THE USA JOURNALS
THE AMERICAN JOURNAL OF ENGINEERING AND TECHNOLOGY (ISSN – 2689-0984)
VOLUME 06 ISSUE09

PUBLISHED DATE: - 12-09-2024
DOI: - https://doi.org/10.37547/tajet/Volume06Issue09-06 PAGE NO.: - 38-44

RESEARCH ARTICLE Open Access

METHODS FOR ENHANCING FAULT TOLERANCE IN SYSTEMS WITH HYBRID ARCHITECTURE

Kuzevanov Igor
Senior Member of Technical Staff @ Oracle, Santa Clara, California, USA

Abstract
The article discusses methods for increasing fault tolerance in systems with a hybrid architecture that includes elements of cloud technologies, local servers and peripheral computing. The main attention is paid to the analysis of vulnerabilities and risks inherent in such systems, as well as practical methods of their elimination. Both traditional methods such as hardware duplication and software redundancy are discussed, as well as modern approaches including the use of sharding, data replication, load balancing and integration of hardware and software. The article also highlights the importance of using preventive measures to minimize the risks of failures and the use of tools for modeling and testing failures. The work focuses on the need for an integrated approach to ensuring fault tolerance, taking into account both technical and economic aspects.

Keywords Fault tolerance, hybrid systems, hardware duplication, software redundancy, sharding, data replication, load balancing, failure modeling, failure testing.

INTRODUCTION

Modern information systems increasingly utilize hybrid architectures that combine elements of cloud technologies, on-premises servers, and edge computing. These systems enable organizations to manage data and computing resources more efficiently by distributing workloads across various components, thereby providing high flexibility and scalability. However, as the complexity of such architectures grows, so does the number of potential vulnerabilities, making the issue of ensuring fault tolerance critically important. Fault tolerance, defined as the system's ability to continue functioning in the event of partial or complete failures, becomes a crucial aspect in the design and operation of hybrid systems.

The relevance of this study is driven by the growing demands for reliability and continuity of business processes, particularly in the context of intensive use of cloud technologies and distributed computing.

The aim of this work is to analyze existing methods for enhancing fault tolerance in systems with hybrid architecture, identify their advantages and disadvantages, and offer recommendations for their optimal application in modern information environments.

1. Types of Failures and Vulnerabilities in Hybrid Systems

The threats and vulnerabilities that arise in distributed systems are associated with risks that can jeopardize the stability and security of these systems. Vulnerabilities are weaknesses within the system, including software bugs, ineffective security measures, or unreliable passwords, which can be exploited by malicious actors to attack the system. Threats, in turn, represent potential negative consequences of exploiting these vulnerabilities, such as unauthorized access, data breaches, or system failures. Ensuring security in distributed systems is of paramount importance given their complexity, high level of integration, and frequent use for critical tasks.

Since distributed systems form the backbone of many essential services, such as online banking, healthcare, and telecommunications, disruptions caused by security threats can have severe consequences. For example, DDoS attacks, which overwhelm the system with fake traffic, can render it inaccessible to legitimate users.

The security of distributed systems also directly impacts customer trust in an organization. Any breach, particularly those involving data leaks, can severely damage a company's reputation, leading to financial losses and a loss of customer trust.

Distributed systems face a variety of threats, including denial-of-service attacks (DoS/DDoS), data interception and eavesdropping, insider threats from employees, and infections by malware and ransomware. Each of these threats can have serious consequences, including data loss and disruption of the organization's operations [1].

Hybrid systems introduce new threat vectors not present in traditional IT environments. In particular, data transmitted between cloud and on-premises components may be at risk of interception or tampering. Moreover, the complex structure of hybrid systems increases the number of potential attack points, giving malicious actors more opportunities to exploit vulnerabilities.

To ensure comprehensive protection of hybrid systems, various penetration testing methodologies are employed. Each of these methods focuses on specific aspects of the system and allows its security to be assessed from different perspectives. The combined use of these methods enables the most accurate identification of potential vulnerabilities and enhances the level of protection [2]. Table 1 presents the existing types of penetration testing.

Table 1. Types of Penetration Testing [3].

Type of Testing	Description
Black Box	This method involves testing the system without prior knowledge of its internal structure, simulating the actions of an external attacker. The goal is to gain unauthorized access and identify vulnerabilities without the involvement of developers.
White Box	Unlike the black box method, white box testing is conducted with full access to system information. This method includes analyzing the source code, architecture, and infrastructure, allowing for the identification of vulnerabilities at the code and configuration levels.
Gray Box	This method is a compromise between black box and white box testing. The tester has limited knowledge of the system, allowing them to focus on identifying

	vulnerabilities in specific components while remaining relatively independent of the developers.
Physical Penetration Testing	This method focuses on the physical security of the system, including attempts to access hardware or facilities where servers and other critical components are housed.
Web Application Testing	This approach aims to identify vulnerabilities in web interfaces, such as SQL injection or cross-site scripting, which could be used by attackers to compromise the system [3].

2. Traditional Methods for Enhancing Fault Tolerance

Fault tolerance refers to the ability of a system or its components to continue functioning after a partial or complete failure. There are several levels of fault tolerance: zero, high, and low. At the zero level of fault tolerance, any critical point of failure leads to a complete shutdown of the system. An example would be a situation where the failure of a single component, such as a valve core, renders the entire system non-functional. A high level of fault tolerance means that the system can continue operating even if individual components fail, as seen in facilities with backup generators that maintain lighting and power supply when the main energy source is lost. A low level of fault tolerance provides only the minimally necessary functions, such as emergency lighting and elevator operation.

The goal of designing for fault tolerance is to minimize risks to people and reduce potential damage to property. Despite the aim to enhance fault tolerance, it is important to consider the trade-offs related to costs and technical capabilities. By analyzing the criticality of systems, the likelihood of failure, and the associated costs, it is possible to determine which system components require increased reliability. Using the right EAM (Enterprise Asset Management) software helps simplify the process of data collection and facilitates informed decision-making regarding fault tolerance [4].

Fault tolerance is therefore of paramount importance in the design and deployment of modern information systems.

First, it ensures the continuity of critical services, which is especially important for organizations where any downtime can lead to significant financial losses or damage to reputation.

Second, fault tolerance is directly linked to enhancing system reliability. Redundant mechanisms help prevent data loss and minimize the risk of prolonged downtime, which is critical for maintaining operational stability.

Third, fault tolerance contributes to improving system scalability. Automatic load redistribution among redundant components makes it easier for the system to adapt to increased loads while maintaining stable performance. Table 2 below presents the types of fault tolerance at different levels of the system.

Table 2. Types of Fault Tolerance at Different Levels of the System [4].

Type of Fault Tolerance	General Description of Fault Tolerance Type

at Various System Levels	
Server Fault Tolerance	Involves the presence of backup servers that automatically take over workloads in the event of a primary server failure.
Database Fault Tolerance	Involves creating backups of databases that come into play if the primary server fails.
Network Component Fault Tolerance	Achieved by redundant routers, switches, and connections, allowing communication to be maintained even during network failures.
Data Storage Fault Tolerance	Involves replicating data across multiple storage devices, ensuring access even if one device fails.
Application Fault Tolerance	Implemented through routing user requests to backup instances of applications in case of a primary failure.

To successfully implement fault tolerance, careful planning and the deployment of several key strategies are necessary:

1. Redundancy: Incorporating redundant components that can be used as backups ensures the immediate restoration of system functionality.

2. Automated Monitoring: Continuous monitoring of the system's status allows for the prompt detection and resolution of issues before they impact performance.

3. Rapid Recovery: Implementing procedures that enable instant switching to backup components reduces downtime and minimizes the impact on users.

4. Load Balancing: Distributing the load across various resources prevents individual components from being overloaded and maintains stable system operation [5].

Fault-tolerant systems are designed with additional components that can replace failed devices or processes, thereby preventing operational disruptions. The main elements of fault-tolerant systems are outlined in Table 3.

Table 3. The Main Elements of Fault-Tolerant Systems [6].

Elements of Fault-Tolerant Systems	**Description of Elements**
Diversification	Diversification involves creating backup components that operate independently of the primary ones. This can include using alternative power sources in case the primary source fails, allowing the system to continue functioning even under extreme conditions.
Redundancy	The introduction of redundant elements helps minimize the impact of failures. For instance, a system with multiple power supplies, where each supply can maintain system operation, ensures continuous functioning even if one power supply fails.
Replication	Replication is the method of creating copies of the system or its components that operate in parallel and deliver identical results. If one copy fails, another

	continues to perform the tasks, minimizing the risk of data loss or operational downtime. This approach can be implemented at both the component level and the entire system level, creating duplicate structures to enhance reliability [6].

3. Modern Approaches and Tools for Ensuring Fault Tolerance in Hybrid Systems

One of the most effective methods for increasing a system's resilience to failures is hardware duplication, which involves having multiple components that perform identical functions. For example, additional power supplies, memory modules, hard drives, processors, or network interfaces can be used. If one of these elements fails, another can take over, ensuring the system's continuous operation. Hardware-level duplication can be implemented at various levels, including individual components, devices, boards, or even the entire system. However, it's important to note that implementing hardware duplication increases costs, complicates the system, and raises its energy consumption.

An alternative approach to enhancing fault tolerance is software redundancy, which involves having multiple versions of the same software capable of producing identical results. In this context, different algorithms, programming languages, or implementations can be used, which, despite their differences, solve the same task. If one version produces an erroneous result, another can detect and correct it. Software redundancy can be applied at various stages of development, testing, or operation. It is important to note that this approach increases development time, requires additional resources for testing, and increases the amount of memory used.

Another method for ensuring system reliability is the use of error detection and correction mechanisms. These mechanisms can include

checksums, parity codes, error correction systems, or watchdog timers to detect faults in data or signals. Recovery methods after failures may involve rollback, restart, or process replication. These mechanisms can be implemented at the hardware level, software level, or middleware level. However, applying these methods inevitably increases system load, introduces additional delays, and may reduce throughput [7].

To minimize the risk of errors and failures in the system, preventive measures such as applying design standards, coding rules, and quality assurance procedures can be used. These methods help prevent errors in both the hardware and software components of the system. Additionally, load balancing, proper resource allocation, and effective system planning help avoid failures. Implementing preventive measures requires careful planning, in-depth analysis, and optimization at all stages—from design to system operation.

A fifth way to enhance system reliability is through tools designed for modeling and evaluating the system's response to potential failures. These tools can include software and hardware solutions for testing components like memory, disks, or network interfaces for errors. Various metrics and tests can also be applied to assess system reliability and availability in the event of failures. The modeling and testing process requires significant time and resource investments, as well as a high level of expertise.

Lastly, a comprehensive approach to the development and integration of hardware and software is crucial for enhancing system reliability. This approach involves close coordination between these components, which allows for optimizing system performance under failure conditions. For example, adaptive hardware or software methods can be used to configure the system according to the situation. Joint modeling and debugging can also be applied to verify the system's correct operation. However, this approach requires high skill levels and careful coordination among all participants in the system development and implementation process [8].

The choice between these approaches depends on the goals and specifics of the application. Critically important systems, such as financial or medical applications, require maximum resilience and continuous operation. For less critical applications, a strategy of gradual degradation may be acceptable, allowing for resource savings while maintaining a certain level of functionality.

When designing fault-tolerant systems, architects need to define an acceptable level of system survivability. Complete fault tolerance is unattainable, so specific goals must be set, which can vary depending on the application's architecture and its operational environment.

The most common goals include:

1. Node failure tolerance: Deploying software instances across multiple nodes helps minimize risks in the event of hardware failures.

2. Availability zone failure tolerance: Using multiple data centers within a single cloud region protects the application from failures related to the outage of a specific data center.

3. Cloud region failure tolerance: Deploying software instances across different cloud regions provides protection against more widespread failures, such as the outage of an entire region.

4. Cloud provider failure tolerance: Utilizing a multi-cloud architecture or on-premises deployment helps safeguard against a complete service failure from a specific provider.

Additionally, when designing these systems, factors such as acceptable recovery time (RTO) and allowable data loss (RPO) must be considered. These parameters significantly influence the choice of architectural solutions and, consequently, the associated costs.

The costs of ensuring fault tolerance can vary widely. Operating a system designed for high fault tolerance may require significant financial investment; however, these costs must be weighed against the potential losses in the event of a failure. For example, downtime for a critical application could result in substantial financial losses and damage to the company's reputation.

Moreover, it is essential to consider the costs of restoring system functionality after a failure, as well as the impact of downtime on the morale and productivity of the development team.

An important example of a cost-effective approach to fault tolerance is the transition to specialized solutions, such as the use of distributed databases. For instance, a major company’s switch from MySQL to CockroachDB not only improved the fault tolerance of their database but also significantly reduced maintenance costs through the automation of complex processes.

The following practical examples illustrate modern approaches and tools for ensuring fault tolerance in hybrid systems:

Google Infrastructure: Google’s large-scale distributed infrastructure is an example of a highly fault-tolerant system. One of the key strategies is global data replication, which helps reduce latency and increase system resilience. Another important element of Google’s approach is performance isolation, which helps maintain high availability and low latency even under abnormal load conditions.

Cluster management systems like Borg demonstrate Google's commitment to ensuring reliability and availability. Borg combines cluster management tasks with resource allocation optimization and failure recovery, maintaining high service quality for users.

AWS Route 53: Amazon Web Services' (AWS) Route 53 service is an example of a highly available and fault-tolerant system. It uses a network of devices that monitor the health of resources in various regions, enabling a rapid response to failures. If issues are detected, the system automatically switches to alternative resources, ensuring uninterrupted operation.

Route 53 is designed to remain resilient even under widespread failures. Thanks to its well-considered architecture, which includes redundant resources and efficient load distribution, the service maintains stability and high performance under the most challenging conditions [9].

CONCLUSION

In conclusion, ensuring fault tolerance in hybrid systems requires a comprehensive approach that incorporates both traditional and modern methods. To achieve an optimal level of fault tolerance, it is essential not only to consider the architecture and specific operational characteristics of the system but also to carefully analyze potential risks and costs. A multi-layered approach, which includes hardware and software redundancy, sharding, data replication, and other methods, can significantly enhance the reliability and stability of systems, particularly those critical to essential infrastructures. However, complete fault tolerance is unattainable, making it important to establish acceptable goals and levels of system survivability based on its purpose.

REFERENCES

1. Jansen A. C. H., Kana A. A., Hopman J. J. A Markov-based vulnerability assessment for the design of on-board distributed systems in the concept phase //Ocean Engineering. – 2019. – T. 190. – P. 106-448.

2. Al Shebli H. M. Z., Beheshti B. D. A study on penetration testing process and tools //2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT). – IEEE, 2018. – pp. 1-7.

3. Ur-Rehman A. et al. Vulnerability modeling for hybrid IT systems //2019 IEEE international conference on industrial technology (ICIT). – IEEE, 2019. – pp. 1186-1191.

4. Koren I., Krishna C. M. Fault-tolerant systems. – Morgan Kaufmann, 2020. Sharma P., Prasad R. Techniques for Implementing Fault Tolerance in Modern Software Systems to Enhance Availability, Durability, and Reliability // Eigenpub Review of Science and Technology. – 2023. – T. 7. – No. 1. – pp. 239-251.

5. What is Fault Tolerance? 3 Techniques & Definition. [Electronic resource] Access mode: https://www.wallarm.com/what/what-is-fault-tolerance (access date 08/21/2024).

6. Kumari P., Kaur P. A survey of fault tolerance in cloud computing // Journal of King Saud University-Computer and Information Sciences. – 2021. – T. 33. – No. 10. – pp. 1159-1176.

7. What is fault tolerance, and how to build fault-tolerant systems. [Electronic resource] Access mode: https://www.cockroachlabs.com/blog/what-is-fault-tolerance/ (date of access 08/21/2024).

8. Hybrid -system approach to fault-tolerant quantum communication. [Electronic resource] Access mode: https://archive.org/details/arxiv-1209.3851 (date of access 08/21/2024)