OPEN ACCESS
ISSN 2356-5462
http://socj.telkomuniversity.ac.id/ijoict/

Intl. Journal on ICT
Vol. 9, No. 2, Dec 2023. pp. 41-52
doi: doi.org/10.21108/ijoict.v9i2.761

Development of High Availability Database Infrastructure for OSS Projects with Monitoring Systems in Cloud Computing Environments

Akmal Ikhsan Ollong[1], Dana Sulistyo Kusumo[2*]

[1,2]*School of Computing, Telkom University*
Telecommunication street no. 1, Terusan Buahbatu, Bandung, West Java, Indonesia 40257
danakusumo@telkomuniversity.ac.id

Abstract

This study addressed a critical problem in the Kominfo's Operation Support System (OSS) project, which significantly impacted business operations. The outage of the MongoDB database, a vital component of OSS, resulted in server crashes and data damage. To overcome this issue, the OSS team initiated a project to migrate to PostgreSQL for high availability and improved database performance. This choice was based on the fact that PostgreSQL outperformed MongoDB in join collection operations in terms of performance and this had an impact on the project's requirements, the implementation involved the use of HA PostgreSQL technology, with multiple connected servers sharing data in real-time. Through functional and performance testing, the study has demonstrated that the HA PostgreSQL system increased database availability, managed server failures, and facilitated effective cluster administration. The findings of this research can guide the development of the OSS project's IT infrastructure and serve as a reference for similar projects utilizing HA PostgreSQL technology.

Keywords: High Availability, PostgreSQL Cluster, Data damage, Operation Support System

I. INTRODUCTION

The occurrence of a serious issue certainly had a significant impact on the company, including one that rred in the Operation Support System (OSS) project [6]. OSS was one of the Kominfo projects aimed at strengthening the resilience and preparedness of the community in facing disasters, such as earthquakes, as well as strengthening digital infrastructure throughout Indonesia. Previously, the OSS team encountered a problem where the MongoDB database in OSS experienced a serious issue caused by a sudden datacenter outage. This resulted in several OSS servers going down, so when the servers came back up, some data was affected and became corrupt, including WT (Wired Tiger), which was the engine of MongoDB. As a result, the OSS MongoDB database frequently went down, even almost every hour, requiring constant recovery. This problem posed a challenge for the OSS team in finding a solution.

Received on 09 Jul 2023, Revised on 21 Jul 2023, Accepted and Published on 12 Dec 2023.

With an awareness of the importance of data security, availability, and integrity for business continuity[6], the OSS team undertook a refactoring project using the PostgreSQL database. Based on analysis by one of the data engineer colleagues, the choice was based on the fact that PostgreSQL outperformed MongoDB in join collection operations in terms of performance and this had an impact on the project's requirements[11]. PostgreSQL enabled the OSS team to improve database performance and reduce downtime due to datacenter failures. By adopting High Availability technology, it was expected that the availability of the database could be well-maintained, and downtime due to maintenance could be minimized[5].

However, implementing High Availability PostgreSQL was not simple and required a deep understanding of the architecture and related concepts. Additionally, there were many technologies and methods that could be used to implement high availability PostgreSQL [3]. High Availability (HA) was not always identical to a database cluster, but HA could be achieved by using a database cluster. By employing HA, a system's ability to remain available and operate effectively over a long period, with minimal downtime and data loss, could be achieved [4].

We endeavored to provide several coordinated database servers working together to deliver better database services and improve system availability. What is meant by HA (high availability)? In this context, what we were trying to build was to ensure that our database remained online for as long as possible. An important component here was the hardware that housed the database itself. No matter how perfect a machine and its contents were, failures or unexpected behaviors from any element could result in downtime [3].

High availability PostgreSQL was crucial for the OSS project because the PostgreSQL database was used as the main database for continuously operating applications. In a business environment, database downtime could cause significant financial and reputational losses [2]. Therefore, HA PostgreSQL had to be ensured to minimize downtime and maximize system availability [4]. With high availability solutions such as data replication and automatic failover, the OSS project could deliver reliable and uninterrupted services to its application users.

II. RELATED STUDIES

A. *PostgreSQL (PG) Cluster*

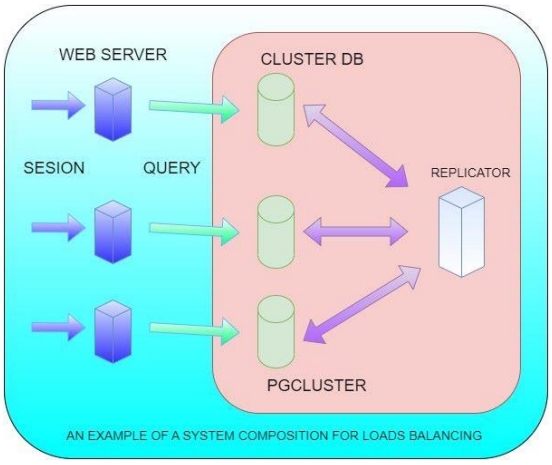

Fig 1. PGCluster Components for System Load Balancing [1]

PG Cluster or PostgreSQL Cluster is a collection of several PostgreSQL instances that interact and work together to provide a more reliable and scalable database service. The Figure 1 showed that PG Cluster is typically used in high availability environments, where multiple PostgreSQL instances are connected and

share data to achieve high redundancy and reliability. PG Cluster allows PostgreSQL servers to be accessed through a single endpoint, thus improving service scalability, performance, and availability. One of the functions of PG Cluster is Load Sharing, where the request load on the server can be distributed[1].

B. High Availability

High Availability is the ability of a system to continue operating even if there are failures in its components. High Availability can be achieved in various ways such as using redundancy on important components, or using clustering technology to ensure that the system is always available[1].

2.3. Monitoring and Managing Cluster

To ensure and stabilize High Availability on PostgreSQL to run well, there are several technologies that can be used, including:

a. Ansible

In this case, Ansible is used to automate IT tasks, including managing cluster deployment and configuration. Ansible allows system administrators to compose and execute commands on a set of servers at one time, thus speeding up the deployment and configuration process. In cluster deployment, Ansible can be used to speed up the installation and configuration process on each node in the cluster, as well as ensuring uniformity of configuration on each node. Ansible also provides features for monitoring and automating tasks periodically, thus facilitating the maintenance process on the cluster.

b. Zabbix

The usefulness of Zabbix here is to monitor networks, servers, IT applications, and services in real-time. Zabbix has the ability to monitor the availability, performance, and utility of various devices, including clusters and servers.

In the context of the OSS project, Zabbix can be used to monitor the availability, performance, and resource usage of clusters and servers. Some examples of using Zabbix in OSS projects include:

1. Monitoring server availability: Zabbix can monitor servers in real-time and provide notifications if there is downtime or problems with the server.

2. Monitoring server performance: Zabbix can monitor server performance, including CPU usage, RAM, and disk space. This can help in taking proactive action to optimize server performance.

3. Monitoring cluster availability: Zabbix can also monitor cluster availability and provide notifications if there is a problem with one of the nodes in the cluster.

4. Monitoring cluster performance: Zabbix can monitor cluster performance, including CPU usage, RAM, and disk space on each node. This can help in taking proactive action to optimize cluster performance.

By using Zabbix, the OSS team can monitor clusters and servers effectively and efficiently. This can help in maintaining the availability and performance of services provided by the OSS project.

c. Slack

Slack is a communication and collaboration platform that allows users to communicate in real-time via text and audio/video messages. Slack also has notification features, where users can receive certain notifications through their application or device.

To get notifications from Zabbix through Slack, users can integrate the two platforms using third-party plugins or applications such as "Zabbix Slack Alert". With this integration, Zabbix can send notifications to a specific Slack channel when there is a problem or anomaly on the server being monitored, so that users can take immediate action. Users can also configure to customize the types of notifications they want to receive or to limit the number of notifications received within a certain period of time.

C. Implementation of High Availability on PostgreSQL in the OSS environment

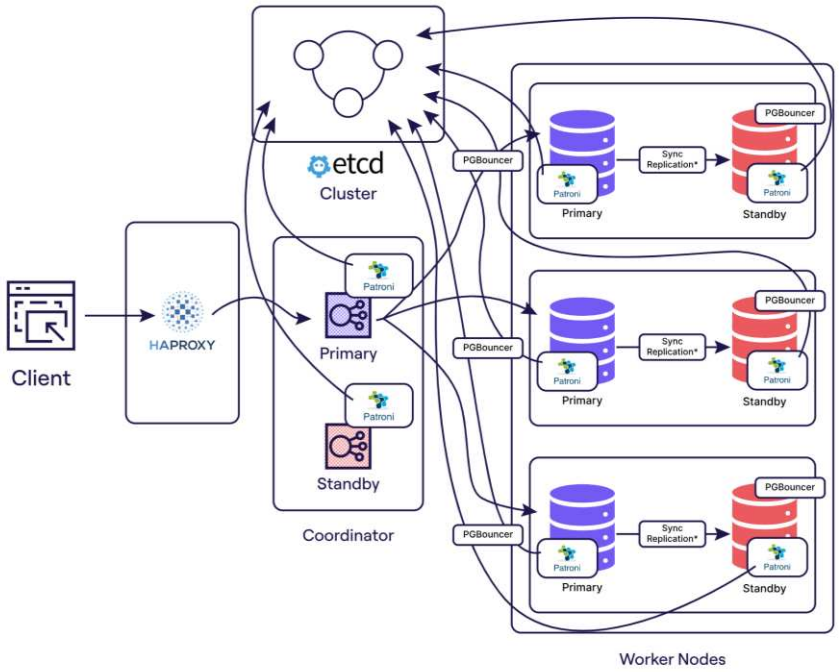

Fig 2. PostgreSQL High-Availability Design with Load Balancing [14]

The Figure 2 was a Complete Sample Configuration for high availability and ensured High Availability on PostgreSQL. There were several technologies that could be used, including:

a. Replication: Replication technology allows data on one PostgreSQL server to be copied to another PostgreSQL server. This allows the creation of redundant databases so that if there is a failure on one server, data can still be accessed on another server.

b. Clustering: Clustering technology allows multiple PostgreSQL servers to be combined into a single unit called a cluster. Thus, if there is a failure on one server, the system can still operate because resources can be taken from other servers within the cluster.

c. Load Balancing: Load balancing technology allows client connections to be distributed to multiple available PostgreSQL servers. Here we use HAproxy which allows the system to continue operating even if there is a failure on one server, because the connection can be redirected to another server that is still operating.

III. RESEARCH METHODOLOGY

This chapter can explain the research methodology used to implement High Availability on PostgreSQL in the OSS environment. The research methodology used includes the following stages:

A. Research Method

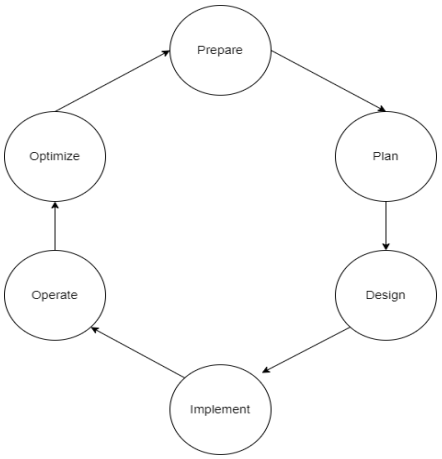

Fig. 3. PPDIOO Model [7]

The Fig. 3 was a methodology used by Cisco to assist organizations in effectively and efficiently planning, designing, implementing, operating, and maintaining their network solutions[7]. Here, I combined the PPDIOO Model with the concept of High Availability Architecture to enable the design and implementation of the required High Availability architecture. The PPDIOO Model consisted of the Prepare, Plan, Design, Implement, Operate, and Optimize stages.

1. Prepare Stage: In this stage, literature study and needs analysis are conducted to understand the requirements that must be met in implementing High Availability. This involves identifying the necessary components and understanding the desired features and functions.

2. Plan Stage: In this stage, based on the needs analysis, planning is done to determine the components that can be used in the High Availability architecture. This includes selecting the appropriate hardware, network topology, and required configurations and settings.

3. Design Stage: This stage involves the detailed design of the High Availability architecture. The PPDIOO Model helps in designing the selected components from the previous stage and ensuring they work together effectively. This includes database design, selection of suitable technologies and methods, and mapping of network structures.

4. Implement Stage: After the design is completed, the implementation stage begins. The PPDIOO Model assists in implementing the designed High Availability architecture, including hardware and software installation, component configuration, and integration with existing systems.

5. Operate Stage: In this stage, the implemented High Availability architecture is actively operated. Operations can monitor system performance, ensure database availability, perform routine maintenance, and handle any issues that arise.

6. Optimize Stage: The final stage in the PPDIOO Model is Optimize. In this stage, the performance of the High Availability architecture is evaluated, and any necessary improvements or enhancements are made while monitoring system performance. The operations team can analyze performance metrics, identify areas that require improvement, and implement appropriate solutions. The goal of this stage is to

continuously improve the performance and efficiency of the High Availability architecture according to business needs.

In implementing High Availability in the OSS environment, the PPDIOO Model can be used as a research method, focusing on the Plan, Design, and Implement stages. The Plan stage involves project planning and identifying business needs, while the Design stage involves designing the High Availability PostgreSQL solution, including aspects of design, architecture, and configuration. The Implement stage involves installation, configuration, and integration of the designed solution, as well as testing and verification of the implemented solution.

B. High Availability Architecture Design

In this research, High Availability Architecture refers to the Design stage of the PPDIOO Model, as explained in the previous stage. Based on the needs analysis, the design of the High Availability architecture was carried out, including the selection of required components such as network topology and configuration of each component [7]. We aimed to provide multiple coordinated database servers to improve database services and system availability. Our goal was to keep the database online at all times. The hardware that houses the database is a critical component. Even if the machine and its contents are perfect, failure or unexpected behavior of any element can cause downtime [3]. We provided two clusters, each consisting of four servers.

The first cluster, known as the "Master DB," functions as a storage for alarm and notification data. Meanwhile, the second cluster, known as "KPI and RAW KSO," is used to obtain raw data from the KSO database, which has specific packages. Each cluster consists of a dedicated HAproxy server, a leader (pg_node1), and two other servers acting as replicators. Through the port 5000 configuration, both clusters can exchange data with each other. Additionally, in our architecture design, we have also implemented a cron server that controls incoming packages to the system [10]. The cron server is responsible for scheduling and sending these packages through HAproxy to the servers within cluster 2.

With the presence of the cron server, we can effectively manage and control the flow of data within our system. This server ensures the speed and accuracy of delivering incoming data packages to our infrastructure [10]. Through integration with HAproxy, the cron server ensures that each package is smoothly and efficiently delivered to the servers within cluster 2. With this architecture design, we can achieve our goals of maintaining high system availability, improving efficiency, and ensuring seamless integration between components in our database infrastructure.

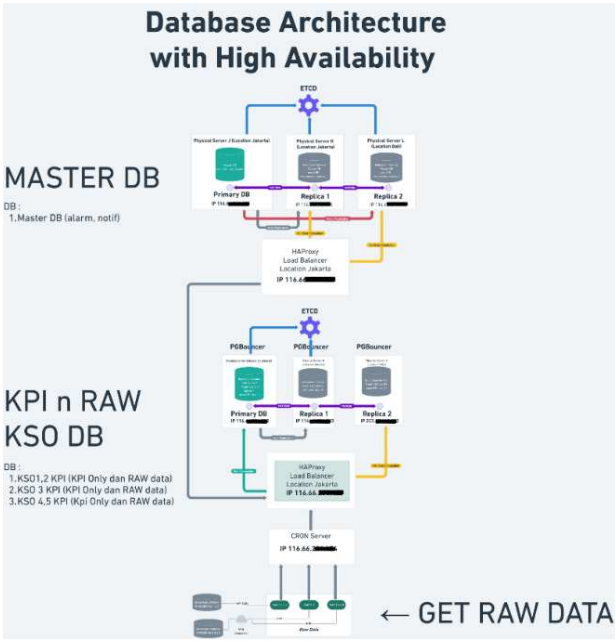

Fig 4. Result of Database Design Architecture with High Availability in OSS

B.1. High Availability Implementation

The Figure 4 was result after the architecture design is completed, the next step is to implement High Availability on PostgreSQL in the OSS environment. In this stage, the installation and configuration of each component in the High Availability architecture designed in the previous stage is carried out.

C. *Testing Method*

After the implementation process is completed, testing and evaluation can be carried out to ensure that High Availability on PostgreSQL in the OSS environment runs well and according to the established needs. Testing is carried out using the previously designed test scenarios, and performance measurements of the implemented system are conducted [6].

Testing and evaluating High Availability PostgreSQL in the OSS environment is carried out to measure the performance of the implemented system. Testing is carried out using a representative workload from applications running on the OSS server. The workload includes read and write operations with predetermined data sizes.

In this testing, four scenarios are conducted to test the performance of High Availability on PostgreSQL. These scenarios are as follows:

1. *Failover on the master node.*

In this scenario, a failure simulation is carried out on the node that acts as the master. The aim is to test whether the implemented High Availability system can perform failover correctly and without disrupting service availability.

2. *Write Performance Test.*

This scenario is conducted to test write performance, such as measuring response time for write operations on the PostgreSQL database. In this scenario, 29G of data is written to the database with a stable connection simultaneously.

3. *Read Performance Test.*

This scenario is conducted to test read performance, such as measuring response time for read operations on the PostgreSQL database. In this scenario, 29G of data is read from the database with a stable connection simultaneously.

4. *Monitoring Performance Test.*

In this scenario, testing is carried out to measure the response time of the system when switching leaders or downtime occurs on one of the servers. Testing is carried out using Zabbix, and notifications can be sent to Slack.

IV. RESULTS AND DISCUSSION

A. *High Availability Implementation Results on PostgreSQL in OSS Environment*

In this research, High Availability implementation was carried out on PostgreSQL in the OSS environment. This implementation involves the use of PostgreSQL replication and clustering technology to ensure the availability of the database system. Here are the results of the High Availability implementation on PostgreSQL in the OSS environment:

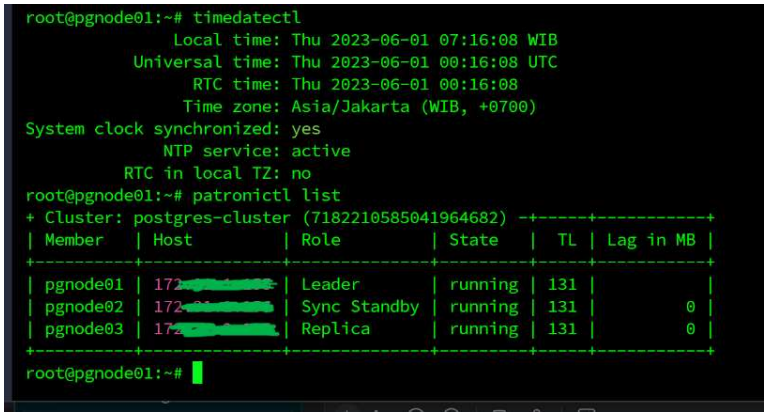

Fig 5. Result of Postgres-Cluster on Cluster 1

A.1. PostgreSQL Replication Configuration

The Figure 5 showed the result of the implementation of the PostgreSQL replication configuration was done using streaming replication method. There is one master node responsible for receiving data writes and several slave nodes receiving data replication from the master node. Every data change that occurs on the master node can be automatically replicated to the slave nodes.

A.2. PostgreSQL Clustering Configuration

In addition to replication, clustering configuration was also done on PostgreSQL using technologies such as Haproxy, pgbouncer, patroni, and others. These tools allow several PostgreSQL servers to be combined into one unit called a cluster. In this implementation, there is one coordinator node responsible for managing data distribution and several data nodes that store data in a distributed manner. The results of this implementation can be monitored via IP in the browser.

B. Cluster Analysis and Evaluation

Table 1 performed analysis and evaluation of the High Availability implementation results on PostgreSQL in the OSS environment, several tests and observations were performed on the system. Here are some aspects that were evaluated:

TABLE 1
EXPERIMENT RESULTS IN CLUSTER ENVIRONMENTS

Action	Start time	End time	Total
Failover	Sat 2023-05-27 07:51:45 WIB	Sat 2023-05-27 07:51:58 WIB	13.41s
Switch leader	Sat 2023-05-27 07:46:45 WIB	Sat 2023-05-27 07:47:05 WIB	20.43s
Recovery Time	Manually	Manually	± 12.55s

B.1. System Availability

With PostgreSQL replication and clustering, the system can still operate even if there is a failure on one of the nodes. Data remains available and can be accessed through other nodes that are still

operational. This increases the overall system availability called high availability. Table 2 was the result of testing in a cluster environment.

TABLE 2
EXPERIMENT RESULTS IN CLUSTER ENVIRONMENTS

Action	Backup-Restore	Mirroring	Availability
Response Time	± 360s	± 240s	± 240s
Database Availability	Available on all nodes	Available on all nodes	Available on all nodes
Failover	Auto	Auto	Auto Failover
Data Integrity	No data gaps	Has been checked	Synchronized data automatically

B.2. Recovery Time

In case of failure, system recovery time becomes an important factor. In this implementation, system recovery time depends on the failover time from the master node to the slave node. The evaluation was done to measure the system recovery time during a failure and ensure that the recovery time meets the set target.

B.3. System Performance

System performance was also evaluated in this implementation. Tests were conducted to measure system performance in normal conditions and when replication or clustering is running. Performance parameters such as response time, data access speed, and system workload were analyzed to ensure that the High Availability implementation does not significantly sacrifice system performance.

TABLE 3
EXPERIMENT RESULTS IN DATABASE ENVIRONMENTS

Action	Start time	End time	Total
Write Performance Test	Sat 2023-05-28 09:54:43 WIB	Sat 2023-05-28 10:04:43 WIB	600s
Read Performance Test	Sat 2023-05-28 09:46:43 WIB	Sat 2023-05-28 10:02:43 WIB	960s

The Table 3 showed results of the tests that indicated data writing performance on the PostgreSQL database was good. The average data writing time was 600 seconds. Additionally, the test results showed that data reading performance was also good, with an average of 960 seconds.

B.4. Monitoring

Monitoring on PostgreSQL using Zabbix and Slack provided important information about the health and performance of the database system. The following are the monitoring results obtained through the integration between Zabbix and Slack in the context of PostgreSQL high availability:

1. Resource Usage: Zabbix monitored resource usage such as CPU, memory, and disk space on each node. Notifications were received if resource usage reached a certain limit or if there was a significant increase. This information was sent to Slack, enabling action to optimize resource usage.
2. Figure 6 Showed the Connection and Replication Status: Zabbix monitored the connection status between nodes in the high availability cluster. Direct notifications were received if there were connection problems or replication failures between nodes. This information was sent to Slack, facilitating immediate response and problem resolution.
3. Query Performance: Zabbix helped monitor query performance in PostgreSQL. Query execution time, the number of calls, and other query statistics were tracked. If there were slow queries or queries impacting the overall system performance, notifications were received through Slack to promptly address the issue.

C. *Discussion*

In this research, the implementation of High Availability was carried out on PostgreSQL in the OSS environment using replication and clustering technologies. Replication was performed using streaming replication method, where there was one master node that received data writes and several slave nodes that automatically replicated data from the master node. Additionally, PostgreSQL clustering was configured using technologies such as Haproxy, pgbouncer, patroni, and others, combining multiple PostgreSQL servers into one unit (cluster) with one coordinator node and several datanode nodes that store data in a distributed manner.

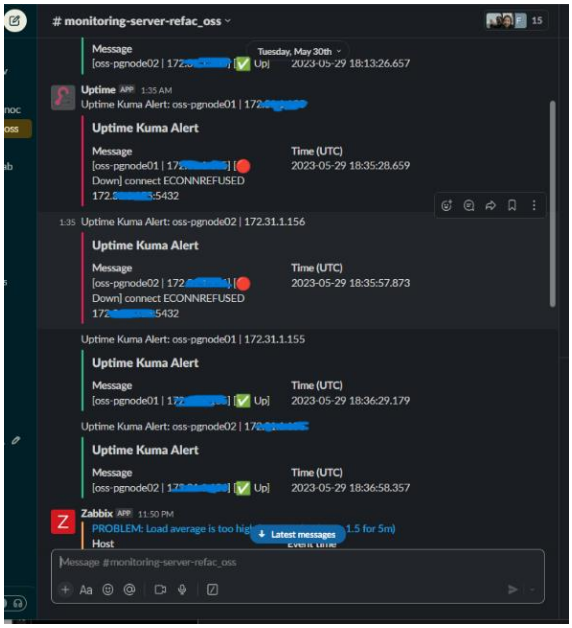

Fig 6. Result of Send Notification from Zabbix Monitoring to Slack

The results of High Availability implementation showed that the database system remained operational and data remained available through other nodes even in the event of failure on one of the nodes[3]. Overall system availability increased (high availability). Testing and evaluation were conducted to analyze the performance and effectiveness of the implementation, including the recovery time during failures (failover). The performance of writing and reading data in the PostgreSQL database was quite good, with an average writing time of about 600 seconds and reading time of about 960 seconds.

Monitoring with Zabbix and Slack provided important information about the health and performance of the database system, including resource utilization and connection status, as well as replication between nodes in the high availability configuration. This enabled quick response to potential issues. Overall, the implementation of High Availability on PostgreSQL in the OSS environment successfully increased system availability and performance effectively.

V. CONCLUSION

The implementation of High Availability on PostgreSQL in the OSS environment successfully increased the availability and reliability of the database system significantly through PostgreSQL replication and clustering. Failover from the master node to the slave node can be done automatically or manually with a recovery time of approximately ±12.55 seconds, reducing downtime and maintaining service availability. The performance evaluation of the system showed that the data writing and reading performance on the PostgreSQL database was good, with an average data writing time of about 600 seconds and data reading time of about 960 seconds. The High Availability implementation did not sacrifice system performance, allowing users to access and manipulate data efficiently.

The effective integration of Zabbix and Slack in monitoring provided essential information about the health and performance of the system. The IT team could promptly know the connection status and replication between nodes in the high availability configuration, as well as monitor resource usage on each node for timely optimization.

Overall, the implementation of High Availability on PostgreSQL in OSS yielded positive results with more reliable and responsive services. This solution can be a dependable alternative for organizations seeking to enhance the availability and performance of their database systems. The research findings are expected to be beneficial for the development of database systems and IT infrastructure in the future.

I. REFERENCES

[1] M. Suryanto, Suryanto. "Implementasi clustering database server menggunakan pgcluster untuk optimalisasi kinerja sistem basis data," *Jurnal Khatulistiwa Informatika* 1.1: 134-143. Feb 2015.

[2] M. R. Sulistiawan. “Implementasi kubernetes dan patroni postgresql cluster sebagai infrastruktur aplikasi keluhan pelanggan,” Diss. Universitas teknologi digital indonesia, 2023.

[3] S. M. Thomas. “Master over 100 recipes to design and implement a highly available server with the advanced features of PostgreSQL,*” PostgreSQL High Availability Cookbook Second Edition*. Feb 2017.

[4] H. J. Schonig. *“*Leverage the power of PostgreSQL replication to make your databases more robust, secure, scalable, and fast.*”* PostgreSQL Replication Second Edition. July 2015.

[5] P. Ahlawat, S. Dahiya, “Choosing between high availability solutions in microsoft sql server”, 4(6), 387–391. 2015.

[6] A. Nugraha, and R. Amin. "Analisis metode replikasi sistem basis data di pusintek kementerian keuangan." Jurnal SISKOM-KB *(Sistem Komputer dan Kecerdasan Buatan)* 6.1: 73-77. 2022.

[7] A. Nirwana, M. A. Hasibuan, and U. Y. K. S. Hediyanto. "Perancangan Network Structure Data Center Untuk Meningkatkan Availability Jaringan Di Pemerintah Kabupaten Bandung Menggunakan Standar TIA-942 Dengan Metode PPDIOO Life-cycle Approach." *JRSI (Jurnal Rekayasa Sistem dan Industri)* 5.01: 8-14. 2018.

[8] H. Garcia-Molina, and C. A. Polyzois. "Issues in disaster recovery." *1990 Thirty-Fifth IEEE Computer Society International Conference on Intellectual Leverage*. IEEE Computer Society, 1990.

[9] H. Xiong, F. Fowley, and C. Pahl. "A database-specific pattern for multi-cloud high availability and disaster recovery." *Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2015, Taormina, Italy, September 15-17, 2015, Revised Selected Papers 4*. Springer International Publishing, 2016.

[10] W. A. Yuliono, A. Prihanto. "Sinergi Replikasi Server dan Sistem Failover pada Database Server untuk Mereduksi Downtime Disaster Recovery Planing (DRP)." *Journal of Informatics and Computer Science (JINACS)* 3.01: 29-38. 2021.

[11] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis, & D. Anagnostopoulos, “MongoDB Vs

PostgreSQL: A comparative study on performance aspects," *GeoInformatica*, *25*, 243-268, 2021

[12] PGCluster Documentation. "*Clustering System of PostgreSQL chapter 27, High Availability, Load Balancing, and Replication"*. Retrieved from: https://www.postgresql.org/docs/current/high-availability.html/ . August 2023.

[13] E. L. Huamaní, P. Condori, and A. Roman-Gonzalez. "Implementation of a beowulf cluster and analysis of its performance in applications with parallel programming." *International Journal of Advanced Computer Science and Applications* 10.8. 2019

[14] Yugabyte. PostgreSQL High Availability. Retrieved from: https://www.yugabyte.com/postgresql/postgresql-high-availability/. August 2023.

[15] M. Sadikin, and R. Yusuf. "Load balancing clustering on moodle LMS to overcome performance issue of e-learning system." *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 17.1: 131-138. 2019.