The Scientific Temper (2023) Vol. 14 (3): 944-949

E-ISSN: 2231-6396, ISSN: 0976-8653

Doi: 10.58414/SCIENTIFICTEMPER.2023.14.3.59

https://scientifictemper.com/

REVIEW ARTICLE

Fault tolerance systems in open source cloud computing environments–A systematic review

K. Vani[1] and S. Sujatha[2]

Abstract

Considering that there are several distinct cloud computing environments and several suggested approaches for the treatment of fault tolerance for such environments, the objective of the study presented here is a systematization of fault tolerance proposals that results in a survey and the generation of a guided consultation environment for reading the relevant techniques for each case. With the systematization of proposed solutions, it is intended to obtain a document that administrators of cloud computing systems can use. This work points out which techniques apply to which problems, including the advantages and disadvantages of each technique, and facilitates the support process for these administrators in handling the failures. Finally, with the information obtained, a website will be generated to store some of this information. This virtual environment is a prototype of a recommendation environment for cloud fault tolerance. At first, the recommendation will occur through guided search so that administrators of cloud computing systems can have better conditions to handle failures in their environments.

Keywords: Fault tolerant, Load balance, Cloud computing, Failures, Virtual environment.

Introduction

For cloud computing, seen as a business, the provider must fulfill the service level agreement (SLA) established with its customers. Please comply with this contract to ensure the quality of the hosted service is maintained and, consequently, customer satisfaction. The failures can generate numerous losses, both for the customer and for the provider of cloud computing services. Therefore, it is important that the provided service can occur without interruptions or performance losses. That is, it is important that the system is fault tolerant. The quest to maintain a fault-tolerant environment must be constant since numerous failures can be resolved in many ways. This has been worked on in different ways, with proposals that often do not bring a real gain in relation to other existing methods, which is an error. Unfortunately, this error has been repeated repeatedly, which has generated a huge amount of approaches for handling failures, causing difficulties in identifying which techniques should be used. This work presents a guided search website developed based on information extracted from a systematic literature review. With the review, a survey was created regarding fault tolerance in cloud computing, containing solutions to handle failures in different environments and situations. The synthesized form of this survey allowed the creation of the search website (Mell & Grance, 2011). The work presented here evaluates these contributions based on a proposal for a methodology for classifying fault tolerance techniques, which allows the identification of the most appropriate technique for the specific case of an administrator of cloud computing systems.

Systems analysis for fault tolerance in cloud computing environments

This section presents the tools and systems from the articles selected for evaluation with the review described. The tools were subdivided according to the type of failure, which could be data, process, communication, or virtualization failures. Each item will be subdivided according to the technique used for handling failures: checkpointing, replication, work

[1]Department of computer science, Emerald Heights College For Women, Finger Post, Ooty, Tamil Nadu, India.

[2]Department of computer science, Dr.G.R. Damodaran College of Science, Coimbatore, Tamil Nadu, India.

***Corresponding Author:** K. Vani, Department of computer science, Emerald Heights College For Women, Finger Post, Ooty, Tamil Nadu, India, E-Mail: vanicloudresearcher@gmail.com.

How to cite this article: Vani, K., Sujatha, S. (2023). Fault tolerance systems in open source cloud computing environments–A systematic revie. The Scientific Temper, 14(3): 944-949

Doi: 10.58414/SCIENTIFICTEMPER.2023.14.3.59

Source of support: Nil

Conflict of interest: None.

Received: 11/07/2023 **Accepted:** 18/08/2023 **Published** : 25/09/2023

migration, repeat, self-healing and preemptive migration, for example. After these sections, a section was added to describe the website prototype that will serve as a place to carry out guided searches for the solution to a given problem.

In order to more clearly present the solutions identified after the systematic review work, a classification approach was adopted in this section, initially based on the type of failure treated. Thus, the analysis is separated into the following (Javadi *et al.,* 2012):

- Systems handling data-related failures;
- Systems handling communication-related failures;
- Systems handling process related failures; and
- Systems handling virtualization related failures.

data-related failures

In this section, proposals to tolerate data-related failures are described. The study addresses these failures by checkpointing, self-healing, preemptive migration, retry, handling user-defined exceptions, workflow rescue, job migration, or replication. Some of these proposals will be presented in detail in later sections, only named here.

Use of Checkpointing

Checkpointing is ideal for situations with a large volume of data since, in its most straightforward approach, it guarantees the recovery of the environment from the most recent checkpoint. In this category, there is the IGG approach, presented below.

InterGrid Gateway (IGG)

In (Javadi *et al.,* 2012), failure handling is done using checkpointing, which restarts the request for a new VM from the last moment of availability. The filling scheduling algorithms were modified to support the perfect checkpointing mechanism and provide a fault-tolerant environment to serve private cloud requests. Table 1 summarizes the main characteristics of the system.

Preemptive Migration

Preemptive migration makes it possible to migrate data with low computational costs. In this category, there is the HySARC2 approach (Vasile *et al.,* 2015), presented below.

HySARC2

The purpose of HySARC2 is to improve scheduling in a given cloud environment through service grouping and labeling systems, ensuring the proper use of resources and consequently satisfying user requirements and service provider interests. New scheduling approaches must monitor the system's structure and adjust the schedule according to real-time usage. This also aids in a more fault-tolerant system, as certain changes are not always expected. Table 1 summarizes the main characteristics of the system.

Job migration

If, for some reason, a certain task cannot be entirely executed, the task is migrated to a new machine. Therefore, the environment must have adequate machines and/or VMs. The SkyCDS approach is presented below in this category (Gonzalez *et al.,* 2015).

SkyCDS (Prototype)

It is a system focused on content delivery service (CDS) based on overlay publication/subscription in cloud storage. Delivery is split into metadata stream and content store tiers. The system is able to reduce the overhead of content dispersion and process retrieval.

Use of Self-Healing

The self-healing technique makes it possible to recover the environment with little or no human intervention. In this category, there are the DARGOS approaches (Javier *et al.,* 2013) and a prototype for the automatic treatment of anomalies (Gulenko *et al.,* 2016), presented below.

Distributed Architecture for Resource management and monitoring in clouds (DARGOS)

The system was designed to satisfy the main requirements of a cloud environment while having low latency and overhead. Based on the publication/subscription paradigm, the environment allows choosing zones and other communication resources to be monitored, with update rates and a set of sensors (Table 1).

Prototype for automatic treatment of anomalies

Through supervised and unsupervised artificial intelligence, the proposal aims to create a system for automatic and real-time masking of failures, being the focus of Network Function Virtualization (NFV) environments. Table 2 summarizes the main characteristics of the system.

Table 1: Summary of the IGG, HySARC2, SkyCDS, and DRAGOS systems.

System/ characteristics	*Techniques used*	*Compatible cloud manager*	*Programming model*	*Solved problem*	*Type of fault handled*
IGG	Checkpointing	OpenNebula and Eucalyptus	Java	QoS; Resource Provisioning	Data
HySARC2	Preemptive migration	OpenStack	Not informed by the authors	Resource provisioning	Processes and data
SkyCDS	Work migration	OpenStack	Not informed by the authors	A new way to compare storage options	Delivery of the risk assessment
DRAGOS	Self-healing and preemptive migration	OpenStack	C, JavaScript, and Python	Data	Data

Table 2: Summary Prototype for automatic treatment of anomalies

Techniques used	*Replication, Self-healing, and preemptive migration*
Compatible Cloud Manager	OpenStack
Programming Model	Python
Solved problem	Solution involving intelligence for NFV mainly
Type of Fault Handled	Communication; Process; Data; Virtualization
Advantage	Automatic fault-masking, avoiding interruptions in the system
Disadvantage	It is a prototype

Use of Replication

Replication is a simple technique to be implemented and used, which can be applied in different cases within the context of recovery from data-related failures. For example, one (or n) identical copy(s) of the database can be made, thus ensuring an environment with high availability, even if in a highly costly manner. In this category there is the SprintNet approach (Wang *et al.,* 2015), presented below, in addition to proposals such as FIR3 (Vijayakumar *et al.,* 2015), DCR2S (Gill & Singh, 2016) Morpho (Lu *et al.,* 2015), Tahoe-LAFS (Selimi *et al.,* 2019), Hybrid algorithm based on MapReduce (Zhang *et al.,* 2019), GFS (Nakanishi *et al.,* 2014), Private multilayer storage system (Gonzalez *et al.,* 2013) and SwiftER (Datta *et al.,* 2016), which are presented below.

Tahoe-LAFS

It is an open-source system for cloud computing focused on fault tolerance in storage nodes. Offers access through multiple interfaces (Web, OS, SSH), ensuring privacy and security by encrypting data on the client side. In experiments done under different conditions, the application was able to recover all different file sizes, even in a community network (Selimi *et al.,* 2019). The replication system is based on erasure code, in which every new file is separated into n different shares, and can be recovered from any share. Table 3 summarizes the main characteristics of the system.

swifter

The proposal was initially given by (Datta *et al.,* 2016) and aimed to reduce the data stored with replication. By using the Erasure technique, the system was named SwifER (Swift Erasure). The data duplicated by the system is stored as a Raid, which can reduce the storage space by 1.2x to 3x, maintaining reliability and high availability. Its operation occurs in OpenStack's Swift layer (Table 3).

Multi-tier private storage system

The system proposed by (Gonzalez *et al.,* 2013) is a different storage service enabling the transfer of redundant information between levels. File availability is guaranteed through the unified system that allows recovery from different categories of service failures. Table 3 summarizes the main characteristics of the system.

Gluster File System (GFS)

The system was designed based on high-performance computing concepts. At the same time, it has a simplified structure similar to that of RAID10. According to (Nakanishi *et al.,* 2014), in tests compared to Swift, GFS showed data I/O up to 4.5x faster.

Replication is done on a file basis, where a distributed hash is used to statically allocate elementary spaces called "bricks" for the entire space of stored file names. Table 3 summarizes the main characteristics of the system.

Hybrid algorithm based on MapReduce

In this system, the algorithm is based on MapReduce using top down specialization (TDS) and bottom-up generalization (BUG). The proposed algorithm is able to normalize subtrees. According to the tests carried out by the authors (Zhang *et al.,* 2019), the environment significantly improves efficiency and scalability compared to existing approaches. Table 4 summarizes the main characteristics of the system.

Morpho

The proposal is basically a modification of the Hadoop and MapReduce framework. Morpho was designed to improve the cooperativity of the compute and storage layers. That is, MapReduce tasks can fetch information about the physical machines' network topology and the VMs' allocations.

The system also uses complementary strategies for

Table 3: Summary of the Tahoe-LAFS, SwiftER, SwiftER, Multitier storage, and GFS systems.

System/ characteristics	*Techniques used*	*Compatible cloud manager*	*Programming model*	*Solved problem*	*Type of fault handled*
Tahoe-LAFS	Replication	Universal	python	Data recovery (availability)	Data
SwiftER	Replication	OpenStack	Python	Improved storage space; more efficient than "traditional" replication	Data and communication
Multitier Storage	Replication	OpenStack	Java	Data and Communication	File recovery in different layers
GFS	Replication	OpenStack	C/C++	Low CPU usage (less than 20%); faster I/O	Data

allocating data from the VMs, generating better mapping and reducing the input location. Table 4 summarizes the main characteristics of the system.

Dynamic Cost-Aware Re-Replication and Re-balancing Strategy (DCR2S)

The system is compatible with heterogeneous clouds and uses an optimized dynamic replication strategy, identifying the minimum number of replicas necessary to guarantee the desired availability. Table 4 summarizes the main characteristics of the system.

Fuzzy Inference based Reliable Replica Replacement (FIR3)

Data loss or service interruptions can occur frequently in internet-based computing, to solve this problem the authors (Vijayakumar *et al.,* 2015) created a new data replication technique based on the Fuzzy Inference System.

The main idea is to keep each data replica in the different Availability Zones. The algorithm uses fuzzy inference helps in solving space inconsistency problems. Replication is deployed in the cloud stack environment, so this replication technique will improve the entire fault tolerance of the system. Table 4 summarizes the main characteristics of the system.

Medical Image File Accessing System (MIFAS)

The proposed system is based on the Hadoop Distributed File System (HDFS). It brings improvements in medical image storage, stability during transmissions, and reliability, in addition to providing an easy-to-manage interface.
The Replication Location Service automatically duplicates from one cloud to another (even if distinct) when medical images are uploaded to MIFAS. The results of the experiments prove that the system has high reliability and fault tolerance (Sujatha *et al.,* 2010). Table 5 summarizes the main characteristics of the system.

System based on the 80/20 principle

Literature (Vijayakumar *et al.,* 2015) tested a system based on the 80/20 rule (80% of cluster failures come from 20% of physical machines).

The idea is to help identify physical machines prone to failures in clusters. Machines are subdivided into two subsets: reliable (70% to 80% of machines) and risky (20% to 30% of machines). The trusted subgroup includes a highly trusted zone, providing high availability for latent jobs. Table 5 summarizes the main characteristics of the system.

Cloud middleware to ensure real-time performance and high availability of soft applications

The software proposed by (Zhang *et al.,* 2019) presents a framework capable of implementing virtual machine replicas according to users' predefined flexible algorithms. The system includes a Local Fault Manager (LFM) for each host and a Global Replicated Fault Manager (GFM) to manage clusters of physical machines (Table 5).

Mosaic

To address various cloud usage scenarios and provide additional solutions for portability, (Zhang *et al.,* 2019) designed the mOSAIC, whose main characteristics are seen in Table 5. The premises to be fulfilled by the

Table 4: Summary of the Hybrid algorithm based on MapReduce, Morpho, DCR2S, and FIR3 systems

System/ characteristics	*Techniques used*	*Compatible cloud manager*	*Prog. model*	*Solved problem*	*Type of fault handled*
Hybrid algorithm based on MapReduce	Replication	OpenStack	Not informed by the authors	Normalization of subtrees	Data
Morpho	Replication	OpenNebula	Java	Improved resource allocation and data storage	Data and virtualization
DCR2S	Replication	Universal	Not informed by the authors	Data recovery (availability)	Data
FIR3	Replication	Cloud stack	Java	File replacement is effectively handled using the Fuzzy Inference System and effective consistency between replicas.	Data

Table 5: Summary of the MIFAS, 80/20 principle, cloud middleware, and MOSAIC systems

System/ characteristics	*Techniques used*	*Compatible cloud manager*	*Prog. model*	*Solved problem*	*Type of fault handled*
MIFAS	Replication and self-healing	OpenNebula	Not informed by the authors	Data transfer (images)	Data
80/20 principle	Replication, job migration, and job resubmission	OpenStack	Java	Failure prevention; improving reliability and availability in large-scale distributed systems	Data
Cloud Middleware	Replication, checkpointing, handling user-defined exceptions	OpenStack and OpenNebula	C++	Resource optimization; real-time resource sharing usage information	Data; virtualization; law Suit
MOSAIC	Workflow replication and Rescue	OpenNebula and OpenStack	Java	Best value for money, multi-cloud deployment, authentication (prevention)	Communication, process and data

mOSAIC are application, programming, monitoring, and implementation.

Pilot Data

Pilot-Data addresses fundamental data and computing co-placement and scheduling challenges in heterogeneous and distributed environments with interoperability and extensibility as first-order concerns. It also relies on the reliability features built into the transfer service that automatically restart failed transfers. Table 6 summarizes the main characteristics of the system.

Failures–related to Communication

In this section, proposals to tolerate failures related to communication will be described. The proposals listed here address these failures by checkpointing, self-healing, preemptive migration, retry, handling user-defined exceptions, workflow rescue, job migration, or replication. Some of these proposals were presented in previous sections or will be presented later and only be named here.

Use of Checkpointing

With checkpointing, it is possible to deal with problems related to data traffic so that there is no need to restart data transfer processes completely, for example. In this category, the following approaches can be highlighted: A system for redeeming unexpected spots using heterogeneous spot instances and overprovisioning (Zhang *et al.,* 2019), and SymVirt, presented in Table 6.

ONHelp

The system presented by (Sambath *et al.,* 2019) assists OpenNebula with issues such as security, VM monitoring, fault tolerance, and secure storage.

Regarding fault tolerance, the service deals with software and VM-related failures handled using three mechanisms: lightweight VM checkpoint, VM hot backup, and virtual cluster collaborative backup. Table 6 summarizes the main characteristics of the system.

System for rescuing unexpected spots using spot instances heterogeneous and overprovisioning

The system reliably auto-scales web applications using heterogeneous peer instances and on-demand instances.

The system tolerates failures using rescues from unexpected locations that use heterogeneous point instances and over-provisioning. Summarizes the main characteristics of the system (Table 6).

Symbiotic Virtualization (SymVirt)

SymVirt allows a VM to cooperate with a message transfer layer in the guest operating system. It works as a mediator that enables hot migration using the fault tolerance mechanisms SymCR and SymPFT, which may or may not work together as needed.

Use of Preemptive Migration

Thanks to preemptive migration, it is possible to carry out the anticipatory migration of a task, which allows the possibility of failure prevention treatment. In this category, the Prototype approach for automatic treatment of anomalies, presented in the previous section, and the mantis, *fault-tolerant stateful firewall*, Advanced Access Control System, Prototype for Systematic Network State Extraction, and pFTree-Ext and pFTree-Wt, presented below.

mantis

Despite the system being compatible with several clouds, the authors (Premalatha & Sujatha, 2021) tested only on OpenStack (Table 7).

Table 6: Summary of the Pilot Data, ONHelp, system for rescuing unexpected spots, and symVirt systems.

System/ characteristics	*Techniques used*	*Compatible cloud manager*	*Prog. model*	*Solved problem*	*Type of fault handled*
Pilot-Data	Replication and repeat	Eucalyptus and OpenStack	Not informed by the authors	Resource allocation	Data
ONHelp	Checkpointing, job migration, user-defined exception handling, preemptive migration	OpenNebula	Not informed by the authors	Deals with various failures in general	Process, communication, and virtualization
System for rescuing unexpected spots	Checkpointing, job migration, workflow replication, and rescue	Amazon EC2	Java	Reducing the financial cost of cloud resources and ensuring high availability	Communication, process
SymVirt	Replication, checkpoint, and Retry	Universal	fortran	Migration and communication between VMM and guest OS communication and virtualization	Allows VM hot migration; easy implementation; API

Table 7: Summary of the mantis, FT-FW, systems.

System/ characteristics	*Techniques used*	*Compatible cloud manager*	*Prog. Model*	*Solved problem*	*Type of fault handled*
Mantis	Preemptive migration	Universal	Ruby and Python	Load balancing in optical network environments	Communication
FT-FW	Preemptive migration	Universal	Not informed by the authors	Fault tolerance for firewalls	Communication

Fault-Tolerant Stateful Firewall (FT-FW)

Literature (Sambath *et al.,* 2019) designed a firewall system with fault tolerance support. Table 7 summarizes the main characteristics of the system.

This section presents the results obtained through the analysis of selected articles in SLR. We also sought to identify whether the proposal was geared towards a specific manager whenever possible. Although this information was only sometimes available, it was possible to identify that most of the proposals present solutions compatible with the OpenStack manager (Premalatha & Sujatha, 2021).

Conclusions

The classification was made to create a document that can help people decide which solution for handling failures in a cloud computing system is more adequate or efficient for the problem faced. A brief introduction was included for each solution described, which aims to explain the purpose of creating the solution, and a summary table that aims to objectively show the main information about it for quick reference. The work also provides information regarding the state of the art (in general) regarding fault tolerance in cloud computing environments. Regarding the objectives proposed in this work, it can be considered that they were achieved since the objective description of each solution was presented, in addition to the creation of a summary table with information that seeks to help in choosing a fault-tolerant system, concluding with the creation of a web site to consult the information in the summary tables in a more practical and accessible way.

Acknowledgment

We are thankful to the management of Emerald Heights College For Women, Ooty, India, and Dr.G.R. Damodaran College of Science, Coimbatore, India, for conducting this collaborative study.

Conflict of Interest

Authors have no Conflict of Interest.

References

Anton Gulenko, Marcel Wallschläger, Florian Schmidt, Odej Kao, Feng Liu. (2016). A system architecture for real-time anomaly detection in large-scale nfv systems. Procedia Computer Science, Elsevier, 94, 491–496.

Datta, A.; Cho, W. H. Swifter. (2016): Elastic erasure coded storage system. In: IEEE. Reliable Distributed Systems (SRDS), 2016 IEEE 35th Symposium on. [S.l.], 229–238.

Gonzalez, J. L. et al. (2015). Skycds: A resilient content delivery service based on diversified cloud storage. Simulation Modelling Practice and Theory, Elsevier, 54, 64–85.

Hideya Nakanishi, Ohsuna Masaki, Kojima Mamoru, Imazu Setsuo, Nonomura Miki, Emoto Masahiko, Yamamoto Takashi, Nagayama Yos. (2014). Revised cloud storage structure for light-weight data archiving in lhd. Fusion Engineering and Design, Elsevier, 89(5), 707–711.

J.L. Gonzalez, Jesus Carretero Perez, Victor Sosa-Sosa, Juan F. Rodriguez Cardoso, Ricardo Marcelin-Jimenez. (2013). An approach for constructing private storage services as a unified fault-tolerant system. Journal of Systems and Software, Elsevier, 86(7), 1907–1922.

Javadi, B.; Abawajy, J.; Buyya, R. (2012). Failure-aware resource provisioning for hybrid cloud infrastructure. Journal of parallel and distributed computing, Elsevier, 72(10), 1318–1331.

Javier Povedano-Molina, Jose M. Lopez-Vega, Juan M. Lopez-Soler, Antonio Corradi, Luca Foschini. (2013). Dargos: A highly adaptable and scalable monitoring architecture for multi-tenant clouds. Future Generation Computer Systems, Elsevier, 29(8), 2041–2056.

Lu Lu, Xuanhua Shi, Hai Jin, Qiuyue Wang, Daxing Yuan, Song Wu. (2015). Morpho: A decoupled mapreduce framework for elastic cloud computing. Future Generation Computer Systems, Elsevier, 36, 80–90.

Mell, P.; Grance, T. (2011). The NIST definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 19, 23 - 25.

N. Premalatha and S. Sujatha. (2021). An Effective Ensemble Model to Predict Employment Status of Graduates in Higher Educational Institutions, 2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT), Erode, India, 2021, 1-4, doi: 10.1109/ICECCT52121.2021.9616952.

S. Sambath Kumar, S. Devi, (2020). Image Privacy Preservation using AES With Salt Key and Gaussian Blur Algorithms with the Application of Data Perturbation in Cloud, International Journal of Innovative Technology and Exploring Engineering, 9(3), 729-735.

S. Sujatha, N. Sudha Bhuvaneswari and R. Yamuna, (2010). Paradigm for integrating Web Services and agent technology with RSA and Digigeo, 2010 International Conference on Communication and Computational Intelligence (INCOCCI), Erode, India, 603-608.

Selimi M, Lertsinsrubtavee A, Sathiaseelan A, Cerdà-Alabern L and Navarro L. (2019). PiCasso: Enabling information-centric multi-tenancy at the edge of community mesh networks. Computer Networks: The International Journal of Computer and Telecommunications Networking. 164:C. Online publication date: 9-Dec-2019.

Ting Wang, Zhiyang Su, Yu Xia, Jogesh Muppala, Mounir Hamdi. (2015). Designing efficient high performance server-centric data center network architecture. Computer Networks, Elsevier, 79, 283–296.

Vasile, M.-A. et al. (2015). Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing. Future Generation Computer Systems, Elsevier, 51, 61–71.

Vijayakumar, D.; Srinivasagan, K.; Sabarimuthukumar, R. (2015) Fir3: A fuzzy inference based reliable replica replacement strategy for cloud data centre. In: IEEE. Computing and Network Communications (CoCoNet), 2015 International Conference on. [S.l.], 473–479.

Xuyun Zhang, Chang Liu, Surya Nepal, Chi Yang, Wanchun Dou, Jinjun Chen. (2014). A hybrid approach for scalable sub-tree anonymization over big data using mapreduce on cloud. Journal of Computer and System Sciences, Elsevier, 80(5), 1008–1020.