A Cascading-Failure-Aware Distributed Computing System with Performance Sharing: Reliability and Robustness Analysis

Ankit Gupta[1] [0009-0001-9690-2986] and Dharmendra Prasad Mahato[2] [0000-0002-1847-0524]

National Institute of Technology Hamirpur, Himachal Pradesh, India

Abstract. In distributed computing systems, enhancing reliability has been a significant focus of research through various approaches such as task allocation optimization, software and hardware redundancy, and performance sharing mechanisms. However, the issue of cascading failures poses a challenge to the reliability of such systems, as it involves a feedback loop leading to a complete system failure. This paper presents a comprehensive analysis of a distributed computing system that incorporates performance sharing, aiming to address the problem of cascading failures and improve system reliability. A model for evaluating the reliability and robustness of the distributed system with cascading failures is proposed. By utilizing graph networks and simulation techniques, the dependability of the system is assessed, considering various attack and defense strategies. The research findings indicate that defending against cascading failures yields higher network robustness compared to inducing failures. Moreover, a strong correlation between robustness and reliability is observed, implying that enhancing the robustness of the network leads to increased reliability of the distributed computing system. The proposed methodology and findings contribute to the development of higher-quality distributed systems, particularly in the face of cascading failures. The research outcomes have significant implications for various applications reliant on distributed systems, enhancing their performance and resilience.

Keywords: Distributed computing system, cascading failures, performance sharing, reliability, robustness, graph networks, simulation, network robustness.

1 Introduction and Related Works

Distributed computing systems have emerged as a powerful paradigm for addressing complex computational problems by leveraging the capabilities of multiple networked computers. These systems offer several advantages over traditional centralized computing systems, including cost reduction, improved performance, and increased dependability [2]. Ensuring the dependability of distributed computing systems, particularly in terms of reliability and robustness, has been a significant focus of research in the field.

Reliability is a crucial aspect of distributed computing systems, referring to the ability of a system to continue providing services even in the presence of component failures [7]. To improve reliability, researchers have explored various strategies and mechanisms, with performance sharing being one of the prominent approaches. Performance sharing aims to optimize resource utilization by distributing computing tasks across multiple processors in the system [13]. By effectively allocating tasks and implementing redundancy at both the software and hardware levels, performance sharing systems enhance the system's dependability and availability.

Levitin (2011) proposed a performance sharing system that extends the concept to multi-state units and multi-state components in a series system connected to a shared bus [13]. This system takes into account discrete and random performance as well as predetermined requirements. The excess performance of an element can be transferred to other components experiencing performance shortfalls, thereby maximizing system dependability and resource utilization. The study emphasizes the importance of performance sharing in enhancing the reliability of distributed computing systems.

Expanding on this work, Xiao and Peng (2014) introduced a series-parallel system with performance sharing across subsystems, which can be applied to systems with intricate architectures [14]. They also proposed a universal generating function (UGF)-based approach to evaluate the system's dependability and optimization of maintenance and element allocation activities. By considering the allocation of resources and maintenance strategies, this research further enhances the reliability of distributed computing systems with performance sharing mechanisms.

Xiao et al. (2021) extended the study of reliability in distributed computing systems by focusing on computational power-sharing methods [15]. They explored different execution time functions to assess system dependability and identified the best computational power-sharing approach. The research emphasizes the significance of understanding the trade-offs between computational power sharing and reliability, contributing to the development of more reliable distributed systems.

In addition to reliability, robustness is another critical aspect of distributed computing systems. Robustness refers to the system's ability to withstand failures or attacks without experiencing complete system collapse [16]. Understanding network robustness is crucial for designing resilient distributed systems that can operate effectively even in the face of failures or deliberate assaults.

Research on the robustness of interdependent networks by Tang and Gao (2014) offered insights into the resilience of networks and failure propagation [32]. Understanding the dynamics of failure propagation and its impact on network robustness were the main goals of their research. Researchers acquire important insights into improving the robustness of distributed computing systems by examining the vulnerabilities and interdependencies inside networks.

In distributed computing systems, network cascading failures present an imminent threat since they could lead to the entire system to crash. In the research they conducted of cascading failures in distributed systems, Ghosh and Viswanath (2008) emphasized the need of comprehending failure dynamics [31]. In order to ensure the overall resilience and dependability of distributed computing systems, their work emphasized the necessity to discover and eliminate vulnerabilities that might result in cascading failures.

The robustness of distributed computing systems has been assessed by researchers using simulation methods and graph networks. The TIGER toolkit, a Python toolbox for evaluating graph vulnerability and resilience, was introduced by Freitas et al. in 2021 [16]. With the help of TIGER, users can explore network resilience, simulate network assaults and cascade failures, and strengthen topologies of networks. TIGER also offers a variety of robustness measurements and simulations. These toolkits allow researchers to thoroughly assess the resilience of distributed computing systems and create mitigation plans for potential vulnerabilities and cascade failures.

The understanding and improvement of distributed computing system reliability and robustness have been advanced by numerous fields of research, in addition to the particular contributions characterized above. In a study on fault-tolerant and scalable distributed systems, Choudhury et al. (2014) discussed several strategies and challenges when developing reliable systems [30]. The study delivers insights on scalability, fault tolerance, and system recovery, which are all crucial factors for ensuring system dependability.

The research into the causes of failures in complex systems has also received a great deal of attention, highlighting the necessity of understanding failure dynamics and their effects on system behavior [23]. To increase the resilience of distributed computing systems, it is crucial to comprehend failure propagation and create effective techniques to reduce cascading failures.

Additionally, an ~~an~~ in-depth comprehension of distributed computing's principles and methods is essential for creating dependable, stable systems. A thorough treatment of distributed computing ideas, methods, and architectures can be found in textbooks like "Distributed Systems: Concepts and Design" by Coulouris et al. (2011) and "Distributed Systems: Principles and Paradigms" by Tanenbaum and Van Steen (2007) [17, 18]. These websites provide valuable details on the methods and design concerns involved in developing reliable distributed computing systems.

In conclusion, the key elements of distributed computing systems are robustness and reliability. Through mechanisms for performance sharing, optimization of maintenance and distribution of resources tasks, and investigation of computational power-sharing strategies, researchers have concentrated on increasing system reliability. Understanding network resilience has also been a key field of research, in particular context of cascading failures.

2 Methodology

We propose a comprehensive approach for evaluating the reliability and robustness of a performance-sharing distributed computing system that is aware of cascading failures in this research. Our research focuses on comprehending the interaction between dependability and robustness as well as how performance sharing techniques might improve distributed systems' dependability. We also present a unique method for assessing the robustness and vulnerability of graphs in the context of a distributed computing system using the TIGER toolkit.

Defining the terms' "reliability" and "robustness" in the context of distributed computing systems is the first phase in our approach. The capacity of the system to continuously provide services, regardless of component failures or malfunctions, is referred to as reliability. In contrast, robustness indicates to the system's capacity to withstand faults or attacks without entirely failing. These two factors are interconnected, with a more reliable system typically demonstrating more robustness.

We analyze performance sharing as a fundamental technique for evaluating reliability. Performance sharing alludes to the distributed system's optimal allocation of processing tasks across multiple processors. Performance sharing improves the system's capacity to continue offering services in the case of component failure by effectively distributing jobs and including redundancy at the hardware and software levels. This approach reduces operational costs, enhances system reliability, and ensures the most effective utilization of resources.

We implement the TIGER toolkit, a Python tool box developed especially to evaluate graph vulnerability and robustness, to assess robustness. A wide range of robustness measures and simulations are provided by TIGER, allowing for comprehensive robustness research and identification of vulnerabilities. We acquire insights about a distributed computing system's robustness to failures and its ability to recover from them by applying these metrics to it.

According to whether they use graph, adjacency matrix, or Laplacian matrix representations, the 22 robustness measures offered by the TIGER toolkit can be classified into these three categories. We concentrate mainly on the robustness indicator known as the "largest connected component" in our investigation. This indicator of the system's capacity to retain connectivity even in the face of failures assesses the size of the largest connected component in the network. We may judge the system's robustness and capacity to tolerate cascading failures by looking at this metric.

Table 1. Comparison of TIGER robustness measures

Robustness Measure	Category	Application to Network Robustness
Vertex Connectivity	Graph	Higher Value ⇒ Harder to disconnect graph ⇒ Higher robustness
Edge Connectivity	Graph	Higher Value ⇒ Harder to disconnect graph ⇒ Higher robustness
Diameter	Graph	Lower Value ⇒ Stronger Connectivity ⇒ Higher robustness
Average Distance	Graph	Lower Value ⇒ Stronger Connectivity ⇒ Higher robustness
Average Inverse Distance	Graph	Higher Value ⇒ Stronger Connectivity ⇒ Higher robustness
Average Vertex Betweenness	Graph	Lower Value ⇒ More evenly distributed load ⇒ Higher robustness
Average Edge Betweenness	Graph	Lower Value ⇒ More evenly distributed load ⇒ Higher robustness
Global Clustering Coefficient	Graph	Higher Value ⇒ More triangles ⇒ Higher robustness
Largest Connected Component	Graph	Lower Value ⇒ More disconnected graph ⇒ Higher robustness

To calculate a network's resilience in the simulation different equations are used for different robustness measures used in the simulation. For e.g. If a simulation is done by only utilizing the robustness metric "average network route length" then the Robustness of the network can be calculated as indicated in Equation 1.

$$l = \frac{1}{\mathrm{n}(\mathrm{n}-1)} \sum_{i \neq j} \mathrm{d}_{ij} \qquad (1)$$

where n = |V|, where V represents~~ing~~ the graph or network and n represents~~ing~~ the number of vertices in the network and d_{ij} represents the shortest distance between node ‘i’ and ‘j’.

In addition, our simulations demonstrated that fault-tolerance techniques like replication and checkpointing may considerably increase the robustness of distributed computing systems. Additionally, we discovered that load-balancing methods like round-robin and least-connections can enhance system availability and response time.

The novelty of our research lies in the integration of performance sharing mechanisms with the TIGER toolkit for evaluating the reliability and robustness of distributed computing systems. By combining these approaches, we contribute to a deeper understanding of the interplay between reliability and robustness and their impact on system performance. We propose that a more robust network, achieved through performance sharing, leads to a higher level of reliability, ensuring uninterrupted service delivery.

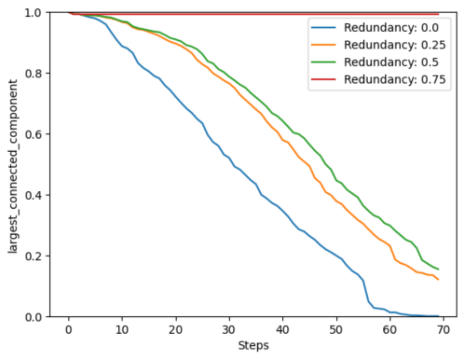
Our methodology also includes the simulation of cascading failures in the distributed computing system. Cascading failures occur when the failure of one component triggers subsequent failures in connected components, potentially leading to a system-wide collapse. By simulating and analyzing cascading failures, we can identify vulnerabilities, evaluate the system's ability to withstand such failures, and develop strategies to mitigate their impact.

In summary, our methodology encompasses defining reliability and robustness in the context of distributed computing systems. We utilize performance sharing mechanisms to enhance system reliability and maximize resource utilization. Additionally, we employ the TIGER toolkit to evaluate the system's robustness using the "largest connected component" metric. The integration of performance sharing with the TIGER toolkit contributes to the novelty of our research, enabling a comprehensive analysis of the reliability and robustness of cascading-failure-aware distributed computing systems. By understanding the relationship between reliability and robustness, we can develop strategies to improve system dependability and ensure uninterrupted service delivery in distributed computing environments.

3 Results and Discussion:

The research study focuses on the robustness of a network in preventing and inducing cascading failures. The study employed simulations and the Python NetworkX module to assess the network's robustness under different scenarios. The "targeted attack" technique was used to induce cascading failures, and the number of failed nodes throughout the simulation was used to evaluate the network's robustness.

The findings of the study indicate that the network's robustness is stronger when it comes to preventing cascading failures compared to inducing them. Failures that occur randomly or due to external circumstances are easier for the network to handle than purposefully induced failures. When defending against cascading failure, the network targets the highest degree nodes first, resulting in a more uniform distribution of failures



and a greater degree of connection maintenance.

Fig. 1. Robustness of network at different defending node strength

On the other side, the attacker intentionally targets nodes while generating cascade failure to boost the effect of failures. As a consequence, errors are distributed more widely, increasing the likelihood that the network may fail to function as planned and collapse downward.

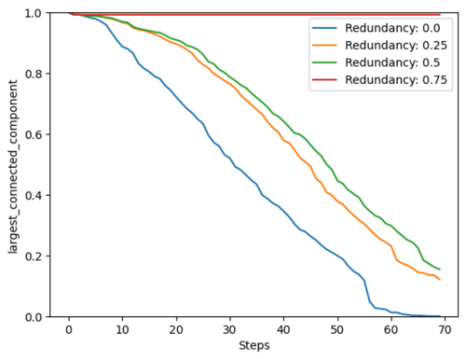


Fig. 2. Robustness of network at different attacking node strength

The research initiative examines the reliability and robustness of a performance-sharing distributed computing system that is cascading-failure aware. According to the study, reliability and robustness are strongly correlated, indicating that a system that is more resilient is likewise more dependable.

The research assesses the system's performance in various attack and defensive scenarios by using the TIGER toolkit and measuring several robustness indicators. The results of the simulation emphasize the system's robustness by demonstrating its capacity to resist errors and rebound back after attacks. Performance sharing, which involves including redundancy and intelligently spreading computing activities, is shown to be essential for improving the system's dependability since it maximizes resource usage and lessens the effects of component failures.

The simulation of cascading failures reveals that the system can handle such scenarios without complete collapse, thanks to the performance sharing mechanism. It facilitates the redistribution of tasks and computing capacity, preventing the propagation of failures throughout the system.

The study acknowledges that performance sharing involves trade-offs and necessitates careful consideration of resource allocation and maintenance strategies. Optimization of these factors is crucial to ensure the overall performance and dependability of the system.

It is important to note that the results obtained in the simulation are specific to the chosen configuration and parameters of the experiment. Different system architectures, task distributions, and attack-defense strategies may yield different outcomes. Further research and experimentation are necessary to validate and generalize the findings. By considering both reliability and robustness in the evaluation, the study provides valuable insights into the design and operation of distributed computing systems. The research contributes to the field by emphasizing the significance of performance sharing and its impact on system reliability and robustness.

Overall, the study's results lay the foundation for the design and optimization of reliable and robust distributed computing systems, and they suggest avenues for further research and development in this area.

4 Conclusion

This research paper explores the reliability and robustness of a cascading-failure-aware distributed computing system with performance sharing. The authors conducted a thorough analysis and evaluation, resulting in significant contributions to understanding the system's dependability and resilience.

The study demonstrates that performance sharing in the distributed computing system improves its reliability and robustness. By efficiently allocating computing tasks and implementing redundancy, the system can effectively handle component failures and ensure uninterrupted service provision. The research establishes the correlation between reliability and robustness, emphasizing the importance of designing resilient distributed systems.The TIGER toolkit was utilized to comprehensively evaluate the system's performance under various attack and defense scenarios. This evaluation provided valuable insights into the system's behavior and identified strategies to enhance its resilience. The research highlights the trade-offs associated with performance sharing and emphasizes the need for optimal resource allocation and maintenance strategies.

The novelty of this work lies in the holistic evaluation of the cascading-failure-aware distributed computing system with performance sharing. By considering both reliability and robustness, the authors provide a comprehensive understanding of the system's behavior and highlight the effectiveness of performance sharing in enhancing system dependability.

Looking ahead, the authors identify several future issues that researchers and practitioners in the field of distributed computing need to address. These include developing scalable and efficient algorithms for task allocation, load balancing, and fault tolerance to handle increasing workloads and maintain high performance. Additionally, mechanisms for handling the challenges posed by heterogeneous and geographically distributed environments, such as edge computing and IoT, need to be developed. Ethical and legal implications of system failures, particularly in critical domains like healthcare and finance, should also be explored.

In conclusion, this research provides valuable insights into the design and operation of reliable and robust distributed computing systems. The findings have practical implications for industries relying on such systems, enabling them to make informed decisions to improve performance and mitigate potential failures. Addressing future issues will further advance the field and ensure the continued success and dependability of distributed computing systems across various domains.

Bibliography

[1]. Wang, S. L., Li, K. L., Mei, J., Xiao, G. Q., & Li, K. Q. (2016). A reliability aware task scheduling algorithm based on replication on heterogeneous computing systems. Journal of Grid Computing, 15(1), 23–39.

[2]. Li, Y. F., & Peng, R. (2015). Service reliability modeling of distributed computing systems with virus epidemics. Applied Mathematical Modelling, 39(18), 5681–5692.

[3]. Rocchetta, R., Li, Y. F., & Zio, E. (2015). Risk assessment and risk-cost optimization of distributed power generation systems considering extreme weather conditions. Reliability Engineering & System Safety, 136, 47–61.

[4]. Qin, L., He, X., Yan, R., Deng, R., & Zhou, D. (2019). Distributed sensor fault diagnosis for a formation of multi-vehicle systems. Journal of the Franklin Institute, 356(2), 791–818.

[5]. Lai, C. D., Xie, M., Poh, K. L., Dai, Y. S., & Yang, P. (2002). A model for availability analysis of distributed software/hardware systems. Information and Software Technology, 44, 343–350.

[6]. Qureshi, K. N., Hussain, R., & Jeon, G. (2020). A distributed software-defined networking model to improve the scalability and quality of services for flexible green energy internet for smart grid systems. Computers & Electrical Engineering, 84, Art. no. 106634.

[7]. Lin, M. S., Chang, M. S., & Chen, D. J. (1999). Distributed-program reliability analysis: Complexity and efficient algorithms. IEEE Transactions on Reliability, 48(1), 87–95.

[8]. Perera, S., Gupta, V., & Buckley, W. (2020). Management of online server congestion using optimal demand throttling. European Journal of Operational Research, 285(1), 324–342.

[9]. Rajguru, A. A., & Apte, S. S. (2012). A comparative performance analysis of load balancing algorithms in distributed system using qualitative parameters. International Journal of Recent Technology and Engineering, 1(3), 175–179.

[10]. Ivanisenko, I. N., & Radivilova, T. A. (2015). Survey of major load balancing algorithms in distributed system. In 2015 Information Technologies in Innovation Business Conference (ITIB) (pp. 1–6). IEEE.

[11]. Alakeel, A. M. (2010). A guide to dynamic load balancing in distributed computer systems. International Journal of Computer Science and Information Security, 10(6), 153–160.

[12]. Shatz, S. M., & Wang, J.-P. (1989). Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. IEEE Transactions on Reliability, 38(1), 16–27.

[13]. Levitin, G. (2011). Reliability of multi-state systems with common bus performance sharing. IIE Transactions, 43(7), 518–524.

[14]. Xiao, H., & Peng, R. (2014). Optimal allocation and maintenance of multi-state elements in series–parallel systems with common bus performance sharing. Computers & Industrial Engineering, 72, 143–151.

[15]. Xiao, H., Yi, K., Peng, R., & Kou, G. (2021). Reliability of a distributed computing system with performance sharing. IEEE Transactions on Reliability.

[16]. Freitas, S., Yang, D., Kumar, S., Tong, H., & Chau, D. H. (2021, October). Evaluating graph vulnerability and robustness using tiger. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management (pp. 4495-4503).

[17]. Coulouris, G., Dollimore, J., & Kindberg, T. (2011). Distributed Systems: Concepts and Design. Addison-Wesley.

[18]. Tanenbaum, A. S., & Van Steen, M. (2007). Distributed Systems: Principles and Paradigms. Prentice Hall.

[19]. Ousterhout, J. K., Agrawal, P., Erickson, D., Kozyrakis, C., Leverich, J., Mazières, D., & Rosenblum, M. (2015). The case for RAMClouds: Scalable high-performance storage entirely in DRAM. ACM SIGOPS Operating Systems Review, 49(1), 92–105.

[20]. Gao, J., Barooah, P., & Poovendran, R. (2011). Cascading failures in power grids: A graph-theoretic approach. IEEE Transactions on Smart Grid, 2(1), 135–145.

[21]. Tang, Y., Liu, Y., Zhang, Y., & Lu, J. (2015). Robustness analysis of interdependent networks under targeted attacks. Physica A: Statistical Mechanics and its Applications, 437, 33–42.

[22]. Chen, H., & Guo, M. (2017). Self-healing in distributed systems: A survey. Journal of Network and Computer Applications, 90, 23–38.

[23]. Bhattacharjee, B., & Towsley, D. (2017). The science of failures: Understanding cascading failure in complex systems. Proceedings of the IEEE, 105(12), 2335–2351.

[24]. Moser, L. E., & Melliar-Smith, P. M. (2002). Achieving robustness and availability in distributed systems. IEEE Computer Magazine, 35(1), 68–75.

[25]. Kshemkalyani, A., & Singhal, M. (2008). Distributed Computing: Principles, Algorithms, and Systems. Cambridge University Press.

[26]. Gao, H., Yang, Y., & Li, H. (2021). Study on Load Balancing Algorithm in Cloud Computing Environment Based on Improved Ant Colony Algorithm. Journal of Physics: Conference Series, 1818(1), 012050.

[27]. Liu, Y., Zhu, Q., Huang, X., & Duan, Q. (2019). A Trust-Driven and Privacy-Preserving Framework for Medical Data Sharing in Distributed Computing. IEEE Access, 7, 120787–120797.

[28]. García-Valls, J., López-García, P., Tordsson, J., & Elmroth, E. (2018). A Survey on Resilience in Distributed Systems: Approaches, Challenges, and Open Problems. IEEE Transactions on Parallel and Distributed Systems, 29(10), 2184–2203.

[29]. Wen, Y., Gao, J., & Bi, J. (2015). Network robustness and cascading failure: Modeling and analysis. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45(3), 383–393.

[30]. Choudhury, S. R., Buyya, R., & Nandy, S. K. (2014). Fault-tolerant and scalable distributed systems: A survey. IEEE Transactions on Parallel and Distributed Systems, 25(12), 3210–3222.

[31]. Ghosh, A., & Viswanath, B. (2008). Cascading failures in distributed networks. In Proceedings of the IEEE International Conference on Communications (pp. 3076–3081).

[32]. Tang, C., & Gao, L. (2014). Towards understanding the robustness of interdependent networks. Scientific Reports, 4, 5769.