



e-ISSN: 2319-8753 | p-ISSN: 2347-6710

# IJIRSET

International Journal of Innovative Research in  
**SCIENCE | ENGINEERING | TECHNOLOGY**



# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 10, October 2024

**ISSN** INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA

**Impact Factor: 8.524**

 9940 572 462

 6381 907 438

 [ijirset@gmail.com](mailto:ijirset@gmail.com)

 [www.ijirset.com](http://www.ijirset.com)

# Failure Free Cloud Computing Architectures

Monika<sup>1</sup>, Santhosha<sup>2</sup>

Assistant Professor, Department of Computer Science & Applications, The Oxford College of Science,  
Bangalore, India

MCA Student, Department of Computer Science & Applications, The Oxford College of Science, Bangalore, India

**ABSTRACT:** Cloud computing has gained popularity over the years, some organizations are using some form of cloud computing to enhance their business operations while reducing infrastructure costs and gaining more agility by deploying applications and making changes to applications easily. Cloud computing systems just like any other computer system are prone to failure, these failures are due to the distributed and complex nature of the cloud computing 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 masked from the cloud users to ensure that users continue accessing the cloud services and this in turn leads to cloud consumers gaining confidence in the availability and reliability of cloud services. In this paper, we propose the use of N-Modular redundancy to design and implement failure-free clouds.

**KEYWORDS:** Cloud Computing, Cloud Consumer, Failure Free, N-Modular Redundancy

## I. INTRODUCTION

Cloud computing is an emerging method of computing that is prone 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 flaws and experience failures. This does not mean that cloud systems should be disqualified 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 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 failure-free and they can handle faults and continue operating even in the event of errors.

## II. THE CONCEPT OF DEPENDABILITY

According to [2] being fault-tolerant or failure-free is strongly related to being dependable. Dependability is a term that covers several useful properties for distributed systems. [2] further define these properties as:

- **Availability:** This is the probability that a system is operating correctly at any given point in time.
- **Reliability:** This refers to the property of a system being able to run continuously without failure.
- **Safety:** This refers to the situation when a system temporarily fails to operate correctly without catastrophic consequences.
- **Maintainability:** This refers to how easily a failed system can be repaired and brought back to a state of operating correctly.

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 four categories [6]:

- **Fault Prevention:** This involves the means to prevent the occurrence of failures.
- **Fault Tolerance:** This involves the means to avoid service failures in the presence of faults.
- **Fault Removal:** This involves the means to reduce the impact of faults.
- **Fault Forecasting:** Involves the means to estimate the present and future fault incidents and the likely consequences of the faults.

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].

### III. SYSTEMS FAILURES

System failures are a result of faults that are caused by errors. [7] define faults as “any event that occurs in a system which impacts the normal operation of the system. A fault is the fundamental impairment of the normal system operation, faults cause errors.”

Fault tolerance is crucial 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.

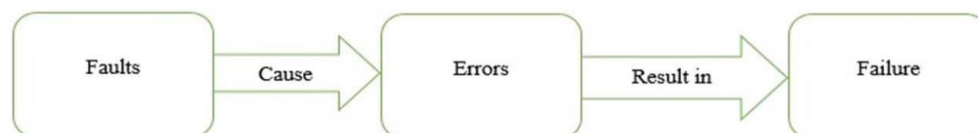


Figure 1. Adapted from [8]

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.



#### IV. BUILDING FAILURE-FREE 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 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.

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:

- **Checkpointing:** This technique involves saving the system state at a certain point in time. If for some reason the system fails, it can still roll back to the point or state that was last saved.[11]. When failures occur, tasks can resume on a different physical machine based on the last saved checkpoint image. In other words, the tasks need not be restarted from the beginning but only from the last saved state. Checkpointing can reduce lost time due to physical machine faults and improve cloud service reliability [12].
- **Retry:** With the retry approach, a task or request is constantly executed until it is processed[8]. This approach is suitable for transient failures. Retry is the simplest of all fault tolerance mechanisms[13]. It is simple because the cloud user resubmits the request until it is executed on the same cloud resource. The retry approach works by simply retrying a failed request on the same resource multiple times [7].

With the retry method, a retry mechanism is applied to recover from the effects of a fault. This mechanism makes the defective module retry its activity for some time. If the fault lasts longer than the retry period, then it is considered a permanent fault. The retry period should be long enough to make the transient fault disappear and short enough to avoid overlapping of faults [14].

- **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].
- **Task Resubmission:** With task resubmission, a failed task is detected and resubmitted to the same or different cloud resource for execution [7]. Task resubmission and replication techniques are widely used techniques in distributed systems for fault tolerance. This technique consumes a lot of system resources since failed tasks are continuously being resubmitted to the same or different cloud resources [7]. Task resubmission enables a system to continue working after the failure of the task until it will not be able to proceed without amending the fault [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].
- **SGuard:** SGuard is based on rollback recovery. The use of the checkpointing technique for fault tolerance is expensive, SGuard improves efficiency by best utilization of resources through new file systems like the GFS(Google File System) and the HDFS(Hadoop Distribution File System)[14]. SGuard checkpoints the state of stream processing nodes periodically and restarts failed nodes from their most recent checkpoints. Unlike other fault-tolerance methods, SGuard performs checkpoints asynchronously by continuously checking operators processing streams during the checkpoint and in turn reducing the potential disruption due to the checkpointing activity [15].
- **Self Healing:** This technique enables the cloud system to automatically detect, diagnose and repair software and hardware faults. Such systems are made up of multiple components that are deployed on multiple Virtual Machines[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].
- **Software Rejuvenation:** This method is designed for periodic restarts[7]. In this approach, the system undergoes periodic reboots and begins from a new state every time [8].

- **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 analyses applications for failure [8].
- **Load Balancing:** This approach is used to balance the load of memory and CPU when it exceeds a certain limit. The load of exceeded CPU is transferred to some other CPU that does not exceed its maximum limit.[8]. Load balancing is not limited to just CPU and memory, load balancing fault tolerance techniques can be implemented via hardware, software, and networks[16]. Load balancing in cloud computing is based upon the distribution of workloads. It balances all workload requests by various resources. 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

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.

## V. 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 [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 strengthen 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 behavior 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].

## VI. N-MODULAR 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-Modular 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-Modular Redundancy [23].

[23]further breaks down the building blocks of N-Modular redundancy by describing them as follows:

1. In N-Modular Redundancy(NMR), 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.

### VII. ARCHITECTURAL DIAGRAMS

The above sections have mostly given descriptions of the N Modular redundancy architectures. In this section, we present the architectural diagrams of some of the N Modular 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.

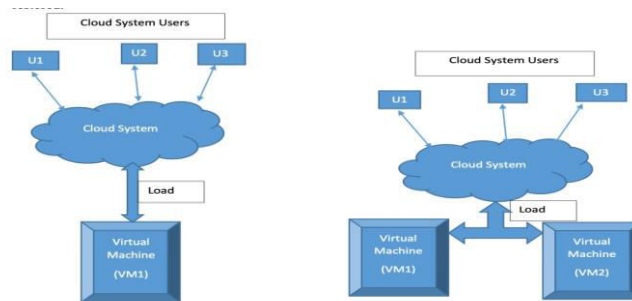


Figure 2. N=1, adapted from [26] Figure 3. N+1, adapted from [26]

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.

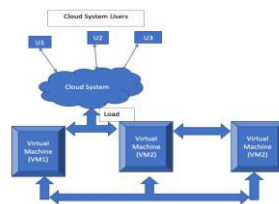


Figure 4. 2N+1, adapted from[26][27]

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.

### VIII. CONCLUSION & RESULT

In this paper we introduced the concept of dependability in computer systems and proceeded by explaining the failures that affect computer systems and the methods that can be used to build failure free clouds. We propose the use N-Modular redundancy topologies when 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 Grance, Robert Patt-Corner, Jeff Voas, NIST Special Publication 800-146: Cloud Computing Synopsis and Recommendations: May 2021.
2. Andrew S. n Education,2017, 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 PellTanenbaum, Maarten Van Steen, Distributed Systems: Principles and Paradigms, Second Edition, Prentice Hall, Pearsoiccione, Architecting Fault-tolerant Component-based Systems: from requirements to testing, Electronic Notes in Theoretical Computer Science 168 (2019) 77–90
5. Ehdi Nazari Cheraghloou, Ahmad Khadem-Zadeh, and Majid Haghparast,2021:A SURVEY OF FAULT TOLERANCE Architecture in Cloud Computing, Journal of Network and Computer Applications,http://dx.doi.org/10.1016/j.jnca.2019.10.004.
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.2018.2816644,2022.
8. Priti Kumari and Parmeet Kaur, A survey of fault tolerance in cloud computing: Journal of King Saud University Computer and Information sciences,2019.
9. Eman AbdElfattah, Mohamed Elkawkagy and Ashraf El-Sisi,A Reactive Fault Tolerance Approach For Cloud Computing, 2019 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 Centers to Improve Performance Efficiency, International Journal of Engineering Research & Technology, ISSN: 2278-0181, Special Issue - 2021.
11. Ao Zhou, Qibo Sun, Jinglin Li, Enhancing Reliability via Checkpointing in Cloud Computing Systems, China Communications 2018.
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 2018.
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.





INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA



# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 9940 572 462  6381 907 438  [ijirset@gmail.com](mailto:ijirset@gmail.com)



[www.ijirset.com](http://www.ijirset.com)

Scan to save the contact details