Journal of Industrial Engineering and Applied Science
ISSN 3005-6071 (Print) | ISSN 3005-608X (Online) | Vol. 2, No. 2, 2024

# Design of Disaster Recovery and Load Balancing Strategies in Traditional Centralized Distributed Web Systems

**SONG, Qingqing [1*] WU, Zhen [2] XIA, Shaoliang [1]**

[1] Belarusian State University, Belarus
[2] Nanjing University of Aeronautics and Astronautics, China

**** SONG, Qingqing is the corresponding author, E-mail: fpm.sunC@bsu.by***

**Abstract:** This paper proposes an effective disaster recovery strategy and load balancing strategy for web systems based on traditional centralized distributed architecture, which are widely used in small and medium-sized enterprises. In the disaster recovery strategy, we have designed a global backup strategy and single node backup strategy to adapt to different system requirements and fault recovery scenarios. In the load balancing strategy, we take the database as the starting point, calculate the load proportion of task execution nodes, effectively determine the load status of each task execution node, and make the load of each task execution node in the distributed system tend to be balanced. The research results of this paper have certain reference value for improving early developed web systems with the same architecture. Future research will focus on optimizing the efficiency of disaster recovery strategies, improving the accuracy of load judgment, and exploring innovative methods to enhance the scalability and stability of web systems based on centralized distributed architecture.

**Keywords:** Centralized distributed architecture, WEB system, Disaster recovery strategy, Redundancy strategy, Load balancing, Database Load.

**DOI:** https://doi.org/10.5281/zenodo.10836116

## 1 Introduction

With the vigorous promotion of information technology construction on a global scale, the application scope of information technology is increasingly expanding. Modern information technology is not only widely used in emerging technology enterprises, but also beginning to penetrate into traditional industry enterprises. A large number of small and medium-sized traditional industry enterprises have begun to actively explore or have integrated information technology into their human resource management, business processes, resource allocation, and financial management in the form of information management systems [1].

However, due to various limitations such as cost, efficiency, and scale requirements, many small and medium-sized enterprises are still using web systems based on traditional centralized distributed architectures [2]. These types of systems are usually built earlier and have relatively simple functions, thus lacking effective disaster recovery strategies [3] and load balancing strategies [4]. With the increasing number of users and the increasing importance of data in enterprise operations, it has become a widespread demand to transform these web systems based on traditional centralized distributed architecture in order to ensure data security and efficient system operation.

This paper aims to design and implement effective disaster recovery and load balancing strategies for the transformation process of such systems, in order to meet the needs of enterprises for data security and system efficiency. We will delve into the design principles of two strategies, analyze their effectiveness in practical applications, and propose possible optimization directions, in order to provide valuable references for future research and practice.

## 2 Centralized Distributed Architecture

Centralized distributed architecture, also known as centralized star architecture [5], is a special type of distributed system architecture characterized by the presence of a central node responsible for coordinating and managing the workflow and task allocation of the entire system. In this architecture, other nodes act as child nodes, responsible for executing tasks assigned by the central node and returning the results to the central node.

In a centralized distributed architecture, nodes in a distributed cluster can be divided into two main roles based on their roles: task control nodes and task execution nodes. Task control nodes are usually responsible for distributing tasks and monitoring the progress of task execution nodes. When the task execution node is idle, the task control node will assign tasks to it; When the task execution node is unable to function properly, the task control node will remove it from the system and assign its tasks to other task

execution nodes. In addition, task control nodes may only be responsible for generating tasks without assigning them, and each task execution node may spontaneously collect tasks.

The reason why centralized distributed architecture is widely used is that its architecture design is simple. The advantages of centralized distributed architecture are mainly the small number of devices, simple architecture design, low coupling between universality and application, flexible resource scheduling, and easy deployment.

Although the advantage of a centralized distributed architecture lies in its simple architecture design, this does not mean that it has no drawbacks. In fact, there are also some challenges and limitations to a centralized distributed architecture. In task control nodes, if the processing capacity of the task control node is not sufficient to efficiently control the operation of the entire system, the performance and efficiency of the system may be affected. This requires us to fully consider and evaluate the processing ability of task control nodes, especially the judgment of task execution node status, when designing and implementing a centralized distributed architecture. How to handle single point of failure in task execution nodes is also a challenge. If a task execution node fails, it may lead to partial functional failure of the system or affect the efficiency of system operation. To ensure the stability of system functionality and efficiency, we need to set up effective disaster recovery plans for task execution nodes.

# 3 Strategy Design

## 3.1 Disaster Recovery Strategy Design

This paper divides task execution nodes into two types, one is a regular node and the other is a backup node. The function and data of the backup node are consistent with the corresponding regular node.

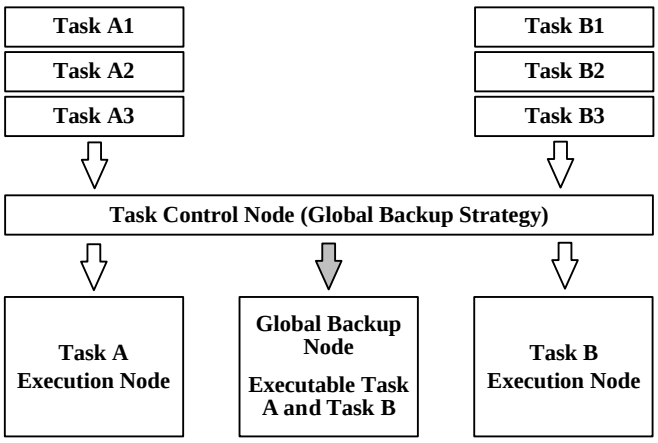


**Figure 1 Global backup strategy**

In order to adapt to different system requirements and fault recovery scenarios, we have designed and implemented two different disaster recovery strategies: global backup strategy and single node backup strategy. Among these two strategies, the design inspiration for the single node backup strategy comes from the RAID1 strategy in independent redundant disk arrays (RAID) [6], which improves data reliability and system recovery ability through mirrored backup.

As shown in Figure 1, in the global backup strategy, we have set up a dedicated global backup node that can perform all the functions of task execution node A and task execution node B. This means that no matter which task execution node fails, we can replace its function with a global backup node to ensure the continuous operation of the system. However, in this strategy, if multiple nodes fail, due to their shared global backup nodes, it will inevitably lead to a decrease in the operational efficiency of the system.

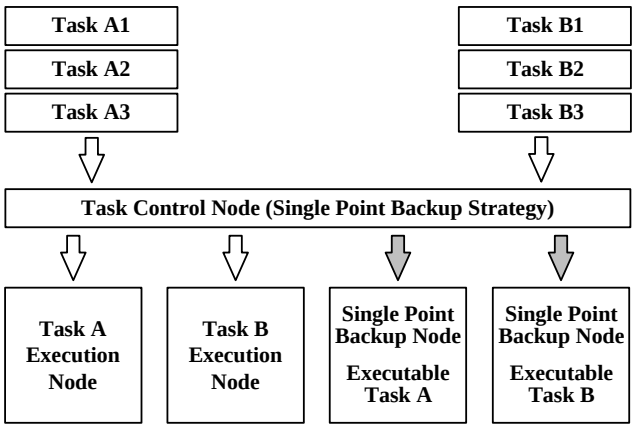


**Figure 2 Single node backup strategy**

In the single node backup strategy, as shown in Figure 2, we set a corresponding backup node for each task execution node. Each backup node is only responsible for implementing the functions of its corresponding task execution node, which means that the function of the backup node is a complete mirror of the task execution node it backs up. This strategy can ensure that the operational efficiency of the system does not decrease in the event of multiple node failures.

In this strategy, attention should be paid to distinguishing whether the task execution nodes in the system are primarily functional services or data storage. If functional services are the main focus, in order to ensure service quality and avoid compatibility issues, a single point backup strategy should be considered; If data storage is the main focus, based on cost considerations, a global backup strategy can be considered. However, it is worth noting that when a node failure occurs, data should be promptly restored from the global backup node to prevent damage to the global backup node.

## 3.2 Load Balancing Strategy

The load balancing strategy is mainly applied to the

task control nodes in the distributed architecture of the system [7], ensuring that when assigning tasks, it can balance the load of each task execution node in the distributed system.

In the current system, most WEB systems belong to query intensive systems, which means that most of the operations performed in the system are query operations on the database.

So when making load judgments on various task execution nodes in the system, the query volume and data volume of each table in the database of that node are used as the main basis for judgment. By conducting performance benchmark tests on the task execution node, the maximum number of queries that can be processed per second for each table in the database of that node at different data volumes is determined. The current number of queries per second in the system is compared to the maximum number of times the system can query each data table per second, and the current load proportion of the system is ultimately determined.

After obtaining the load proportion of all task execution nodes, select the task execution node with the lowest current load based on the load proportion to prioritize task execution; Select the task execution node with the highest current load proportion to temporarily suspend task execution.

The core point of this strategy is to determine the maximum number of times the system can query each data table per second. Essentially, it measures the capacity of a web server under actual load, with the database at its core [8]. After calculating the number of seconds required for querying the data form, taking the reciprocal of the data will determine the number of times the system can query the table per second.

The parameters involved in the implementation process of the strategy described in this paper are shown in Table 1.

**Table 1. The parameters in the formula and their meanings**

| Parameter Name | Meaning |
|---|---|
| DV | The amount of data in this data table |
| Time | System runtime |
| T(n) | The time required to perform a query operation when there are n records in the data table |
| C(s) | The number of records in a data table when the query time is s seconds |
| S(t) | The maximum number of concurrent queries that can be completed on this table within time t |
| QT | The cumulative number of data table queries in this table |
| QPS | The number of queries per second for this table |
| SPQ | The number of seconds required for one query in this table |
| SC | The maximum number of queries that can be processed per second in this table |

The calculation methods for each parameter in the formula are shown in Table 2.

**Table 2. The calculation method of parameters in the formula**

| Parameter Name | Formula |
|---|---|
| QPS | $\mathrm{QT}/\mathrm{Time}$ |
| SPQ | $(1+\mathrm{DV}/\mathrm{C}(2*\mathrm{T}(0)))*\mathrm{T}(0)$ |
| SC | $\mathrm{S}(\mathrm{SPQ})/\mathrm{SPQ}$ |

After completing the parameter calculation, the load proportion of the table can be obtained, as shown in formula (1).

$$Load(table) = \frac{QPS}{SC} \quad (1)$$

The overall load proportion of the task execution node is the sum of the load proportions of each data table, and the calculation formula is shown in (2).

$$\mathrm{NodeLoad} = \sum_{i=1}^{n} Load(table_i) \quad (2)$$

Where n represents the number of data tables in the database.

# 4 Discuss

The disaster recovery strategy and load balancing strategy implemented in this article are relatively basic, and are designed for query intensive centralized distributed WEB systems. In non query intensive systems or web systems without databases, the strategy proposed in this article may not be applicable.

In disaster recovery strategies, more in-depth discussions can be conducted on how to achieve node redundancy more efficiently and with fewer additional nodes; In the load balancing strategy, the strategy described in this paper does not consider peak load and concurrent access situations, so peak load and concurrent access situations considerations should be included in its practical application. At the same time, this paper focuses more on theoretical scheme design and does not verify the accuracy of load balancing strategies. In the future, it is hoped that scholars can verify its accuracy based on this article and improve it according to the verification results.

# 5 Conclusion

This paper proposes a disaster recovery and load balancing strategy for centralized distributed web systems,

which is used to improve early developed web systems with the same architecture.

In this paper, we used two schemes to design disaster recovery strategies. It is recommended to adopt a single point backup strategy when placing greater emphasis on system stability, and a global backup strategy when placing greater emphasis on cost control. In the load balancing strategy, we calculate the load proportion of task execution nodes to effectively determine the load status of each task execution node, and make the load of each task execution node in the distributed system tend to be balanced.

Further research should focus more on the efficiency of disaster recovery strategies and the accuracy of load assessment. In addition, we also need to explore new methods and technologies to improve the scalability and reliability of centralized distributed WEB systems, such as finding a transformation plan from centralized distributed architecture to microservice architecture WEB systems [9].

## Acknowledgments

The authors thank the editor and anonymous reviewers for their helpful comments and valuable suggestions.

## Funding

Not applicable.

## Institutional Review Board Statement

Not applicable.

## Informed Consent Statement

Not applicable.

## Data Availability Statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

## Conflict of Interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## Publisher's Note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

## Author Contributions

Not applicable.

## About the Authors

**SONG, Qingqing**

Faculty of Applied Mathematics and Computer Science; Belarusian State University; 4 Nezavisimosti Avenue, Minsk 220030, Belarus; e-mails: fpm.sunC@bsu.by

**WU, Zhen**

Nanjing University of Aeronautics and Astronautics; No. 29, Yudao Street, Qinhuai District, Nanjing City, Jiangsu Province 210016, China; e-mails: 13wuzhen@sina.com

**XIA, Shaoliang**

Faculty of Applied Mathematics and Computer Science; Belarusian State University; 4 Nezavisimosti Avenue, Minsk 220030, Belarus; e-mails: fpm.sya@bsu.by

## References

[1] Laudon, K. C., & Laudon, J. P. (2017). Essentials of management information systems. Pearson.

[2] Elser, A. (2005). Reliable distributed systems: technologies, web services, and applications. Springer Science & Business Media.

[3] Memon, N., Vighio, M. S., & Hussain, Z. (2019). ‘Web services failures and recovery strategies: A review. Indian J. Sci. Technol, 12(43), 1-6.

[4] Jader, O. H., Zeebaree, S. R., & Zebari, R. R. (2019). A state of art survey for web server performance measurement and load balancing mechanisms. International Journal of Scientific & Technology Research, 8(12), 535-543.

[5] Al Ayubi, S. U., & Nugrahaningsih, N. (2009, December). Centralized-star architecture of web service node as integration solution in complex organization. In Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services (pp. 599-603).

[6] Li, D., Cai, H., Yao, X., & Wang, J. (2005). Exploiting redundancy to construct energy-efficient, high-performance RAIDs. Department of Computer Science and Engineering University of Nebraska-Lincoln.

[7] Ghomi, E. J., Rahmani, A. M., & Qader, N. N. (2017). Load-balancing algorithms in cloud computing: A survey. Journal of Network and Computer Applications, 88, 50-71.

[8] Banga G, Druschel P. Measuring the capacity of a Web server under realistic loads[J]. World Wide Web, 1999, 2(1-2): 69-83.

[9] Salah, T., Zemerly, M. J., Yeun, C. Y., Al-Qutayri, M., & Al-Hammadi, Y. (2016, December). The evolution of distributed systems towards microservices architecture. In 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST) (pp. 318-325). IEEE.