of data and heavy traffic processing, it is becoming difficult to accommodate the transmission of data and real-time processing in the cloud.

In order to address these limitations, fog and edge computing architectures [97] (value addition to the cloud computing architecture) were recently introduced [98]. These computing architectures may appear similar but they vary greatly in terms of their features and they meet differing requirements in certain real-world applications (smart grid, connected vehicles, and smart cities). Furthermore, fog and edge architectures also support emerging technologies such as SDN [99], NFV [100], cloud computing concepts associated with 5G networks.

In Fig. 7 the cloud's fog and edge approaches are illustrated [101].

The requirements for supporting ICTs have recently evolved as never before. Due to the continuous growth of network devices, more and more devices will be connected simultaneously to ICTs infrastructure. To accommodate these devices requires re-designing the network architecture to support the future computing scenarios they enable. As a consequence, Next-generation Networks (NGNs) research is building its foundation based on multiple evolving technologies. These include technologies like NFV, SDN, cloud computing, and the 5G of telecommunications networks, all reputed to transform ICTs infrastructure and fulfill the future needs of computing. It is important to note that all these technologies are directed towards supporting the growing trend of network programmability and network softwarization of telecommunications systems. These concepts

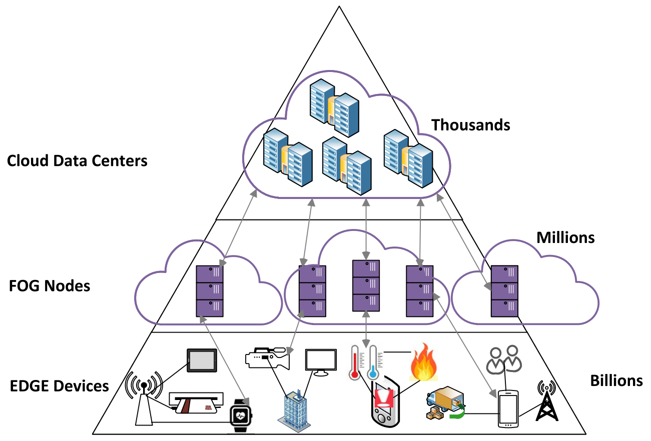

FIGURE 7. Levels of computing: Cloud, Fog, and Edge.

reflect different approaches: NFV provides network service or function abstractions, SDN provides network abstraction, while cloud computing provides computing abstraction. NFV promises to bring flexibility and cost reduction, SDN promises to bring programmable control and open interfaces, and cloud computing promises to bring flexible and efficient pooling and resource sharing of computing power.

In our view, data plane programmability is an important area for future SDN and cloud computing development [102]. Recent studies carried out adopted SDN data plane programmability. New data plane specification such as Protocol Independent Packet Processors (P4) [103] and Protocol-Oblivious Forwarding (POF) [104] has been developed to extend the feature of SDN beyond OpenFlow protocol specifications [105] hence resulting new data plane specification can enhance fault-management in SDN, thus improving the fault-tolerance and reliability aspects of software influenced network based on SDN, NFV, Edge, Fog and Cloud computing-based deployments.

SDN and NFV provide support for multi-tenancy, user, and application quality of service experience. However, in order to sustain the complexity and the heterogeneity requirements associated with 5G technology, for instance, low latency and critical communications scenarios. For such requirements, SDN/NFV needs to be enhanced not only considering orchestration and control plane but also data plane programmability. Considering this enhancement to SDN and NFV a new layer of softwarization has been introduced that provides the feasibility to program the switch data plane through high-level API and languages [106]. The most prominent approach that attracts the research community is the P4 language [107]. The fog/edge node with P4 capabilities can interconnect several heterogeneous network segments and core infrastructures. P4-based data plane programmability enables flexible and advanced programmable functions in SDN/NFV-based cloud computing implementations. Furthermore, P4 programs open the way for the development of novel services to integrate the SDN/NFV and cloud computing paradigms.

In Table 4 we summarize different technological concepts and their features that can impact fault-tolerance in cloud computing environments.

Cloud computing is more applicable to non-real-time applications such as mobile commerce and mobile learning because non-real-time applications require neither mobility localization nor real-time processing of data. Hence, cloud resources can be efficiently utilized. Fog and edge computing are more applicable to real-time applications such as online gaming, smart vehicles, and smart grids. Cloud, fog and edge computing are all relevant and are used today to fulfill different application requirements. It is important to note that cloud, fog, and edge computing paradigms are complementary, and it is desirable to combine these architectures in a network.

VII. OVERALL DISCUSSION AND ANALYSIS

Cloud computing considers the delivery of any hosted services through the Internet. Potential users can access, manipulate, and process their data over the Cloud computing infrastructure instead of purchasing local servers with high maintenance costs. Cloud computing offers elasticity and scalability features to synchronize and update users' data automatically at a low cost. However, despite several advantages of cloud infrastructure, it has inherent vulnerabilities. Therefore, to utilize the cloud computing services, fault-tolerance in cloud computing environments is a critical requirement to maximize the efficiency of cloud computing-based services. To achieve high performance in cloud computing environments, we argue that it is a significant requirement to handle performance-related issues of fault tolerance matrices at the system and component level efficiently (as discussed in sections IV-C and IV-D).

Proactive fault-tolerance approaches are considered better as compared to the reactive fault-tolerance approaches because proactive methods predict fault before their occurrence and replace faulty components or software's with another functional software or components by using the four main proactive fault techniques such as self-healing, preemptive migration, software rejuvenation, and load balancing discussed in section IV-G. For instance, with the self-healing technique, big tasks are divided into multiple small tasks. In the event of a failure, the system is recovered from the faults by analyzing the small tasks and recovering errors without shutting down the complete system. In the preemptive migration technique, failures are being monitored on each running node running on the High-Performance Computing (HPC) system, and in the event of failure, unhealthy nodes are replaced by healthy compute nodes. The software rejuvenation technique is used to deal with software aging-related defects that are caused due to software aging problems. It tolerates software failures, and can perform operations continuously despite interruptions due to software errors. Load balancing provides better scalability and availability by managing system load efficiently. It helps to improve the applications and services response to the end-users by managing task and process flows adequately in the cloud system. However, in general, the proactive approaches rely on predicting the faults before their occurrence, and continuously monitoring the health of the cloud system, which could impose extra overhead and higher execution costs.

On the other hand, reactive methods handle faults after their occurrence, and faulty components or software are managed using a set of already developed maintenance programs based on the selected seven techniques discussed in section IV-F. Reactive approaches are invoked after the occurrence of faults hence at the system level there are no extra overheads used to ensure the functioning of the cloud system itself.

Reactive and proactive fault-tolerance approaches have a significant role in shaping reliable cloud computing applications and services. However, both reactive and proactive approaches are considered in cloud computing based on the applications developed for and purposed used for as summarized in Table 3. The presented state-of-the-art cloud computing fault-tolerance studies show that both techniques have their advantages and limitation. The Hybrid (policies mix of both reactive and proactive) would yield a better result for ensuring better strategies for developing a fault-tolerance cloud computing environment.

VIII. CONCLUSION

In 2019, the global cloud computing market was \$250 billion, and this trend keeps growing. Recent forecast results show that 80% of organizations are poised to migrate to the cloud computing environment by 2025. Therefore, the ability to design, build and migrate cloud-based applications is in very high demand.

In the study of Wang *et al.* [79] and Ding *et al.* [108] about 95% of task completion rate is achieved on real-world cloud workload. However, the remaining 5% tasks were failed which indicates that thousands of tasks were not completed.

Therefore, highly efficient fault-tolerant models and frameworks are required that can increase task completion rate as well as minimize the service failure in cloud-based environments to further enhance fault-tolerance in the cloud computing environments.

In this survey, we overview cloud computing salient features, advantages, components, business/service deployment models, types of cloud, and cloud implementation using distributed, centralized, and decentralized architecture to establish an understanding of cloud computing topics and related concepts comprehensively.

We study cloud fault-tolerance by analyzing fault categories, methods, metrics, tools, and applied fault-tolerance frameworks. Furthermore, we discussed fault-tolerance specific to cloud computing and organize a comprehensive review of the state-of-the-art research focusing on fault-tolerance in cloud computing, considering reactive and proactive approaches and existing fault-tolerance architecture/frameworks and outline future research directions specific to cloud computing fault-tolerance.

The fault-tolerance frameworks studied provide solutions to specific fault types, and a single fault-tolerance framework can not provide a solution to all types of faults. Currently, Service providers choose fault-tolerance mechanisms as per the end-user requirements. The end-user can select services from the service provider catalog (tailor-made to fulfill customer needs). After that service level agreements in terms of fault-tolerance must be guaranteed by the service provider to meet end-user quality of service expectations. However, it may be possible to develop a unified fault-tolerance framework that could provide solutions to most fault types. A unified fault-tolerance framework is easy to manage but difficult to develop due to design complexity.

In summary, in this paper, we highlight salient features and advantages of cloud computing and provide a detailed background of cloud computing. Furthermore, we present a fault-tolerance overview, as well as techniques in the scope of cloud computing. We then highlight proactive and reactive approaches that can provide fault-tolerance in the cloud, after which we structure and organize state-of-the-art research efforts addressing fault-tolerance by utilizing these approaches and discuss existing fault-tolerance architecture/frameworks support for cloud computing. We also outlined important future research directions for fault-tolerance in cloud from multiple perspectives.

REFERENCES

[1] *The NIST Definition of Cloud Computing*, Standard SP 800-145, National Institute of Science and Technology, 2011.

[2] A. Keshavarzi, A. T. Haghighat, and M. Bohlouli, “Research challenges and prospective business impacts of cloud computing: A survey,” in *Proc. IEEE 7th Int. Conf. Intell. Data Acquisition Adv. Comput. Syst. (IDAACS)*, vol. 2, Sep. 2013, pp. 731–736.

[3] M. G. Avram, “Advantages and challenges of adopting cloud computing from an enterprise perspective,” *Proc. Technol.*, vol. 12, pp. 529–534, Jan. 2014.

[4] A. Apostu, F. Puican, G. Ularu, G. Suciu, and G. Todoran, “The advantages of telemetry applications in the cloud,” in *Recent Advances in Applied Computer Science and Digital Services*, vol. 2103, H. Fujita and M. Tuba, Eds. 2013, pp. 118–123. [Online]. Available: https://www.wseas.org/main/books/2013/Morioka/DSAC.pdf

[5] A. Gajbhiye and K. M. P. Shrivastva, “Cloud computing: Need, enabling technology, architecture, advantages and challenges,” in *Proc. 5th Int. Conf. Confluence Next Gener. Inf. Technol. Summit (Confluence)*, Sep. 2014, pp. 1–7.

[6] V. Rajaraman, “Cloud computing,” *Resonance*, vol. 19, no. 3, pp. 242–258, 2014.

[7] P. R. Kumar, P. H. Raj, and P. Jelciana, “Exploring data security issues and solutions in cloud computing,” *Proc. Comput. Sci.*, vol. 125, pp. 691–697, Jan. 2018.

[8] N. Subramanian and A. Jeyaraj, “Recent security challenges in cloud computing,” *Comput. Electr. Eng.*, vol. 71, pp. 28–42, Oct. 2018.

[9] R. Buyya *et al.*, “A manifesto for future generation cloud computing: Research directions for the next decade,” *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–38, Nov. 2018.

[10] M. N. O. Sadiku, S. M. Musa, and O. D. Momoh, “Cloud computing: Opportunities and challenges,” *IEEE Potentials*, vol. 33, no. 1, pp. 34–36, Jan./Feb. 2014.

[11] F. Durao, J. F. S. Carvalho, A. Fonseka, and V. C. Garcia, “A systematic review on cloud computing,” *J. Supercomput.*, vol. 68, no. 3, pp. 1321–1346, Jun. 2014.

[12] R. Lin, Y. Cheng, M. D. Andrade, L. Wosinska, and J. Chen, “Disaggregated data centers: Challenges and trade-offs,” *IEEE Commun. Mag.*, vol. 58, no. 2, pp. 20–26, Feb. 2020.

[13] C. Kachris and I. Tomkos, “A survey on optical interconnects for data centers,” *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1021–1036, 4th Quart., 2012.

[14] H. Qi, M. Shiraz, J.-Y. Liu, A. Gani, Z. A. Rahman, and T. A. Altameem, “Data center network architecture in cloud computing: Review, taxonomy, and open research issues,” *J. Zhejiang Univ. Sci. C*, vol. 15, no. 9, pp. 776–793, Sep. 2014.

[15] B. Mohammed, M. Kiran, K. M. Maiyama, M. M. Kamala, and I.-U. Awan, “Failover strategy for fault tolerance in cloud computing environment,” *Softw., Pract. Exp.*, vol. 47, no. 9, pp. 1243–1274, 2017.

[16] H. P. Zima and A. Nikora, *Fault Tolerance*. Boston, MA, USA: Springer, 2011, pp. 645–658.

[17] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” *IEEE Trans. Depend. Sec. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

[18] M. V. Steen and A. S. Tanenbaum, *Distributed Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2017.

[19] S. Hukerikar and C. Engelmann, “Resilience design patterns—A structured approach to resilience at extreme scale (version 1.0),” 2016, *arXiv:1611.02717*.

[20] A. U. Rehman, R. L. Aguiar, and J. P. Barraca, “Fault-tolerance in the scope of software-defined networking (SDN),” *IEEE Access*, vol. 7, pp. 124474–124490, 2019.

[21] A. Ganesh, M. Sandhya, and S. Shankar, “A study on fault tolerance methods in cloud computing,” in *Proc. IEEE Int. Adv. Comput. Conf. (IACC)*, Feb. 2014, pp. 844–849.

[22] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, “Elasticity in cloud computing: A survey,” *Ann. Telecommun.-Annales des Télécommun.*, vol. 70, no. 7, pp. 289–309, 2015.

[23] M. N. Cheraghlou, A. Khadem-Zadeh, and M. Haghparast, “A survey of fault tolerance architecture in cloud computing,” *J. Netw. Comput. Appl.*, vol. 61, pp. 81–92, Feb. 2016.

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

[25] P. Kumari and P. Kaur, “A survey of fault tolerance in cloud computing,” *J. King Saud Univ., Comput. Inf. Sci.*, vol. 33, no. 10, pp. 1159–1176, 2018.

[26] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: State-of-the-art and research challenges,” *J. Internet Services Appl.*, vol. 1, no. 1, pp. 7–18, May 2010.

[27] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A view of cloud computing,” *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[28] T. Noor, S. Zeadally, A. Alfazic, and Q. Z. Sheng, “Mobile Cloud computing: Challenges and future research directions,” *J. Netw. Comput. Appl.*, vol. 115, pp. 70–85, Aug. 2018.

[29] K. Bilal, O. Khalid, S. U. R. Malik, M. U. S. Khan, S. U. Khan, and A. Y. Zomaya, *Fault Tolerance in the Cloud*. Hoboken, NJ, USA: Wiley, 2016, ch. 24, pp. 291–300.

[30] H. T. M. Gamage, H. D. Weerasinghe, and N. G. J. Dias, "A survey on blockchain technology concepts, applications, and issues," *Social Netw. Comput. Sci.*, vol. 1, no. 2, pp. 1–15, Mar. 2020.

[31] C. V. N. U. B. Murthy, M. L. Shri, S. Kadry, and S. Lim, "Blockchain based cloud computing: Architecture and research challenges," *IEEE Access*, vol. 8, pp. 205190–205205, 2020.

[32] M. van Steen and A. S. Tanenbaum, "A brief introduction to distributed systems," *Computing*, vol. 98, no. 10, pp. 967–1009, Oct. 2016.

[33] K. Nørvåg, "An introduction to fault-tolerant systems," Dept. Comput. Inf. Sci., Norwegian Univ. Sci. Technol., Trondheim, Norway, Tech. Rep. IDI 6/99, Jul. 2000. [Online]. Available: https://folk.idi.ntnu.no/noervaag/papers/IDI-TR-6-99.pdf

[34] R. Arno, P. Gross, and R. Schuerger, "What five 9's really means and managing expectations," in *Proc. Conf. Rec. IEEE Ind. Appl. Conf. 41st IAS Annu. Meeting*, Oct. 2006, pp. 270–275.

[35] O. Haq, M. Raja, and F. R. Dogar, "Measuring and improving the reliability of wide-area cloud paths," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 253–262.

[36] D. K. Pradhan, *Fault-tolerant Computer System Design*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[37] E. Dubrova, *Fault-Tolerant Design*. New York, NY, USA: Springer, 2013. [Online]. Available: http://www.alavinia.ir/ebook/FaultTolerantDesign.pdf, doi: 10.1007/978-1-4614-2113-9.

[38] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Reliability and high availability in cloud computing environments: A reference roadmap," *Hum.-Centric Comput. Inf. Sci.*, vol. 8, no. 1, pp. 1–31, Dec. 2018.

[39] A. J. Frank. (Dec. 2018). *Fault Tolerance*. [Online]. Available: http://u.cs.biu.ac.il/~ariel/download/ds590/pdfs/chp08.pdf

[40] S. M. A. Ataallah, S. M. Nassar, and E. E. Hemayed, "Fault tolerance in cloud computing—Survey," in *Proc. 11th Int. Comput. Eng. Conf. (ICENCO)*, Dec. 2015, pp. 241–245.

[41] H. Agarwal and A. Sharma, "A comprehensive survey of fault tolerance techniques in cloud computing," in *Proc. Int. Conf. Comput. Netw. Commun. (CoCoNet)*, Dec. 2015, pp. 408–413.

[42] E. Dubrova, *Software Redundancy*. New York, NY, USA: Springer, 2013, pp. 55–86,157–179.

[43] D. A. Rennels, "Fault-tolerant computing," in *The Encyclopedia of Computer Science*. Chichester, U.K.: Wiley, 2000, pp. 698–702.

[44] J. Zhang, A. Zhou, Q. Sun, S. Wang, and F. Yang, "Overview on fault tolerance strategies of composite service in service computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–8, Jun. 2018.

[45] 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.

[46] L. P. Saikia and Y. L. Devi, "Fault tolerance techniques and algorithms in cloud computing," *Int. J. Comput. Sci. Commun. Netw.*, vol. 4, no. 1, pp. 1–8, 2014.

[47] D. Poola, M. A. Salehi, K. Ramamohanarao, and 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*, I. Mistrik, R. Bahsoon, N. Ali, M. Heisel, and B. Maxim, Eds. Boston, MA, USA: Morgan Kaufmann, 2017, ch. 15, pp. 285–320.

[48] G. C. Fox, R. D. Williams, and G. C. Messina, *Parallel Computing Works!* Amsterdam, The Netherlands: Elsevier, 2014.

[49] C. Wu, R. Buyya, and K. Ramamohanarao, "Big data analytics = machine learning + cloud computing," in *Big Data*, R. Buyya, R. N. Calheiros, and A. V. Dastjerdi, Eds. San Mateo, CA, USA: Morgan Kaufmann, 2016, ch. 1, pp. 3–38.

[50] T. Sterling, M. Anderson, and M. Brodowicz, "MapReduce," in *High Performance Computing*, T. Sterling, M. Anderson, and M. Brodowicz, Eds. Boston, MA, USA: Morgan Kaufmann, 2018, ch. 19, pp. 579–589.

[51] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of 'big data' on cloud computing: Review and open research issues," *Inf. Syst.*, vol. 47, pp. 98–115, Jul. 2015.

[52] I. Polato, R. Ré, A. Goldman, and F. Kon, "A comprehensive view of Hadoop research—A systematic literature review," *J. Netw. Comput. Appl.*, vol. 46, pp. 1–25, Nov. 2014.

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

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

[55] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, "FTCloud: A component ranking framework for fault-tolerant cloud applications," in *Proc. IEEE 21st Int. Symp. Softw. Rel. Eng.*, Nov. 2010, pp. 398–407.

[56] I. P. Egwutuoha, "A proactive fault tolerance framework for high performance computing (HPC) systems in the cloud," Ph.D. dissertation, Dept. Electr. Inf. Eng., Univ. Sydney, Sydney, NSW, Australia, Dec. 2013. [Online]. Available: http://hdl.handle.net/2123/11484

[57] J. Liu, S. Wang, A. Zhou, S. A. P. Kumar, F. Yang, and R. Buyya, "Using proactive fault-tolerance approach to enhance cloud service reliability," *IEEE Trans. Cloud. Comput.*, vol. 6, no. 4, pp. 1191–1202, Oct./Dec. 2018.

[58] J. Liu, J. Zhou, and R. Buyya, "Software rejuvenation based fault tolerance scheme for cloud applications," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 1115–1118.

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

[60] M. R. Chinnaiah and N. Niranjan, "Fault tolerant software systems using software configurations for cloud computing," *J. Cloud Comput.*, vol. 7, no. 1, p. 3, Dec. 2018.

[61] P. Zhang, S. Shu, and M. Zhou, "An online fault detection model and strategies based on SVM-grid in clouds," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 2, pp. 445–456, Mar. 2018.

[62] V. Kaushal and A. G. Bala, "Autonomic fault tolerance using haproxy in cloud enviorment," M.S. thesis, Dept. Comput. Sci., Thapar Univ. Patiala, India, 2011. [Online]. Available: http://hdl.handle.net/10266/1439

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

[64] A. Zhou, S. Wang, B. Cheng, Z. Zheng, F. Yang, R. N. Chang, M. R. Lyu, and R. Buyya, "Cloud service reliability enhancement via virtual machine placement optimization," *IEEE Trans. Services Comput.*, vol. 10, no. 6, pp. 902–913, Nov./Dec. 2017.

[65] D. Sun, G. Chang, C. Miao, and X. Wang, "Modelling and evaluating a high serviceability fault tolerance strategy in cloud computing environments," *Int. J. Secur. Netw.*, vol. 7, no. 4, pp. 196–210, 2012.

[66] A. Zhou, Q. Sun, and J. Li, "Enhancing reliability via checkpointing in cloud computing systems," *China Commun.*, vol. 14, no. 7, pp. 1–10, Jul. 2017.

[67] J. Zhao, Y. Xiang, T. Lan, H. H. Huang, and S. Subramaniam, "Elastic reliability optimization through peer-to-peer checkpointing in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 491–502, Feb. 2017.

[68] Y. Zhang, Z. Zheng, and M. R. Lyu, "BFTCloud: A Byzantine fault tolerance framework for voluntary-resource cloud computing," in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, Jul. 2011, pp. 444–451.

[69] C.-A. Chen, M. Won, R. Stoleru, and G. G. Xie, "Energy-efficient fault-tolerant data storage and processing in mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 28–41, Jan./Mar. 2015.

[70] R. Jhawar, V. Piuri, and M. Santambrogio, "Fault tolerance management in cloud computing: A system-level perspective," *IEEE Syst. J.*, vol. 7, no. 2, pp. 288–297, Jun. 2013.

[71] J. Lin, X. Lu, L. Yu, Y. Zou, and L. Zha, "VegaWarden: A uniform user management system for cloud applications," in *Proc. IEEE 5th Int. Conf. Netw., Archit., Storage*, Jul. 2010, pp. 457–464.

[72] G. E. Fagg and J. J. Dongarra, "FT-MPI: Fault tolerant MPI, supporting dynamic applications in a dynamic world," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, J. Dongarra, P. Kacsuk, and N. Podhorszki, Eds. Berlin, Germany: Springer, 2000, pp. 346–353.

[73] I. P. Egwutuoha, S. Chen, D. Levy, and B. Selic, "A fault tolerance framework for high performance computing in cloud," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2012, pp. 709–710.

[74] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, Jun. 2007.

[75] P. K. Senyo, E. Addae, and R. Boateng, "Cloud computing research: A review of research themes, frameworks, methods and future research directions," *Int. J. Inf. Manage.*, vol. 38, no. 1, pp. 128–139, 2018.

[76] A. Abdelaziz, M. Elhoseny, A. S. Salama, and A. M. Riad, "A machine learning model for improving healthcare services on cloud computing environment," *Measurement*, vol. 119, pp. 117–128, Apr. 2018.
[77] P. Guo, M. Liu, J. Wu, Z. Xue, and X. He, "Energy-efficient fault-tolerant scheduling algorithm for real-time tasks in cloud-based 5G networks," *IEEE Access*, vol. 6, pp. 53671–53683, 2018.
[78] Y. Zhang, X. Cheng, L. Chen, and H. Shen, "Energy-efficient tasks scheduling heuristics with multi-constraints in virtualized clouds," *J. Grid Comput.*, vol. 16, no. 3, pp. 459–475, Sep. 2018.
[79] J. Wang, W. Bao, X. Zhu, L. T. Yang, and Y. Xiang, "FESTAL: Fault-tolerant elastic scheduling algorithm for real-time tasks in virtualized clouds," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2545–2558, Sep. 2015.
[80] M. Rath, J. Satpathy, and G. S. Oreku, "Artificial intelligence and machine learning applications in cloud computing and Internet of Things," in *Artificial Intelligence to Solve Pervasive Internet of Things Issues*, G. Kaur, P. Tomar, and M. Tanque, Eds. New York, NY, USA: Academic, 2021, ch. 6, pp. 103–123.
[81] S. Mahfoudhi, M. Frehat, and T. Moulahi, "Enhancing cloud of things performance by avoiding unnecessary data through artificial intelligence tools," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 1463–1467.
[82] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, "Trends in worldwide ICT electricity consumption from 2007 to 2012," *Comput. Commun.*, vol. 50, pp. 64–76, Sep. 2014.
[83] A. Fehske, G. Fettweis, J. Malmodin, and G. Biczok, "The global footprint of mobile communications: The ecological and economic perspective," *IEEE Commun. Mag.*, vol. 49, no. 8, pp. 55–62, Aug. 2011.
[84] W. Vereecken, W. Van Heddeghem, D. Colle, M. Pickavet, and P. Demeester, "Overall ICT footprint and green communication technologies," in *Proc. 4th Int. Symp. Commun., Control Signal Process. (ISCCSP)*, Mar. 2010, pp. 1–6.
[85] W. Van Heddeghem, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester, "Distributed computing for carbon footprint reduction by exploiting low-footprint energy availability," *Future Gener. Comput. Syst.*, vol. 28, no. 2, pp. 405–414, Feb. 2012.
[86] A. P. Bianzino, C. Chaudet, D. Rossi, and J.-L. Rougier, "A survey of green networking research," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 1, pp. 3–20, 1st Quart., 2012.
[87] K. Gai, M. Qiu, and H. Zhao, "Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing," *J. Parallel Distrib. Comput.*, vol. 111, pp. 126–135, Jan. 2018.
[88] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 2, pp. 319–333, Feb. 2019.
[89] G. Ortiz, M. Zouai, O. Kazar, A. Garcia-de-Prado, and J. Boubeta-Puig, "Atmosphere: Context and situational-aware collaborative IoT architecture for edge-fog-cloud computing," *Comput. Standards Interfaces*, vol. 79, Jan. 2022, Art. no. 103550.
[90] M. Kasmi, F. Bahloul, and H. Tkitek, "Smart home based on Internet of Things and cloud computing," in *Proc. 7th Int. Conf. Sci. Electron., Technol. Inf. Telecommun. (SETIT)*, Dec. 2016, pp. 82–86.
[91] H. P. Breivold and K. Sandström, "Internet of Things for industrial automation—Challenges and technical solutions," in *Proc. IEEE Int. Conf. Data Sci. Data Intensive Syst.*, Dec. 2015, pp. 532–539.
[92] O. Ali, A. Shrestha, J. Soar, and S. F. Wamba, "Cloud computing-enabled healthcare opportunities, issues, and applications: A systematic review," *Int. J. Inf. Manage.*, vol. 43, pp. 146–158, Dec. 2018.
[93] S. Bansal, R. K. Bansal, and K. Arora, "Energy efficient backup overloading schemes for fault tolerant scheduling of real-time tasks," *J. Syst. Archit.*, vol. 113, Feb. 2021, Art. no. 101901.
[94] Z. Li, V. Chang, H. Hu, H. Hu, C. Li, and J. Ge, "Real-time and dynamic fault-tolerant scheduling for scientific workflows in clouds," *Inf. Sci.*, vol. 568, pp. 13–39, Aug. 2021.
[95] R. Arshad, S. Zahoor, M. A. Shah, A. Wahid, and H. Yu, "Green IoT: An investigation on energy saving practices for 2020 and beyond," *IEEE Access*, vol. 5, pp. 15667–15681, 2017.
[96] Z. Niu, "Green communication and networking: A new horizon," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 3, pp. 629–630, Sep. 2020.
[97] P. Varshney and Y. Simmhan, "Characterizing application scheduling on edge, fog, and cloud computing resources," *Softw. Pract. Exp.*, vol. 50, no. 5, pp. 558–595, May 2020.
[98] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Gener. Comput. Syst.*, vol. 79, pp. 849–861, Feb. 2018.
[99] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[100] A. U. Rehman, R. L. Aguiar, and J. P. Barraca, "Network functions virtualization: The long road to commercial deployments," *IEEE Access*, vol. 7, pp. 60439–60464, 2019.
[101] H.-J. Cha, H.-K. Yang, and Y.-J. Song, "A study on the design of fog computing architecture using sensor networks," *Sensors*, vol. 18, no. 11, p. 3633, Oct. 2018.
[102] O. Michel, R. Bifulco, G. Rétvári, and S. Schmid, "The programmable data plane: Abstractions, architectures, algorithms, and applications," *ACM Comput. Surveys*, vol. 54, no. 4, pp. 1–36, May 2022, doi: 10.1145/3447868.
[103] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.
[104] H. Song, "Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*. New York, NY, USA: Association for Computing Machinery, 2013, pp. 127–132.
[105] W. L. da Costa Cordeiro, J. A. Marques, and L. P. Gaspary, "Data plane programmability beyond OpenFlow: Opportunities and challenges for network and service operations and management," *J. Netw. Syst. Manage.*, vol. 25, no. 4, pp. 784–818, Oct. 2017.
[106] F. Paolucci, F. Cugini, P. Castoldi, and T. Osinski, "Enhancing 5G SDN/NFV edge with p4 data plane programmability," *IEEE Netw.*, vol. 35, no. 3, pp. 154–160, May 2021.
[107] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on P4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, vol. 9, pp. 87094–87155, 2021.
[108] Y. Ding, G. Yao, and K. Hao, "Fault-tolerant elastic scheduling algorithm for workflow in cloud systems," *Inf. Sci.*, vol. 393, pp. 47–65, Jul. 2017.

A. U. REHMAN received the Ph.D. degree in telecommunications from MAP-tele–a joint doctoral program of the Universidade do Porto, the Universidade de Aveiro, and the Universidade do Minho, Portugal, three universities with a strong tradition in the area of telecommunications engineering). He received the master's degree with Distinction in telecommunications engineering from The University of Sunderland, U.K., in 2011, having followed an approved program in telecommunications engineering. He was also a Research Fellow of the Fundacao para a Ciencia e a Tecnologia/Ministerio da Educacao e Ciencia (FCT/MEC) hosted by the Instituto de Telecomunicacoes. He has been developing research work in the areas of reliability at a different level from software-defined networking (SDN), network functions virtualization (NFV) to cloud computing, and service assurance of virtual networks. His research interests include software-defined networking, network functions virtualization, reliability and resilience in future networks, and mobile networks (5G, and beyond). He was a Visiting Scholar at Telenor Research ASA, HQ, Norway (research focus: software-defined networks, network functions virtualization, and service assurance). He has working experience of over 7 years of working with research institutes and over 12 years of education and working experience in the EU and EEA area. His professional background is a mix of teaching and practical experience gained from working at telecommunications research institutes and in the industry with different organizations. He received the certificate of excellence in ethics, research data management, manuscript writing, and several others from Elsevier. He already published articles in IEEE high-impact journals and conferences. He is reviewing articles for IEEE ACCESS and IEEE INTERNET OF THINGS JOURNAL. His verified reviews are available at Publons. He is also a member of the Future Networks Community, the Communications Society (ComSoc), and the Software-defined Networks Community, IEEE.

RUI L. AGUIAR (Senior Member, IEEE) received the degree in telecommunication engineering and the Ph.D. degree in electrical engineering from the University of Aveiro, in 1990 and 2001, respectively. He is currently a Full Professor with the University of Aveiro, responsible for the networking area, and has been previously an Adjunct Professor at INI, Carnegie Mellon University. He was a Visiting Research Scholar with the Universidade Federal de Uberlândia, Brazil, and served as an Advisor to the Portuguese government on 5G policies. He is coordinating a research line nationwide in the Instituto de Telecomunicações, in the area of networks and services. During six years, he led the Technological Platform on Connected Communities, a regional cross-disciplinary industry-oriented activity on smart environments. His current research interests include the implementation of advanced wireless networks and systems, with special emphasis on 5G networks, and the future internet. He has more than 500 published articles in those areas, including standardization contributions to IEEE and IETF. He has served as the Technical and General Chair for several conferences from IEEE, ACM, and IFIP, and is regularly invited for keynotes on 5G and future internet networks. He sits on the TPC of most major IEEE ComSoc conferences. He has extensive participation in national and international projects, as well as in industry technology transfer actions. He is currently associated with the 5G PPP Infrastructure Association and is the current Chair of the Steering Board of the Networld Europe ETP. He is the Portugal ComSoc Chapter Chair and a member of ACM. He is an Associate Editor of Wiley's *ETT*, and Springers' *Wireless Networks* and has helped on the launch of Elsevier's *ICT Express*. He is a Chartered Engineer, has acted as a consultant to several major operators, as a technology advisor to bootstrap several SMEs, and as an expert to several public bodies, both on the societal and on the judiciary branches. He currently sits on the Advisory Board of several EU projects and research units.

JOÃO PAULO BARRACA received the Ph.D. degree in informatics engineering from the Universidade de Aveiro, in 2012. He is currently an acting Assistant Professor with the Universidade de Aveiro. He conducts research with the Instituto de Telecomunicações, having led the TN-AV Group, from 2015 to 2016. He has more than 100 peer-reviewed publications and reports related to solutions for the Internet of Things and software for cloud environments, with a focus on software-defined networking and 5G networks. Having participated in many review panels, he has also organized workshops and conferences. He has participated in more than 20 projects, either developing novel concepts or applying these concepts in innovative products and solutions. He leads the FCT/CAPES DEVNF Project in Portugal devoted to NFV orchestration, the local teams of EU LIFE-PAYT, participates in European Science Cloud for Astronomy (EUAENEAS), the local team in the P2020 (CRUISE Project), the security team at P2020-Social, participates in the EU Interreg CISMOB smart cities pilot, the Engage SKA research infrastructure, and the Square Kilometer Array System (SKA) Team, and having lead activities for TM-LINFRA, among a dozen other innovation projects. Recently, he received the third place from the INCM Innovation Challenge, for the development of a project targeting smarter environments for public transports in smart cities, using block chain technologies.