

Enhanced Model of Fault Tolerance in Cloud Computing Architectures

Imran Alam

Sr. Assistant Professor, AI – Barkaat College of Graduate Studies, Aligarh - India

Email: [imranalam504\[at\]gmail.com](mailto:imranalam504[at]gmail.com)

Abstract: *Cloud computing has embarked a revolution in accessing, provisioning and consumption of the information and computing in the ICT industry. It has emerged as a novel paradigm of high performance and large-scale computing that actuates relocation of computing and data from desktops and personal computers to big data centres. Cloud computing refers to manipulating, configuring, and accessing the applications online. It offers online data storage, infrastructure and application and involves both a combination of software and hardware-based computing resources delivered as a network service. Cloud computing's rising popularity has led many organizations to adopt it, streamlining business operations, lowering infrastructure costs, and enhancing agility by making application deployment and modifications easier. Cloud computing, like any computer system, is susceptible to failures. These failures are often a significance of the inherently distributed and complex nature of cloud platforms. Cloud computing systems need to be built for failure to ensure that they continue operating even if the cloud system has an error. The errors should be disguised from the cloud users to ensure that users continue accessing the cloud services and this intern leads to cloud consumers gaining confidence in the availability and reliability of cloud services. In this paper, we propose the use of N+1 Segmental redundancy architecture to design and implement failure-free clouds.*

Keywords: Cloud Computing, Fault Tolerant, ICT Industry, N+1 Segmental Redundancy

1. Introduction

Cloud computing is an emerging method of computing that is inclined to many challenges due to the nature of its complexity. It is therefore important to understand that cloud systems just like any other complex computing system, will contain errors and experience failures. This does not mean that cloud systems should be ineligible from performing important work, but it does mean that techniques for detecting failures, isolating the failures and understanding the consequences of the failures, and remediating the failures, should be employed and should be a central issue for researchers to understand before the wide-scale adoption of cloud computing systems [1].

Cloud Computing systems are based on distributed systems. [2] States that a key feature of distributed systems is that they should mask failures and continue operating if some aspect of the system fails. An important goal of distributed systems design is to design systems that can automatically recover from partial failures without seriously affecting the overall performance of a system. A system should continue to operate even in the event of a failure while the failure is being fixed [2]. The techniques that are used to create the fault tolerance capabilities in cloud computing can be divided into three categories. Redundancy techniques, fault tolerance policies, and load balancing [5].

The key technique for handling failures in computer systems is redundancy [2]. The redundancy techniques ensure that cloud computing architectures are fault tolerance and they can handle faults and continue operating even in the event of errors.

2. Notion of Dependability

According to [2] being fault-tolerant is muscularly related to being dependability. Dependability, in essence, is about the reliability and trustworthiness of a system or individual. It's

the ability to be counted on to perform as expected, even in the face of challenges or failures. It encompasses multiple attributes, including availability, reliability, safety, and security. Dependability encompasses multiple attributes, including availability, reliability, safety, security, and maintainability. [2] further define these properties as:

- **Availability:** The system's readiness to provide correct service when needed.
- **Reliability:** The continuity of correct service over time, ensuring consistent performance.
- **Safety:** The absence of catastrophic consequences on users or the environment.
- **Security:** The protection of information and systems from unauthorized access or damage.
- **Maintainability:** The ease with which the system can be repaired, updated, or modified.

The dependability of a system according to [6] is defined as “the system’s ability to avoid service failures that are more frequent and more severe than is accepted”. Dependable cloud systems should deliver the correct services. Correct service is delivered when the service implements the system function it was designed for [6]. Service failures occur when a system transitions from correct service to incorrect service.

Many means have been developed to ensure that dependability in computer systems is attained. These means can be grouped into the following categories [6]:

- **Fault Tolerance:** Implement redundancy, backup systems, and failover mechanisms to minimize the impact of failures.
- **Monitoring and Alerting:** Continuously monitor cloud resources and infrastructure to detect and address potential issues proactively.
- **Security Measures:** Implement robust security controls, including firewalls, intrusion detection systems, and access controls, to protect against security threats.

- **Performance Optimization:** Optimize resource allocation, network configuration, and application design to ensure optimal performance and availability.
- **Collaboration and Communication:** Establish clear communication channels and collaboration mechanisms between cloud providers and users to address issues effectively.

Fault prevention and fault tolerance techniques are aimed at ensuring that the services provided by a computer system can be trusted, while fault removal and fault forecasting are aimed at reaching confidence in the ability for services to be trusted by justifying that the functional and dependability specifications are adequate and that the system is likely to meet the specifications and functions [6].

3. Systems Failures

"System failure" are caused by breakdown, malfunction, error, defect, collapse, and crash. [7] define faults as



Figure 1: Adapted from [8]

There are four kinds of system faults, each fault is briefly described below [7]:

- Transient faults:** occur as a result of some temporary condition affecting the system. In cloud computing, this includes conditions such as network connectivity failures and service unavailability. Transient faults disappear as soon as they are rectified. Transient faults can also be resolved by restarting system components.
- Intermittent faults:** occur randomly at irregular intervals and they normally resemble malfunctioning of a system, hardware device, or component. Intermittent faults are extremely difficult to diagnose and resolve permanently. An example could be a hard disk that stops functioning as a result of temperature fluctuations but it returns to normal at some stage.
- Permanent faults:** These faults continue to exist until the root cause of the fault is found and resolved. These faults normally occur as a result of a complete malfunction of a system component, and it is normally straightforward to diagnose permanent faults.
- Byzantine faults:** These faults are the hardest to detect. With Byzantine Faults, a system component might behave illogically and provide incorrect results to the client. This can be a result of a corrupted internal state of a system, data corruption, or incorrect network routes. They are extremely hard and expensive to handle.

4. Building Fault - Tolerant Cloud Systems:

The previous sections have introduced the basic concepts of dependability and the need for dependable cloud systems. This section will delve deeper and look at two main

“a fault is a defect or problem in a system that causes it to fail or act abnormally. A fault is the fundamental damage of the normal system operation, faults cause errors.”

Tolerance techniques need to be implemented or built into a system to ensure that failures are camouflaged and the system is failure-free and this is true for cloud systems. So, the fault tolerance systems are developed. Fault tolerance is a system's ability to continue operating effectively, even in the face of hardware or software failures [7].

Fault tolerance is critical for cloud systems to permit cloud users to continue accessing the needed cloud services even in the presence of cloud failures [8]. The figure below represents how faults cause errors and this, in turn, results in system failure.

approaches that are used to build failure-free cloud systems. Understanding failure in cloud systems helps cloud service providers and cloud system developers build cloud systems that are resilient and able to continue even if there is a failure. Building cloud systems for failure involves using fault tolerance techniques to ensure that cloud systems continue operating even in the event of a failure. [9] theorises that fault tolerance ensures more availability and reliability of cloud services during application execution.

systems that may occur either due to hardware failure or software faults [8]. Fault tolerance is especially crucial in cloud platforms as it gives assurance regarding the performance, reliability, and availability of cloud applications and services. It is therefore important to understand fault tolerance in order to build failure-free cloud systems. Fault-tolerant systems can continue responding to client requests even when certain parts of a system are experiencing failures. There are two common approaches used to build resilient cloud systems. The first approach is the reactive approach. With this approach, the influence of the failure on the cloud system is decreased after the fault or failure has occurred [8]. The following techniques are used to achieve reactive fault tolerance:

Check Pointing: Regular snapshots of the system state are taken, allowing for a rollback to a stable state in the event of a failure [11]. This method acts as a time machine for the system, where it can "go back in time" to a moment before things go wrong. By periodically saving the state of the system, checkpointing minimizes data loss and recovery time, facilitating a quick return to normal operations after a fault is detected [12].

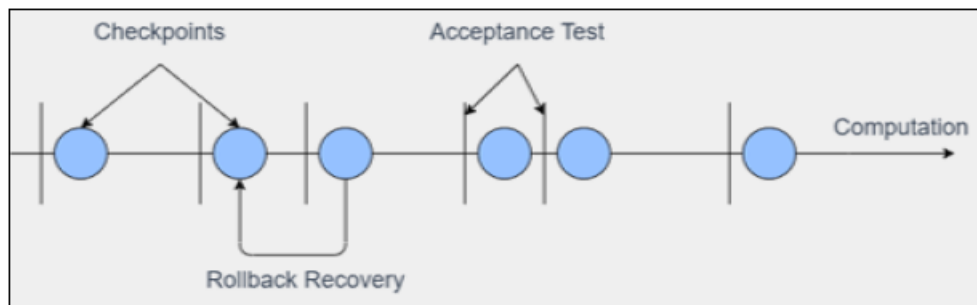


Figure 2: Adapted from [12]

Replication: Data replication across different geographical locations ensures that a copy of the data is always available, even if one site goes down [7]. This is crucial for disaster recovery and maintaining data availability [8].

BIST (Built – In – Self - Test): This technique enables systems to conduct automatic diagnostics to detect hardware failures promptly [10]. By regularly checking their own health, systems can identify potential issues before they escalate, ensuring that maintenance can be performed proactively rather than reactively. This self-awareness is key to minimizing downtime and maintaining system integrity [15].

TMR (Triple Modular Redundancy): A method where three systems run in parallel; if one fails, the other two can continue to provide uninterrupted service [9]. This redundancy ensures that the system remains operational even in the face of hardware failure, making it an essential strategy for critical applications where downtime is not an option. The automatic failover process ensures a seamless transition with no service interruption [8].

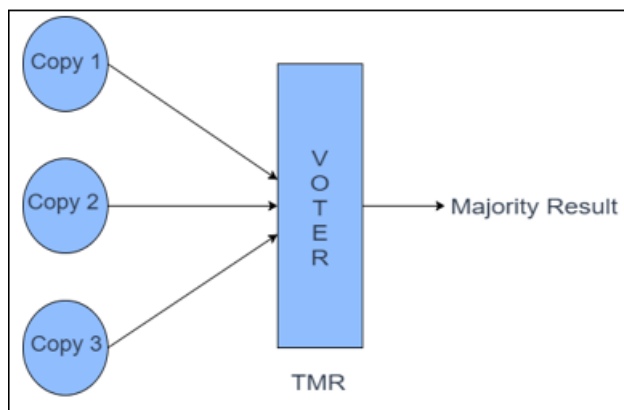


Figure 3: Adapted from [24]

Load Balancing: In computing, load balancing is the process of distributing a set of tasks over a set of resources (computing units), with the aim of making their overall processing more efficient. Load balancing can optimize response time and avoid unevenly overloading some compute nodes while other compute nodes are left idle. [16]. In cloud environments, load balancing is used to distribute workloads across multiple virtual machines and ensure high availability and scalability. These resources could be virtual machines, physical servers, and frameworks [17]. Load balancing mechanisms are considered to be one of the best fault-tolerance methods in cloud computing because these methods provide easy logical resource management in cloud computing environments [16]. Load balancing allows for

easy scaling of applications by adding or removing servers as needed. Load balancers use different algorithms to distribute traffic, such as round-robin, least connections, and IP hash.

Self-Healing: Self-healing in cloud computing refers to systems that can automatically identify and resolve issues, like performance degradation or component failures, without human intervention [7]. Numerous instances of the same application run on various Virtual Machines; this ensures that the failure of the application's instances are handled automatically. This method permits the cloud system to recognize and heal from problems occurring, without depending on the administrator [8].

Rescue Workflow: Rescue workflow techniques are aimed at solving fault tolerance for workflow-based systems. The workflow is allowed to continue even if the task fails until it becomes impossible to continue without attending to the failed task [7].

Preemptive Migration: This technique prevents the system components that are about to fail from impacting the performance of the system [7]. This is achieved through monitoring and by moving components away from nodes that are about to fail and run them on more stable nodes. This method uses a feedback loop to constantly monitor and analyse applications for failure [8].

Job Migration: This technique involves migrating tasks that have failed to execute on a specific physical resource because of component failure onto a different physical resource [8].

A third method for building fault-tolerant cloud systems called the Resilient Fault Tolerance method. This approach is proposed by [7]. Resilient fault tolerance methods involve building intelligent cloud systems. These cloud systems can predict failure in the cloud system and respond to failure automatically without human intervention. These smart cloud systems can learn failures and the appropriate response to the failure by using Machine Learning and Artificial Intelligence. The use of resilient fault tolerance techniques for cloud systems is leading to the development of cloud systems that are termed smart clouds [7].

All the techniques mentioned above are aimed at developing architectures and services that are highly available, scalable, secure, and fault-tolerant [22] to ensure that the services provided by cloud systems are failure-free. These approaches bring about a justifiable level of trust in the cloud systems and services provided by cloud service providers.

5. Related Work

The previous sections introduced the concepts of dependability and fault-tolerance as a means to build failure-free clouds. This section will look at what other researchers have written or done concerning the subject of fault-free cloud computing architectures.

In [5] Fault Tolerance Policies (Reactive and Proactive Policies) are suggested as a means of building failure-free clouds. The policies in [5] can be applied to hardware and software. These techniques are quite complex to implement and might introduce points of failure in a cloud system due to the nature of their complexity.

In [18] the use of an Artificial Neural Network for fault detection and a heartbeat detection strategy is proposed. This is rather complex and might need specialized skills to manage. This complexity in itself makes it difficult to manage this implementation. [22] define a heartbeat as a cluster management software that enables a cluster infrastructure to identify its hosts, active and inactive, by periodic message exchanges.

In [19] a hybrid cloud using open-source software to implement fault tolerance is proposed. [21] used an architectural approach to effectively represent and analyse fault-tolerant software systems. This solution relies on exception handling to tolerate faults associated with component and connector failures, architectural mismatches, and configuration faults. The approach by [21] is more focused on software systems.

Antifragility is a technique proposed by [24], the proposal by [24] suggests the use of failure induction techniques which are comprised of monitoring and learning mechanisms. Antifragility is a phenomenon that proposes that systems can be strengthened when they are exposed to aberrant conditions in their operating environment. This technique is not very different from chaos engineering which proposes the experimental injection of faults into systems in production so as to observe the behaviour of the system and come up with resilient solutions that enable the system to operate under aberrant conditions [25]. Through exposure to shocks and knocks the system is able to adjust and adapt to these conditions [24].

[25] posits that chaos engineering is one technique that can be used to build reliable cloud systems by injecting faults in the system while it is in production. Chaos engineering is widely used at Netflix, [25] further states that other cloud service providers use this technique but use other names to describe the same phenomenon.

6. N-Segmental Redundancy

The previous section looked at some of the research that has been done in terms of fault tolerant computing. These techniques are rather complex and bring in overheads in terms of processing and the cost of the

infrastructure and daily operation. This section will focus on redundancy as a solution for providing failure free clouds.

Redundancy implementation topologies are discussed. Cloud computing is distributed in nature and this in itself brings in complexity. We propose the use of N+1 Segmental redundancy especially for hardware redundancy.

Redundancy may be applied to hardware, information, and software that governs the operations of a cloud system. Various configurations of redundant system design may be used based on the cost, performance, associated risk, and management complexity. These configurations take various forms, such as N, N+1, N+2, 2N, 2N+1, 2N+2, 3N/2. These multiple levels of redundancy topologies are described as N-Segmental Redundancy [23]. [23] further breaks down the building blocks of N-Modular redundancy by describing them as follows:

- 1) *In N-Segmental Redundancy (NSR), N refers to the bare minimum number of independent components required to successfully perform the intended operation.*
- 2) *N+X refers to a redundant system that contains X number of spare components to act as an independent backup when the primary component fails to operate as intended.*
- 3) *YN refers to the number of times the capacity is available to replace the entire set of original components.*
- 4) *YN+X refers to the combination of the above two topologies (N+X and YN).*
- 5) *AN/B refers to a shared redundancy topology, where A amount of backup capacity is available for total B amount of load or original components.*

The choice of the redundancy topology largely depends on what level of availability is required. Redundancy brings in extra cost and complexity in cloud systems. It is critical that organisations consider the level of availability required and they should be able to take on the risk of increased cost and system complexity.

Redundancy systems may offer Active, Passive, Load Sharing, or Standby configuration. Active redundancy means that the redundant component is operating simultaneously with the primary component, but the secondary component or node is only used when the primary component fails. The Passive component is switched on only after the original component fails. The Standby redundancy component fills in the availability gap temporarily until the issue with the primary component is resolved. Additional load-sharing redundancy may be applied to offer partial redundancy in meeting the necessary resilience goals in an event that there is an increase in workloads leading to the stressing of the cloud system [23]. The load balancing mechanism should detect this increase and distribute the workload to other standby nodes to ease the stress on the cloud system. This will in turn lead to improved quality of service in terms of the response time experienced by the cloud consumer.

7. Architectural Diagrams

The above sections have mostly given descriptions of the N-Segmental Redundancy architectures. In this section, we present the architectural diagrams of some of the N-Segmental Redundancy architectures as a way of qualifying the concepts described above. In this section, we present the architecture when N=1, N+1 and 2N+1. When N=1, the failure of the Virtual Machine (VM1) will mean that the cloud system will not be

accessible. Users will not be able to perform any computing functions on the system until VM1 is restored.

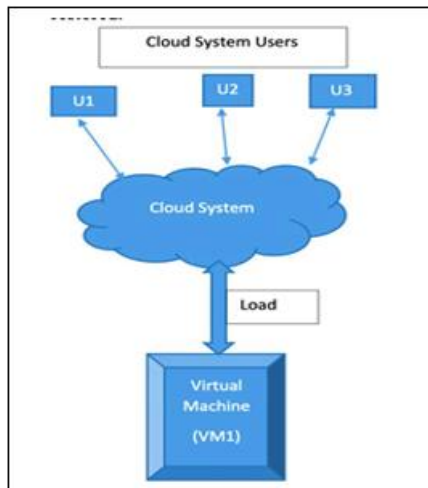


Figure 4: N=1, adapted from [26] [27]

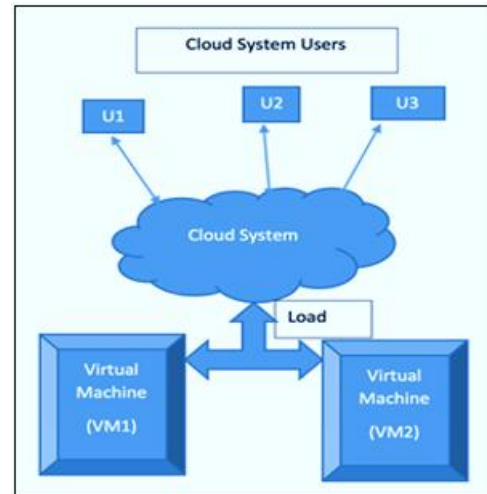


Figure 5: N+1, adapted from [26] [27]

In the figure above, the N+1 architecture is shown. In this architecture, VM2 can either be live or on standby. When both virtual machines are live, the load is shared between the two virtual machines. Data is replicated between VM1 and VM2 so that if VM1 has a problem, the load can be transferred to VM2 and users will continue operating without noticing a significant shift in the performance of the cloud system.

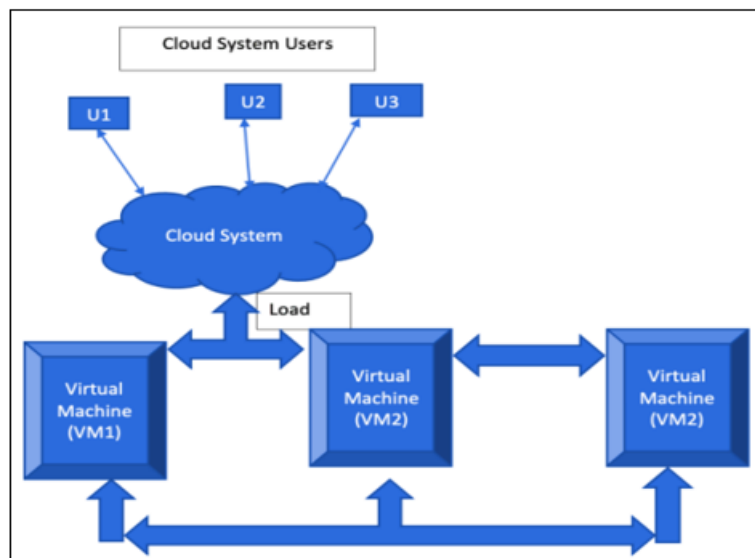


Figure 6: 2N+1, adapted from [26][27]

In the architecture shown above, VM1, VM2 and VM3 are all connected, data is replicated amongst the virtual machines. The load can be shared amongst the three virtual machines to improve the performance of the virtual machine. VM1 and VM2 can be in production while VM3 is on standby, VM3 will move into live status if there is a problem with VM1 or VM2 and in some cases VM1 and VM2 can have a problem at the same time. In such a scenario VM3 will be live while errors on VM1 and VM2 are resolved.

8. Conclusion

In this paper we introduced the notion of dependability in cloud computing and proceeded by explaining the failures that affect computing and the methods that can be used to build failure free clouds. We propose the use of N+1 Segmental Redundancy topologies for designing and

implementing failure free clouds. We also state that the topology chosen depends on the level of availability or dependability that should be gained from the cloud system. We further show examples of the architectural diagrams as proof of concept.

References

- [1] Lee Badger, Tim Grace, Robert Patt-Corner, Jeff Voas, NIST Special Publication 800-146: Cloud Computing Synopsis and Recommendations: May 2012.
- [2] Andrew S. Tanenbaum, Maarten Van Steen, Distributed Systems: Principles and Paradigms, Second Edition, Prentice Hall, Pearson Education, 2007, Upper Saddle River NJ 07458.

- [3] Martin L. Shooman, Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design, 2002 John Wiley & Sons, Inc.
- [4] Antonio Bucchiarone, Henry Muccini and Patrizio Pelliccione, Architecting Fault-tolerant Component-based Systems: from requirements to testing, Electronic Notes in Theoretical Computer Science 168 (2007) 77–90 International Journal of Computer Science & Information Technology (IJCSIT) Vol 14, No 2, April 2022 28
- [5] Ehdi Nazari Cheragh Lou, Ahmad Khadem-Zadeh, and Majid Haghparast, 2015: A SURVEY OF FAULT TOLERANCE Architecture in Cloud Computing, Journal of Network and Computer Applications, <http://dx.doi.org/10.1016/j.jnca.2015.10.004>.
- [6] Algirdas Avizienis, Jean-Claude Laprie, Brian Randall and Carl Landwehr, Basic Concepts and Taxonomy of Dependable and Secure Computing, IEEE Transactions on Dependable and Secure Computing, Volume 1, Number 1, January-March 2004.
- [7] Mukosi A. Mukwevho and Turgay Celik, Toward a Smart Cloud: A Review of Fault-Tolerance Methods in Cloud Systems, Transactions on Services Computing, DOI 10.1109/TSC.2018.2816644, 2018.
- [8] Priti Kumari and Parmeet Kaur, A survey of fault tolerance in cloud computing: Journal of King Saud University Computer and Information sciences, 2018.
- [9] Eman Abdelfattah, Mohamed Elkawagy and Ashraf El-Sisi, A Reactive Fault Tolerance Approach for Cloud Computing, 2017 13th International Computer Engineering Conference (ICENCO), Electronic ISSN: 2475-2320.
- [10] Madhu B. K and Ghamdan. M. Q, Proactive Fault Tolerance for Resilience Cloud Data Centres to Improve Performance Efficiency, International Journal of Engineering Research & Technology, ISSN: 2278-0181, Special Issue - 2016.
- [11] Ao Zhou, Qibo Sun, Jinglin Li, Enhancing Reliability via Checkpointing in Cloud Computing Systems, China Communications 2017.
- [12] Jialei Liu, Shangguang Wang, Ao Zhou, Sathish A.P Kumar, Fangchun Yang, and Rajkumar Buyya, Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability, IEEE Transactions on Cloud Computing, DOI 10.1109/TCC.2016.2567392.
- [13] Zeeshan Amin, Nisha Sethi, Harshpreet Singh, Review on Fault Tolerance Techniques in Cloud Computing, International Journal of Computer Applications (0975 – 8887) Volume 116 – No. 18, April 2015.
- [14] Atul Kumar, Deepti Malhotra, Study of Various Reactive Fault Tolerance Techniques in Cloud Computing, International Journal of Computer Sciences and Engineering, Vol-6, Special Issue-5, Jun 2018 E-ISSN:2347-2693.
- [15] Yong Chul Kwon, Magdalena Balazinska, Albert Greenberg, Fault-tolerant Stream Processing using a Distributed, Replicated File System, 2008.
- [16] Mehdi Nazari Cheraghlou, Ahmad Khadem-Zadeh, Majid Haghparast, A survey of fault tolerance architecture in cloud computing, Journal of Network and Computer Applications, 2015.
- [17] Renu Sharma, Manohar Mishra, Janmenjoy Nayak, Bighnaraj Naik, Danilo Pelusi, Innovation in Electrical Power Engineering, Communication, and Computing Technology, Proceedings of IEPCCCT 2019.
- [18] Amin, Z., Sethi, N., Singh, H., “Review on Fault Tolerance Techniques in Cloud Computing” International Journal of computer Applications, Vol. 116, April 2015, p. 11-17.
- [19] E.M. Hernandez-Ramirez, V.J. Sosa-Sosa, I. Lopez-Arevalo, A Comparison of Redundancy Techniques for Private and Hybrid Cloud Storage, Journal of Applied Research and Technology, Vol. 10, December 2012
- [20] Elena Dubrova, Fault-Tolerant Design, Springer New York Heidelberg Dordrecht London, DOI 10.1007/978-1-4614-2113-9, 2013.
- [21] Rogério de Lemos, Paulo Asterio de Castro Guerra and Cecília Mary Fischer Rubira, A Fault-Tolerant Architectural Approach for Dependable Systems, IEEE Computer Society, 2006.
- [22] Rosangela Melo, Vicente de Paulo F. Marques Sobrinho, Ivanildo José de Melo Filho, Fábio Feliciano, Paulo Romero Martins Maciel, Redundancy Mechanisms Applied in Cloud Computing infrastructures, 2019.
- [23] Muhammad Raza, N, N+1, N+2, 2N, 2N+1, 2N+2, 3N/2 Redundancy Explained, <https://www.bmc.com/blogs/n-n1-n2-2n-3n-redundancy/>, accessed on 2nd July 2021.
- [24] Amal Abid, Mouna Torjmen Khemakhem, Soumaya Marzouk, Maher Ben Jemaa, Thierry Monteil, Khalil Drira, 2014, Toward Antifragile Cloud Computing Infrastructures, 1st International Workshop” From Dependable to Resilient, from Resilient to Antifragile Ambients and Systems” (ANTIFRAGILE 2014), Procedia Computer Science 32 (2014) 850 – 855
- [25] Russ Miles, 2019, Learning Chaos Engineering, First Edition, O'Reilly Media, United States of America.
- [26] Data Centre Redundancy: N+1, 2N, 2(N+1) or 3N2, https://datacenter.com/news_and_insight/data-center-redundancy-2plus1-2n-distributed-redundancy/ last accessed on 28th April 2022.
- [27] An Introduction to UPS Redundancy, <http://www.feace.com/single-post/an-introduction-to-ups-redundancy> last accessed on 28th April 2022.