Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

A Case Study of Implementation Strategy for Performance Optimization in Distributed Cluster System

Taufik Rendi Anggara
Senior Lecturer, Faculty of Computer Science, Universitas Esa Unggul, Jakarta, Indonesia
Corresponding Author: Taufik Rendi Anggara, **E-mail**: taufik.anggara@esaunggul.ac.id

| ABSTRACT
Nowadays, many people spend their time on the Internet, and the number of people subscribed to mobile phones is 69.4% of the 5.61 billion population in the world. To handle this situation, we need to implement a high-performance Distributed Cluster System (DCS) in the correct architecture as well. We separated the cluster for each purpose and gave it a unique VLAN. This study uses a mix of methodologies between case study and system development with evaluation after implementation. We observe all aspects of built-in technologies. In this research, monolith spikes us for performance issues, and also, the infrastructure is messy implemented. Event Based System (EBS) helps DCS to absorb high processing tasks in peak situations. EBS can easily lose a couple as needed. Labeling the incoming data assists us in managing inconsistent distributed data in the environment. Our research was evaluated for two weeks. The result is very pleasant, and the requirements in this research were satisfied.

| KEYWORDS

Cluster System, Distributed System, High Performance Systems, Implementing Strategy, Load Balancing, High Availability.

| ARTICLE INFORMATION

ACCEPTED: 01 March 2024 **PUBLISHED:** 14 March 2024 **DOI:** 10.32996/jcsts.2024.6.1.27

1. Introduction

Nowadays, hours spent on the internet are growing significantly. The data said the average hours spent using the internet through mobile phones were 4 hours (data.ai, 2023). Furthermore, the number of people subscribed to mobile phones was 69.4% of the 5.61 billion population (DataReportal, 2023). From this activity, we can see that the application/website is being accessed a lot by mobile phones and is generating significant revenue. For example, Facebook's largest revenue comes from mobile phone advertisements (Statista, 2019). Therefore, to ensure the stability and continuity of the application, it is necessary to have the right architecture and calculations to create distributing processes, queue systems, and process resolution within it. Implementing a clustered and distributed system alone is not enough; deep knowledge of how the system works is required. Lack of knowledge and experience in maintenance, management, and improper configuration can lead to serious and costly mistakes, damaging both the company's financial aspect and reputation (Clinch, 2015), (Guardian, 2015). Therefore, implementing a distributed system poses various challenges, such as difficulty in finding suitable experts and high implementation costs.

Dispersed distributed cluster frameworks speak to a crucial column in computer science, advertising capable arrangements to address the ever-growing requests for versatility, unwavering quality, and execution in cutting edge computing situations. Be that as it may, in spite of their various benefits, these frameworks are not without challenges and issues. One critical concern rotates around the complexity inalienable in planning, actualizing, and keeping up conveyed clusters. As these frameworks regularly include various interconnected hubs or servers, guaranteeing consistent communication, information consistency, and blame resilience over the cluster presents impressive challenges. Furthermore, the heterogeneity of equipment and program components inside conveyed clusters can present compatibility issues and assist in complicating framework administration and optimization endeavors. In addition, accomplishing ideal stack adjusting and asset assignment over conveyed clusters remains a progressing

challenge, requiring modern calculations and techniques to successfully convey workloads and maximize framework efficiency (Lou, 2021).

Another basic issue in dispersed cluster frameworks relates to security and information keenness. With information being disseminated over numerous hubs or servers inside the cluster, guaranteeing secrecy, astuteness, and accessibility becomes vital. The decentralized nature of disseminated clusters presents vulnerabilities to unauthorized access, information breaches, and malevolent assaults. Actualizing strong security components, encryption procedures, and getting controls is fundamental to relieving these dangers and protecting delicate information. Additionally, keeping up information consistency and coherence over disseminated clusters poses critical challenges, especially in scenarios including concurrent get to and upgrades to shared information assets. Procedures such as disseminated exchanges, replication, and synchronization components are frequently utilized to address these challenges and guarantee information keenness and consistency over the disseminated cluster environment. In general, whereas dispersed cluster systems offer immense potential for scalable and reliable computing, addressing these issues is crucial to unlocking their full benefits and ensuring their successful deployment in real-world applications.

2. Literature

2.1 Distributed System Cluster

Distributed System Cluster (DSC) is the grouping or merging of uniform functions from nodes/servers/VMs, aiming to enable applications to process data simultaneously with large and fast capacities (Talmale, 2021), (Mohd, 2001), (Cheng, 2018). The large capacity is achieved through the distribution of tasks within the cluster system. Another advantage of this distributed cluster system is the assurance of availability and ease of maintenance. However, the implementation of DSC also comes with a variety of complexities, depending on the number, diversity of functions, and the type of infrastructure services owned. In some industries that have implemented DSC, various types of infrastructure are used, including cloud, on-premises, and hybrid (mixed infrastructure).

2.2 High Availability and Load Balancing

Load Balancing (LB) is a configuration technique that utilizes two or more nodes/servers/VMs to distribute processing loads, making the created processes faster and capable of completing many tasks within a specific timeframe. On the other hand, High Availability (HA) is used for mutual backup between interconnected nodes/servers/VMs (Shahid, 2020), (Eibl, 2019), (Afzal, 2019). This configuration technique is commonly used for applications requiring high performance, substantial processing capacity, and guaranteed availability, achieved when both LB and HA are combined in the configuration. The combination forms a cluster, maximizing usage compared to utilizing only one of them. This technique can be applied to various types of functions on nodes/servers/VMs, such as databases, web servers, queuing systems, cookie/cache systems, and more.

2.3 Monolith and Event Based System

Monolithic architecture is a traditional way of designing and implementing a system. The monolithic system is considered to have many drawbacks, such as the difficulty of making changes to the software features that need modification, challenges in scaling up, and system processes slowing down as the number of users accessing the system increases, among other issues (Auer, 2021), (Gos, 2020), (Elgheriani, 2022). The difficulty in development becomes a bottleneck when the system is used by an increasing number of customers. Therefore, architectural changes are needed before the trend of increasing customer numbers experiences a significant rise. This change requires thorough planning and an experienced team to carry out the architectural transformation.

To achieve optimal performance in a production system, it is essential to have a system that facilitates development, vertical and horizontal scaling, and maintenance without prolonged downtime. This means the system can be easily assembled and disassembled as needed. The design of this system leverages Event-Based System (EBS) technology. EBS utilizes events to process data across various services. In other words, when a data processing request occurs, this service will work and send the data to the intended service for processing and storing it in the database (Eisenhauer, 2009), (Tragatschnig, 2018), (Lazzari, 2023). Additionally, EBS can assist in managing queues in data processing asynchronously.

3. Methodologies

This research employs a system development approach combined with a literature review and case studies (Yearworth, 2013), (Hassani, 2017), (Mbanaso, 2023), (Nallaperumal, 2013), (Van der Mewer, 2019). We obtained case studies from a big application which served customers approx. 1K – 5K traffic an hour. A literature review is used to provide insights, draw conclusions from various scientific references, and address problems identified through observation. The research method can be depicted in the figure 1 below.

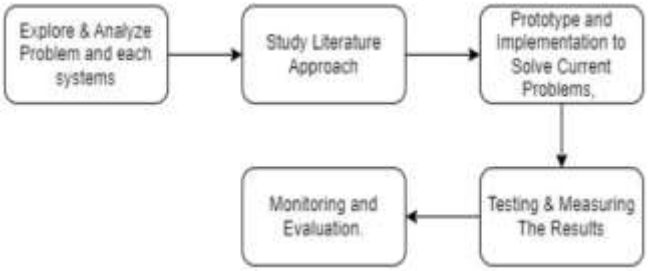

Fig 1. Research Method

The observation stage was conducted with several different applications that we used for this research, and the conclusion is as follows:

- Lack of Performance in Infrastructure and Data Processing
- Lack of Data Consistency, integrity, and prevention data loss (System Disruption)
- Difficulties of workloads, scaling, recovery, and maintenance of the systems.
- Difficulties reach zero downtime services.
- Difficulties of Security Configuration.

4. Analysis, Results, Evaluation

4.1 Analysis

To address the above question, a thorough analysis of the research object is required. This analysis starts from the infrastructure scheme to the process flow of the running application and the configuration scheme of the database. The goal is to understand the root cause of the issues occurring in the research object. This analysis is conducted through focus group discussions with literature studies, where the results yield two architectures: the infrastructure architecture and application flow process architecture, which can be depicted in Figures 2 and 3.

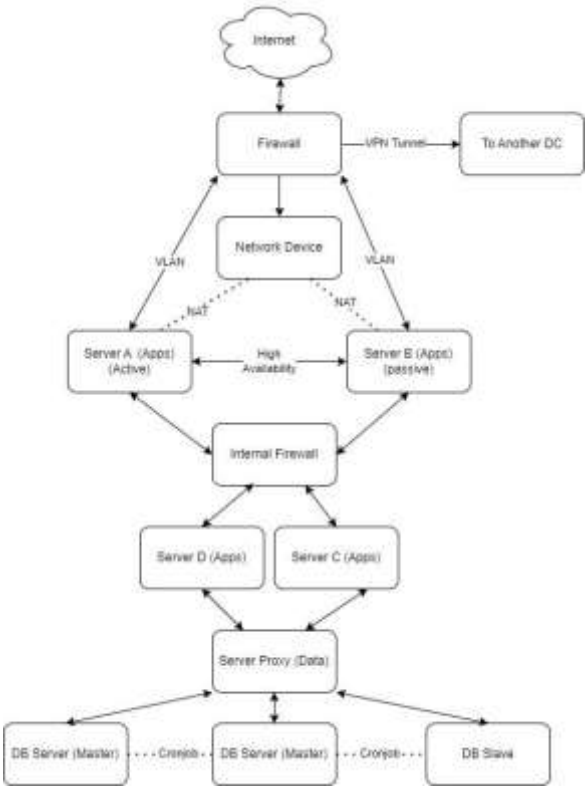

Fig 2. Messy Infrastructure Architecture Scheme.

Figure 2 represents the observation results of the existing issues. This figure explains that there is no automatic High Availability function yet, no load balancing function, and no grouping of device functions. This is indicated by Server A, which actively functions as the workload processor, while Server B functions passively. In other words, if a failure occurs on Server A, Server B will replace it manually. Of course, this would be inconvenient in the event of device failures or flooding traffic incidents and could lead to data redundancy when processing data experiences delays. After obtaining observation results from the infrastructure, we proceeded to observe the process flow, which can be depicted below.

Similarly, Server C and Server D function as API Services, serving as business logic connected to the database. The connection between servers C and D to the database systems is still managed through a proxy server (data). The purpose of this setup is to distribute the task of storing data on the Database Server located at the end of the diagram. Unfortunately, automatic replication between Database Server 1 (Master) and Database Server 2 (Master) and the Database Slave has not been found in the database server configuration. Replication configuration between database servers is still done through cronjob/crontab services. Such configuration can lead to data inconsistency within the database. Additionally, the use of 2 database servers (Master) and 1 database slave may not be appropriate because application users are more likely to view the content of their applications rather than inputting data.

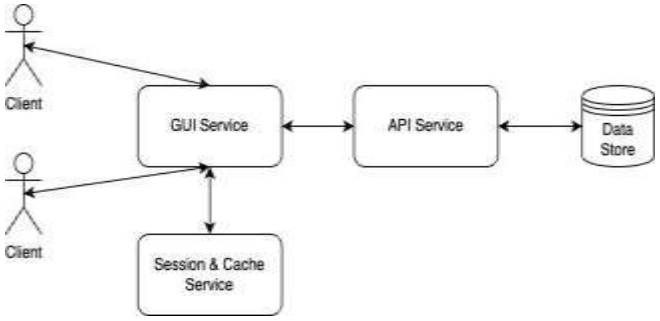

Fig 3. Monolith Application Flow Process.

Further observations were conducted on the process flow, which can be depicted in Figure 3. In this observation, a simple process within the application flow was found, known as the Monolith Scheme. The monolith scheme explains the use of simple processes that may potentially lead to process overload due to still using a single process and not combined with an event broker system. Another potential issue with using the monolith system is the loss or corruption of data due to unfinished data processing when an overload occurs. This can lead to difficulties in filtering and identifying such data.

4.2 Results

After conducting observations to analyze the issues, we proceeded with a literature review as outlined in Point 2 above. From the literature review, conclusions were drawn, which can be depicted in Figure 4 below, regarding the improvement of architecture for cluster schemes and distribution systems. In addition to the improved infrastructure architecture, we also made improvements to the cluster system, as shown in Figure 5 and Figure 6. As for the improvement of process flow, it is illustrated in Figure 7 below. To facilitate data classification, the addition of identifiers indicating the origin of the data processed was implemented, as shown in Table 1.

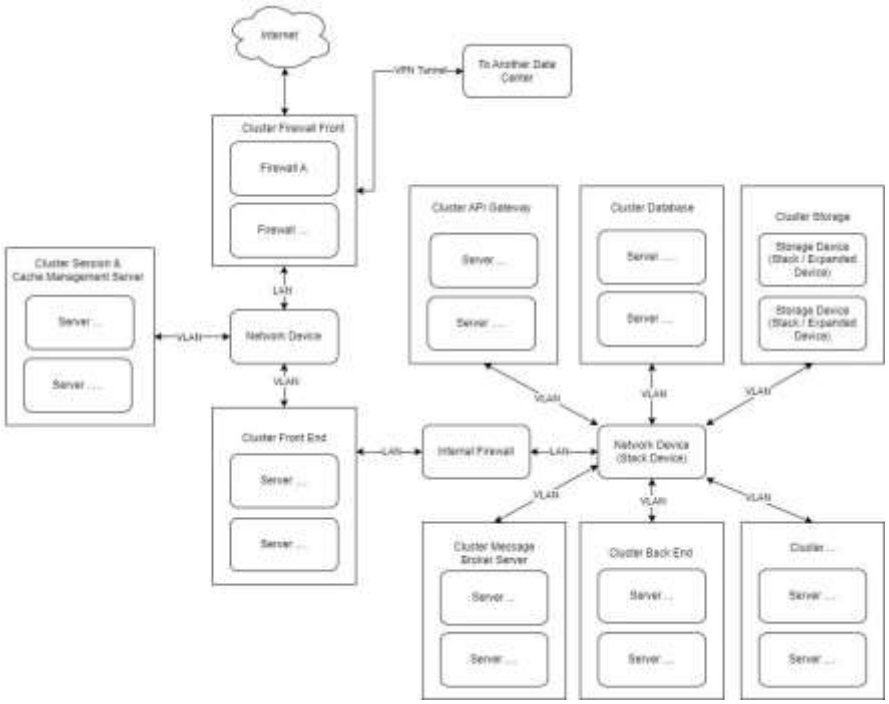

Fig 4. Secure Cluster & Distributed Architecture Scheme

Figure 4 explains that in order to address the issues in this research, it is necessary to improve the architecture of the case studies obtained from Figures 2 and 3. The architecture improvement is done by grouping and remapping according to the function of each machine. The aim is to enable infrastructure managers to easily manage, maintain, and respond quickly to downtimes, distribute tasks in processing, perform mass configuration, and scale horizontally if there is a need in the future. Once grouped, new communication pathways (VLANs) can be established for each group to ensure the security of each device group.

In Figure 4, we also designed the architecture by emphasizing security aspects. Security in this architecture can start from the Segregation of the Network, the use of API Gateway Management, the clustering of server types (cluster), which are configured as depicted in Figure 5, and the use of 2 layers of firewall (internal and front firewall). We also utilize pre-built hardening OS (CIS, 2024) to enhance security at the operating system level, and configuring OS security will be easier and faster. Additionally, this architecture can address the questions raised during the observation, specifically the difficulties of security configuration.

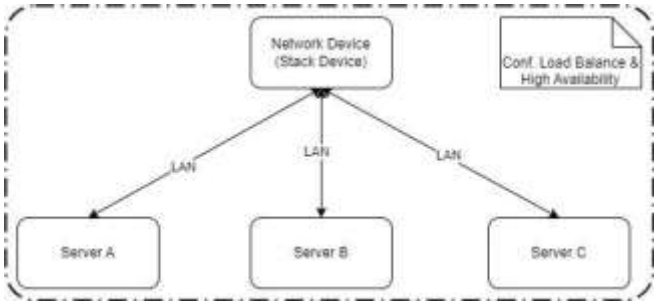

Fig 5. Inside Sever Group (cluster) in Architecture Scheme

Figure 5 illustrates the necessity of configuring Load Balancing combined with high availability. The purpose of using this configuration is to ensure the reliability and continuity of the system and applications. Additionally, the implemented load balancing will be capable of distributing processes according to the capacity of each node. This will align with addressing the problem of lack of performance.

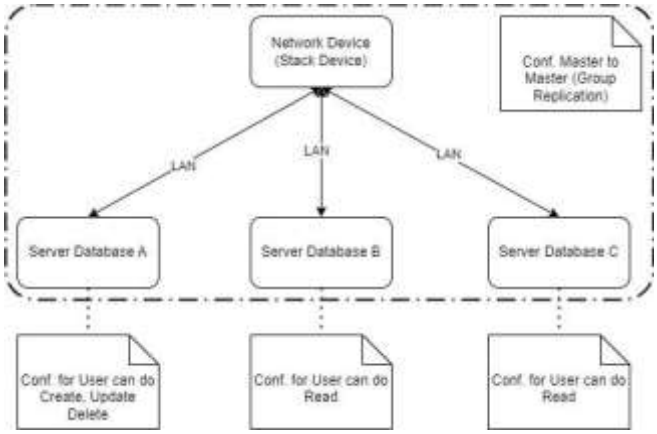

Fig 6. Inside Database Group (cluster) in Architecture Scheme

Figure 6. depicts the separation and delegation of tasks in data processing within the database. This separation and delegation are carried out to enhance performance in data processing. The scheme formed in this research clusters the database using a 1:2 ratio, where the ratio for processing data to be displayed using queries (Views Data) is much larger than the data processed for purposes of alteration, deletion, and creation of new data. The objective of creating this ratio is to provide capability and performance improvement in the process of viewing data because viewing data will be processed far more frequently than processes for altering, deleting, and creating new data.

Table 1. Data Labeling in Incoming data and process.

ID	Name	Incoming data From
acc_xj2.112	John	192.168.10.13
acc_xj2.113	Mayer	192.168.10.12

In Table 1, we provide labels for each newly created data and data that has been successfully updated. This data labeling is done by appending the IP Host where the data was successfully processed. The purpose of labeling this data is to avoid inconsistent data (Data Redundancy) resulting from system failures so that the team can easily identify this data if a system failure occurs. Data labeling is performed for each data that is processed for alteration as well as for new data.

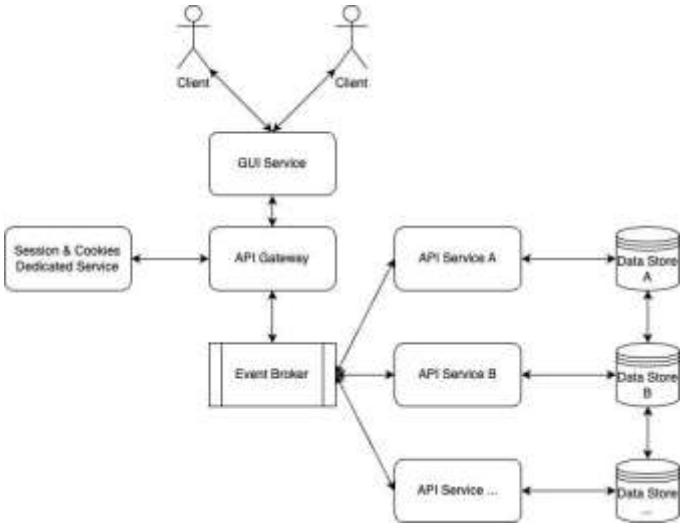

Fig 7. Event Based System in Application Flow Process Scheme.

Figure 7 represents the changes that have been made to improve the previous system. These changes are supported by the use of an event-based system. The purpose of using this system is to easily manage, distribute, and distribute processes (workloads at a specific point), as well as replicate processed data to data storage locations. Additionally, an API Gateway has been added to

protect endpoints from direct access to the node where the endpoint originates. Furthermore, we have also divided and grouped endpoints for API services with similar functions and processes to facilitate the management of these endpoints.

4.4 Evaluation

After two weeks of evaluation, which involved monitoring performance, workload distribution, and data processing, the following results were obtained:

- No slow processes were found due to an overloaded system.
- No decrease in application performance was observed.
- No downtime was experienced during monitoring.
- No inconsistent data was found due to production system failures.
- The management team could easily manage both the infrastructure and applications.

5. Conclusion

This research undertook a comprehensive investigation into the implementation strategy for performance optimization in a Distributed Cluster System (DCS) within the context of increasing global internet usage and the growing reliance on mobile phones. The study revealed challenges within the existing monolithic architecture, including performance issues and messy infrastructure implementation. The adoption of an Event-Based System (EBS) emerged as a pivotal solution to absorb high processing tasks during peak periods, manage data inconsistencies, and enhance the overall system performance. Additionally, the research addressed challenges related to load balancing, high availability, and security configurations. Through a two-week evaluation, the implemented improvements demonstrated significant success, eliminating slow processes, maintaining high application performance, ensuring zero downtime, and preventing data inconsistency. The findings highlight the importance of strategic architectural adjustments and the integration of advanced systems to meet the demands of a rapidly evolving technological landscape.

In summary, the research not only identified crucial issues in the existing system but also proposed a refined architecture featuring secure clusters and distributed schemes. The improvements encompassed load balancing, high availability configurations, and the incorporation of an Event-Based System to streamline processes. The evaluation phase validated the effectiveness of these enhancements, showcasing their capability to overcome previous challenges and provide a stable, high-performance environment. The research underscores the significance of well-planned architectural changes and the utilization of modern technologies to optimize the performance of distributed cluster systems in the face of increasing demands on internet-based applications and services.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
ORCID ID: https://orcid.org/0009-0002-1510-3961
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.

References

[1] Afzal, S., & Kavitha, G. (2019). Load balancing in cloud computing a hierarchical taxonomical classification. *Journal of Cloud Computing, 8*(1). https://doi.org/10.1186/s13677-019-0146-7
[2] Auer, F., Lenarduzzi, V., Felderer, M., & Taibi, D. (2021). From monolithic systems to microservices: An assessment framework. *Information and Software Technology*, 137, 106600. https://doi.org/10.1016/j.infsof.2021.106600
[3] Cheng, Y., Chai, Z., & Ali, A. (2018). Characterizing co-located datacenter workloads: An alibaba case study. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.1808.02919
[4] CIS. (2024). CIS benchmarksTM. Https://Www.cisecurity.org/Cis-Benchmarks. https://www.cisecurity.org/cis-benchmarks
[5] Clinch, M. (2015). Bloomberg hit with outage. In CNBC. https://www.cnbc.com/2015/04/17/bloomberg-trading-terminals-experience-outage.html
[6] Data.ai. (2023). State of mobile 2023 - data.ai. Data.ai. https://www.data.ai/en/go/state-of-mobile-2023/
[7] DataReportal. (2023). Digital around the World. DataReportal Global Digital Insights. https://datareportal.com/global-digital-overview
[8] Eibl, S., & Rüde, U. (2019). A systematic comparison of runtime load balancing algorithms for massively parallel rigid particle dynamics. Computer Physics Communications, 244, 76–85. https://doi.org/10.1016/j.cpc.2019.06.020
[9] Eisenhauer, G., Wolf, M., Abbasi, H., & Schwan, K. (2009). Event-based systems. Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, DEBS '09(2), 1–10. https://doi.org/10.1145/1619258.1619261
[10] Elgheriani, N., Ali, N., & Ahmed, S. (2022). Microservices vs. monolithic architecture, the differentials structure between two architecture. *Minar International Journal of Applied Sciences and Technology, 3*(6). https://doi.org/10.47832/2717-8234.12.47

[11] Gos, K., & Zabierowski, W. (2020, April). The comparison of microservice and monolithic architecture. 2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH). https://doi.org/10.1109/memstech49584.2020.9109514
[12] Guardian. (2015, June 17). RBS could take until weekend to make 600,000 missing payments after glitch. The Guardian. https://www.theguardian.com/business/2015/jun/17/rbs-fails-to-make-600000-payments-customers-it-technology-failure-bank
[13] Hassani, H. (2017). Research methods in computer science: The challenges and issues. ArXiv:1703.04080 [Cs]. https://arxiv.org/abs/1703.04080
[14] Lazzari, L., & Farias, K. (2023). Uncovering the hidden potential of event-driven architecture: A research agenda. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.2308.05270
[15] Luo, P., Huang, Q., & Tung, A. K. H. (2021, June 19). *A generic distributed clustering framework for massive data*. ArXiv.org. https://doi.org/10.48550/arXiv.2106.10515
[16] Mbanaso, U. M., Abrahams, L., & Okafor, K. C. (2023). Research techniques for computer science, information systems and cybersecurity. SpringerNature. https://doi.org/10.1007/978-3-031-30031-8
[17] Mohd Y S, & Evans, D. J. (2001). Distributed computing on cluster systems. *International Journal of Computer Mathematics, 78*(3), 383–397. https://doi.org/10.1080/00207160108805118
[18] Nallaperumal, K., & Krishnan, A. (2013). Engineering research methodology a computer science and engineering and information and communication technologies perspective. Publisher: PHI Learning Private Limited,. (Original work published 2014)
[19] Shahid, M. A., Islam, N., Alam, M. M., Su'ud, M. M., & Musa, S. (2020). A comprehensive study of load balancing approaches in the cloud computing environment and a novel fault tolerance approach. IEEE Access, 8, 130500–130526. https://doi.org/10.1109/access.2020.3009184
[20] Statista. (2019). Facebook: Mobile ad revenue share per quarter 2019. Statista. https://www.statista.com/statistics/999580/share-of-mobile-facebook-ad-revenue-quarter/#:~:text=As%20of%20the%20third%20quarter
[21] Talmale, G., & Shrawankar, U. (2021). Cluster based real time scheduling for distributed system. Revistas.usal.es, 10(2). https://revistas.usal.es/cinco/index.php/2255-2863/article/download/ADCAIJ2021102137156/26181?inline=1
[22] Tragatschnig, S., Stevanetic, S., & Zdun, U. (2018). Supporting the evolution of event-driven service-oriented architectures using change patterns. Information and Software Technology, 100, 133–146. https://doi.org/10.1016/j.infsof.2018.04.005
[23] Van der M, A., Gerber, A., & Smuts, H. (2019). Guidelines for conducting design science research in information systems. Communications in Computer and Information Science, 1136(1136), 163–178. https://doi.org/10.1007/978-3-030-35629-3_11
[24] Yearworth, M., Edwards, G., Davis, J., Burger, K., & Terry, A. (2013). Integrating problem solving and research methods teaching for systems practice in engineering. *Procedia Computer Science, 16*(16), 1072–1081. https://doi.org/10.1016/j.procs.2013.01.113