**Jurnal Teknologi Informasi dan Pendidikan**
Volume 19, No. 2, September 2026
https://doi.org/10.24036/jtip.v19i2.1129

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

# Performance Trade-offs of Synchronous and Asynchronous Replication in Patroni-based PostgreSQL Clusters

**I Putu Suwidnyana Putra[1], I Nyoman Gede Arya Astawa[1*✉], Luh Gede Putri Suardani [1]**
[1]Department Information Technology, Politeknik Negeri Bali, Badung, Indonesia
*Corresponding Author: arya_kmg@pnb.ac.id*

**Article Information**

***Article history:***

*No. 1129*
*Rec. March 23, 2026*
*Rev. May 29, 2026*
*Acc. June 03, 2026*
*Pub. June 20, 2026*
*Page. 1628 – 1638*

***Keywords:***

- High Availability
- PostgreSQL
- Patroni
- Replication
- Failover

**ABSTRACT**

*This research aims to evaluate the effectiveness of High Availability (HA) architectures and analyze the performance trade-offs of database replication modes in critical environments. The study employs an experimental method using a distributed cluster consisting of four virtualized nodes. PostgreSQL is used as the core database, orchestrated by Patroni and Etcd for failover management, with HAProxy as the load balancer. Performance was measured using pgbench with TPC-B standards under varying concurrent loads (10, 30, and 50 clients). The results demonstrate that the Patroni cluster successfully performed auto-failover with an average Recovery Time Objective (RTO) of under 30 seconds in normal conditions, which increased to 75 seconds during peak workloads due to resource contention. Benchmarking reveals that Asynchronous replication achieved a peak throughput of 247 ops/s, while Synchronous replication guaranteed absolute data integrity (RPO=0) but incurred a significant latency increase, reaching 21.36 ms under a 30-client load due to 98.0% CPU saturation. This study concludes that the proposed architecture effectively eliminates Single Point of Failure (SPOF), providing a critical reference for system architects in balancing transactional speed and data consistency.*

***How to Cite:***

Putra, I.P.S., & et al. (2026). Performance Trade-offs of Synchronous and Asynchronous Replication in Patroni-based PostgreSQL Clusters. Jurnal Teknologi Informasi Dan Pendidikan, 19(2), 1628-1636. https://doi.org/10.24036/jtip.v19i2.1129



## 1. INTRODUCTION

In today's digital era, data availability has become a most crucial element in operational continuity, particularly in the banking sector that demands real-time

transactions. Single-node server infrastructures possess high vulnerability to system failures, with error rates increasing significantly alongside transaction spikes [1]. This condition creates a Single Point of Failure (SPOF) that forces administrators to perform manual recovery, triggering prolonged downtime and risks of data inconsistency [2].

The implementation of High Availability (HA) Clustering technology using PostgreSQL serves as a solution to ensure service continuity [3]. However, traditional cluster management remains susceptible to split-brain issues. This research applies the Patroni orchestrator, which utilizes a distributed consensus algorithm to promote a replica into a new leader automatically without human intervention [4]. Nevertheless, the monitoring aspect is often overlooked, leaving administrators struggling to validate cluster health in real-time.

Based on previous studies, the majority of HA research focuses solely on the functional success of the failover mechanism. There is a research gap regarding quantitative data analysis related to the impact of Synchronous and Asynchronous replication modes within the Patroni architecture. This absence makes it difficult to determine the optimal configuration between absolute data security and transaction speed.

To address these challenges, this study aims to design a Patroni-based PostgreSQL HA infrastructure equipped with a centralized monitoring system using Prometheus and Grafana [5]. The contribution of this research is providing a comparative analysis of replication performance using TPC-B standards to dissect the trade-offs between data integrity and transaction speed. This approach is expected to provide a robust infrastructure solution for critical systems by presenting comprehensive analytical data.

## 2. RESEARCH METHOD

This research employs an experimental method by re-engineering the system within a virtual simulation environment. The architecture adopts a Distributed Database Cluster (Shared-Nothing) model consisting of four virtual machines (VMs) with the following roles: three nodes are allocated for the PostgreSQL 15 cluster orchestrated by Patroni and Etcd (1 Leader, 2 Replicas), while one node acts as the access layer using HAProxy for application traffic load balancing.

The monitoring system is configured using metrics collected by Prometheus and visualized through a Grafana dashboard. The comprehensive system architecture is presented in Figure 1.

**Jurnal Teknologi Informasi dan Pendidikan**
Volume 19, No. 2, September 2026
https://doi.org/10.24036/jtip.v19i2.1129

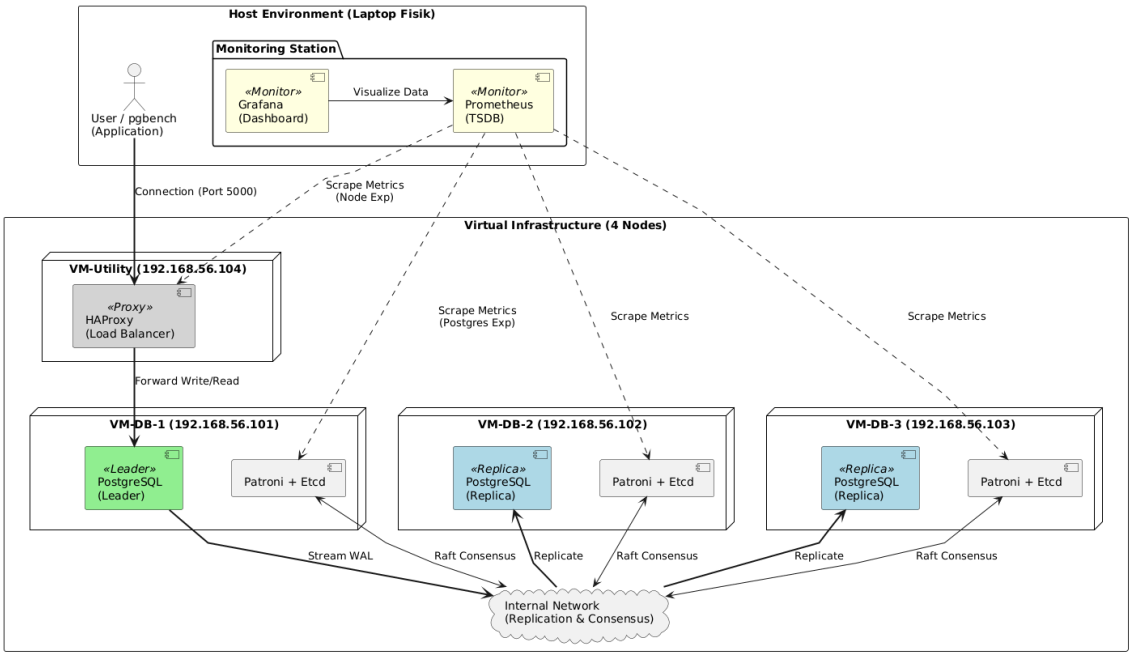


**Figure 1.** System Architecture

Figure 1 Description: Figure 1 illustrates the traffic flow and interconnection between nodes. HAProxy acts as a single gateway that receives requests from the application and forwards them to the Leader by checking the Patroni API status. Data synchronization between the Leader and Replicas is performed via streaming replication, while Etcd serves as the Distributed Configuration Store (DCS) to maintain cluster leadership consensus.

The testing scenario is divided into two main stages to test system reliability and performance:

1. Functional Testing (Black Box Testing): Conducted using Chaos Engineering methods by forcibly shutting down (power off) the Leader node to validate the auto-failover mechanism and measure the Recovery Time Objective (RTO).
2. Performance Testing (Benchmark Testing): Utilizes the pgbench tool with TPC-B transaction standards at concurrent loads of 10, 30, and 50 clients for 60 seconds. The independent variable in this test is the replication mode (Synchronous vs. Asynchronous) to compare Throughput (TPS) and Latency.

Details of the hardware specifications and testing parameters are presented in Table 1 and Table 2 below:


P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

**Table 1.** pgbench Performance Testing Scenario Parameters

| No | Independent Variable (Replication Mode) | Client Load (Concurrent Clients) | Test Duration | Analysis Objective |
|---|---|---|---|---|
| 1 | Asynchronous (*synchronous_commit = off*) | 10, 30, 50 Clients | 60 Seconds | Measuring the maximum performance baseline (highest throughput) the infrastructure can handle |
| 2 | Synchronous (*synchronous_commit = on*) | 10, 30, 50 Clients | 60 Seconds | Measuring the performance degradation (overhead) occurring because the system must wait for write confirmation from the Replica |

Table 1 Description (Scenario Parameters): The data in Table 1 shows that the testing focuses on comparing two main parameters: synchronous_commit = on and off. Testing was conducted for 60 seconds per scenario to ensure the stability of the generated *throughput* data before the average values were taken.

**Table 2.** Hardware and Virtual Machine Infrastructure Specifications

| No | Component | Host Specifications | Virtual Machine (VM) Allocation |
|---|---|---|---|
| 1 | Processor (CPU) | AMD Ryzen 5 / Intel Core i5 (6 Cores) | 2 vCPU (per VM) |
| 2 | Memory (RAM) | 16 GB DDR4 | 2 GB (Node Database x 3 VM)<br>1 GB (Node Load Balancer x 1 VM) |
| 3 | Storage (Disk) | SSD NVMe 512 GB | 20 GB Virtual Disk (per VM) |
| 4 | Jaringan (Network) | Host-Only Network Adapter | Virtual Network Adapter (Internal IP) |

Table 2 Description (Specifications): The specifications in Table 2 are designed to simulate a standard production server environment with 2 vCPU allocated per VM. The use of NVMe SSD aims to minimize bottlenecks on the disk I/O side, allowing the performance analysis to focus entirely on replication protocol overhead and CPU utilization.

**Jurnal Teknologi Informasi dan Pendidikan**
Volume 19, No. 2, September 2026
https://doi.org/10.24036/jtip.v19i2.1129

## 3. RESULTS AND DISCUSSION

### 3.1. Failover Implementation and Reliability

The distributed database infrastructure was successfully implemented. Patroni effectively identified one node holding the Etcd consensus key as the Leader, while HAProxy dynamically routed data traffic exclusively to the active Leader node.

In the fault injection test (Hard Failure), when the Leader node was abruptly shut down, Patroni detected the loss of heartbeat, and the leader key in Etcd expired according to the Time to Live (TTL) parameter. This immediately triggered a Leader Election, promoting a Replica node to become the new Leader. The cluster leadership transition process was recorded in real-time on the monitoring system (Figure 2).


**Figure 2.** Shows the Leader role transition on the terminal during a failover.

The time required from the moment of failure until the service was fully restored (Recovery Time Objective or RTO) was recorded at under 30 seconds in the normal scenario. This is evidenced by the cluster status returning to a healthy state with the new leadership arrangement (Figure 3). This HA mechanism successfully eliminated the SPOF without requiring manual intervention.

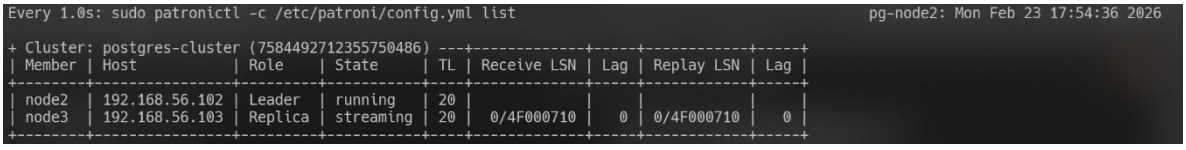


**Figure 3.** Shows The Patroni Cluster Status After the Failover Process Was Successfully Completed.


P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

### 3.2. Comparative Analysis of Replication Performance

Benchmark testing using pgbench provided empirical evidence regarding significant performance differences between the two replication modes (Figure 4):

1) Asynchronous Mode: Produced a maximum average throughput of 247 ops/s at a 50-client load. This was achieved because the protocol does not require an Acknowledgement (ACK) signal from the Replica. The Leader immediately issues a success status after the data is written to the local log (Write Ahead Log). However, this mode leaves a data loss risk gap (Recovery Point Objective or RPO > 0) due to the delay in data transmission to the Replica..
2) Synchronous Mode: Produced an average throughput of 161 ops/s at a 10-client load (a 14.3% decrease). This performance is heavily influenced by network latency and Round-Trip Time (RTT) because every transaction must wait for write confirmation from the Replica (Quorum Commit). This causes the transaction pipeline to stall and triggers a spike in CPU utilization up to 99.5% due to accumulated process queues, while ensuring absolute data integrity (RPO=0)

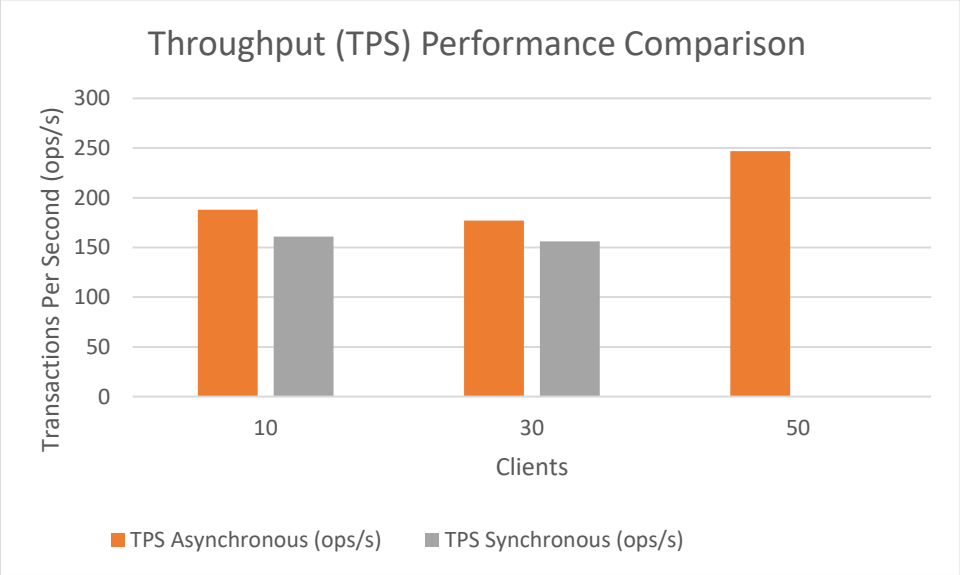


**Figure 4.** Throughput (TPS) Comparison between Synchronous and Asynchronous Replication Modes

Note that under a 50-client workload, the Synchronous mode failed to complete the benchmark due to connection timeouts and severe CPU saturation. Thus, no throughput data is recorded for Synchronous mode at 50 clients.

**Jurnal Teknologi Informasi dan Pendidikan**
Volume 19, No. 2, September 2026
https://doi.org/10.24036/jtip.v19i2.1129

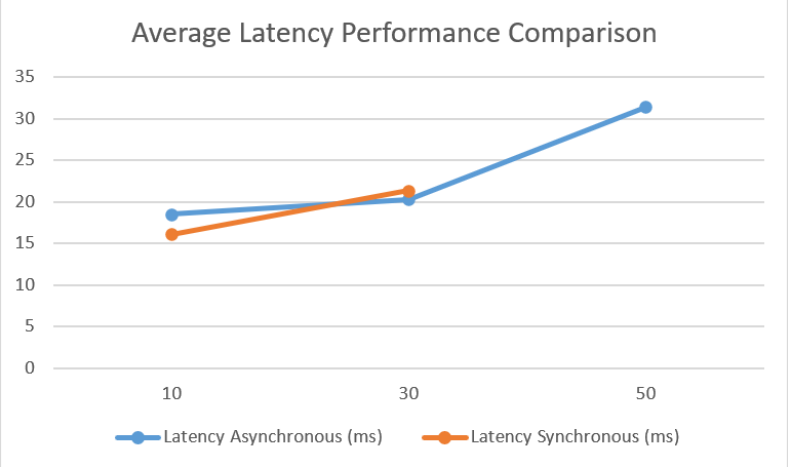


**Figure 5.** Latency Comparison between Synchronous and Asynchronous Replication Modes

The performance degradation in Synchronous mode directly correlates with spikes in resource utilization. At a 30-client load, latency increased sharply to 21.36 ms alongside CPU usage on the Leader node reaching 98.0% (Figure 6). This indicates that strict data consistency imposes a heavy computational burden on the infrastructure.


**Figure 6.** Shows The Leader Node CPU Saturation Graph During Synchronous Mode Testing

The significant write overhead observed in the Synchronous mode approximately 14.4% drop in TPS aligns with the architectural trade-offs documented in previous studies. While researchers like Pratama demonstrated that automated failover via Patroni can drastically minimize RTO to under 30 seconds, this study provides the missing empirical quantification of the replication performance toll. Compared to manual replication managers like Repmgr which often suffer from longer split-brain resolutions, the integration of Patroni and Etcd ensures deterministic consensus at the cost of predictable network round-trip latencies during synchronous verification.

### 3.3. System Observability (Monitoring)

Beyond the failover aspects, this research successfully delivered real-time visibility through a centralized Grafana dashboard (Figure 7). Monitoring TPS graphs, node health status, and replication lag metrics allows administrators to detect anomalies (such as service failures or load spikes) instantly. This feature provides early detection capabilities that are far more efficient compared to conventional manual log analysis methods via CLI.


P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

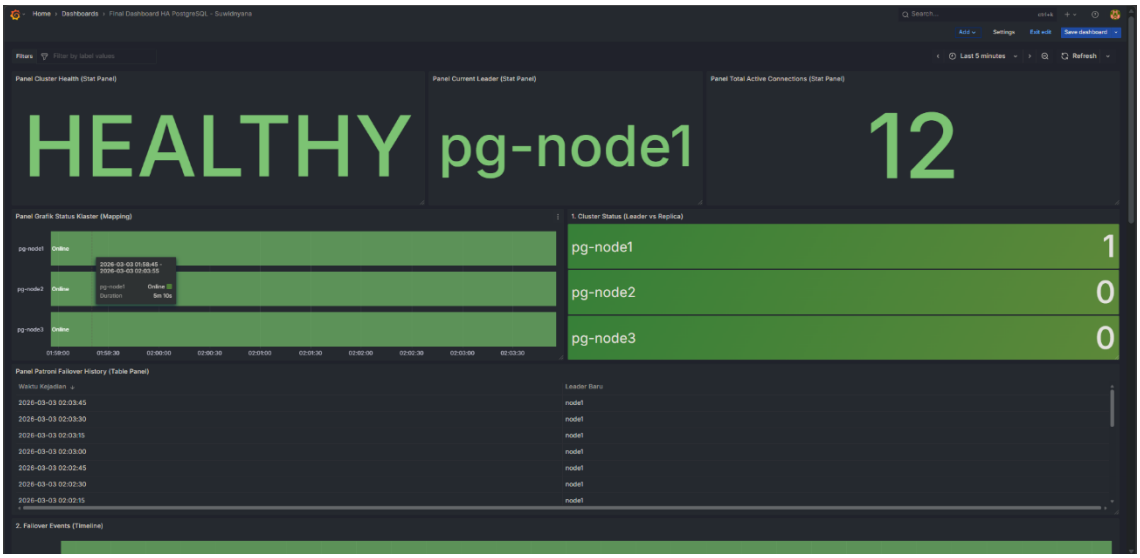


**Figure 7.** Shows The Monitoring Dashboard for Database Cluster Supervision.

## 4. CONCLUSION

This research has successfully designed a robust database infrastructure and extensively analyzed the performance trade-offs of replication modes. The primary findings indicate that Patroni orchestration is capable of performing auto-failover with an effective RTO of under 30 seconds under normal conditions. However, the selection of Synchronous replication mode results in a significant decrease in throughput due to write overhead. The implication of these findings is that the banking system must prioritize Synchronous replication to maintain absolute customer balance integrity (RPO=0), but this must be balanced with higher vCPU allocations to manage latency spikes.

The contribution of this research is the provision of a tested HA framework and quantitative benchmarking data as a basis for configuration decision-making in critical systems. For future development, the implementation of connection pooling mechanisms such as PgBouncer is highly recommended to optimize system stability under massive traffic loads exceeding 50 clients.

This study is limited to an isolated virtual environment with specific node specifications (2 vCPU, 2 GB RAM) and internal host networks. Therefore, performance figures may vary in real-world data centers with cross-building network latencies.

## REFERENCES

[1] R. Pratama, "Pengaruh infrastruktur high availability terhadap kinerja dan responsivitas web server di google cloud platform: analisis perbandingan," Skripsi, Universitas Muhammadiyah

Surakarta, Surakarta, 2023.
[2] W. A. Yuliono and 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), vol. 3, no. 1, pp. 29-38, 2021.
[3] S. H. Ahsana, M. B. Syahputra, A. F. F. M. Putri, and A. A. Prasetyo, "Analisis perbandingan performa antara mysql dan postgresql," dalam Prosiding Seminar Nasional Teknologi dan Sistem Informasi (SITASI) 2023, Surabaya, 2023, pp. 13-18.
[4] I. Ollong and D. S. Kusumo, "Development of high availability database infrastructure for oss projects with monitoring systems in cloud computing environments," International Journal on ICT, vol. 9, no. 2, pp. 41-52, 2023.
[5] F. S. Musthofa, "Rancang bangun sistem monitoring performa database server postgresql dengan notifikasi email menggunakan zabbix yang terintegrasi grafana," Skripsi, Politeknik Negeri Jakarta, Depok, 2023.
[6] E. S. Alim, "Implementasi load balancing pada google cloud platform," Jurnal Teknik Informatika, vol. 8, no. 1, 2025.
[7] D. Irwan, dkk., "Service high availability pada native server dan virtual server menggunakan proxmox," Jurnal Teknologi Informasi, vol. 6, no. 2, 2020.
[8] M. Rafli and A. Aryo, "Analisis perbandingan efisiensi metode trigger, timestamp dan log-based cdc," Jurnal Pengembangan Teknologi Informasi, vol. 2, no. 5, 2017.
[9] M. Suryanto, "Implementasi clustering database server menggunakan pgcluster untuk optimalisasi kinerja sistem basis data," Jurnal Khatulistiwa Informatika, vol. 3, no. 2, 2015.
[10] Asriyar and U. Sutendi, "Implementasi sistem replikasi database postgresql master slave repmgr," Jurnal Teknologi Informasi dan Komunikasi, vol. 7, no. 1, 2019.
[11] L. Wulandari, dkk., "Infrastruktur high-available lms universitas menggunakan least connected lb," Jurnal Sistem Informasi, vol. 10, no. 2, 2022.
[12] Muliawan and D. Iskandar, "Penerapan replikasi synchronous pada aplikasi data krs online," Jurnal Algoritma, vol. 20, no. 1, 2023.
[13] Satyarca, "Implementasi back up dan replikasi data yang efisien dengan metode synchronous," Jurnal Teknik Informatika, vol. 5, no. 2, 2019.
[14] F. Apriliansyah, "Implementasi load balancing pada web server menggunakan nginx," Jurnal Informatika, vol. 7, no. 1, 2020.
[15] Gusti, dkk., "Performance analysis high availability web server cluster gke," Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 5, no. 10, 2021.
[16] PostgreSQL Global Development Group, "Chapter 27. High Availability, Load Balancing, and Replication," PostgreSQL 15 Documentation, 2023. [Online]. Available: https://www.postgresql.org/docs/15/high-availability.html
[17] Zalando SE, "Patroni: A Template for PostgreSQL HA with ZooKeeper, etcd, or Consul," GitHub Repository, 2023. [Online]. Available: https://github.com/zalando/patroni
[18] Etcd Core Team, "etcd: A distributed, reliable key-value store for the most critical data of a distributed system," Raft Consensus Implementation, 2023. [Online]. Available: https://etcd.io/docs/


P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id