JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 14 No. 2, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v14i2.10140

PERBANDINGAN KINERJA GALERA CLUSTER DAN INNODB CLUSTER PADA BASIS DATA AKADEMIK

Raka Putra Atmaja[1*], Kadek Yota Ernanda Aryanto[2], Ketut Agus Seputra[3]

[1,2,3]Universitas Pendidikan Ganesha; Jl. Udayana No.11, Singaraja, Bali 81116, Telp. (0362) 22570, Fax. (0362) 25735.

Keywords:
Galera Cluster;
InnoDB Cluster;
replikasi sinkron;
high availability;
Sysbench.

Corespondent Email:
raka.putra@student.undiksha.ac.id

Abstrak. Sistem basis data yang digunakan pada layanan akademik membutuhkan konsistensi data dan ketersediaan layanan yang tinggi. Penelitian ini menyimulasikan implementasi *Galera Cluster* sebagai solusi replikasi sinkron *multi-master* dan membandingkannya dengan *InnoDB Cluster* sebagai sistem *baseline*. Metode penelitian menggunakan pendekatan *Network Development Life Cycle* yang dibatasi pada tahap analisis, desain, simulasi prototipe, implementasi, dan monitoring. Pengujian dilakukan pada lingkungan *virtual* tiga *node* menggunakan *Sysbench*, *Prometheus*, dan *Grafana*. Metrik yang diamati meliputi *throughput*, *latency*, penggunaan *CPU*, penggunaan memori, waktu replikasi, *Recovery Time Objective*, dan *Mean Time To Recovery*. Hasil rekapitulasi rata-rata menunjukkan bahwa *Galera Cluster* menghasilkan *throughput* lebih tinggi, *latency* lebih rendah, serta penggunaan memori lebih efisien pada sebagian besar skenario. *Galera Cluster* juga memiliki *RTO* lebih kecil dibandingkan *InnoDB Cluster*, meskipun *MTTR InnoDB Cluster* lebih rendah. Dengan demikian, *Galera Cluster* lebih sesuai untuk meningkatkan ketersediaan layanan dan konsistensi data pada sistem basis data terdistribusi, sedangkan *InnoDB Cluster* lebih cepat dalam pemulihan *node*.

Abstract. *Database systems used in academic services require high data consistency and service availability. This study simulates the implementation of Galera Cluster as a synchronous multi-master replication solution and compares it with InnoDB Cluster as the baseline system. The research method applies the Network Development Life Cycle approach, limited to analysis, design, simulation prototyping, implementation, and monitoring stages. Testing was carried out in a three-node virtual environment using Sysbench, Prometheus, and Grafana. The observed metrics include throughput, latency, CPU usage, memory usage, replication time, Recovery Time Objective, and Mean Time To Recovery. The average recap results show that Galera Cluster provides higher throughput, lower latency, and more efficient memory usage in most scenarios. Galera Cluster also achieves a lower RTO than InnoDB Cluster, although InnoDB Cluster has a lower MTTR. Therefore, Galera Cluster is more suitable for improving service availability and data consistency in distributed database systems, while InnoDB Cluster is faster in node recovery.*

1. PENDAHULUAN

Sistem basis data merupakan komponen utama dalam pengelolaan layanan digital pada organisasi. Pada lingkungan perguruan tinggi, basis data digunakan untuk mendukung layanan akademik, keuangan, kepegawaian, dan sistem informasi lain yang harus tersedia secara berkelanjutan. Kegagalan pada *server* basis data dapat menyebabkan gangguan akses layanan, keterlambatan proses operasional, dan potensi inkonsistensi data.

Oleh karena itu, dibutuhkan arsitektur basis data yang mampu menjaga ketersediaan layanan, redundansi, serta konsistensi data antar *server* [1] [2]. *UPA TIK* Universitas Pendidikan Ganesha telah menerapkan sistem klaster basis data menggunakan *InnoDB Cluster*. Namun, arsitektur yang digunakan masih berorientasi pada *single-primary*, sehingga operasi tulis bergantung pada satu *node* utama, sedangkan *node* lain berperan sebagai replika. Kondisi ini dapat menimbulkan ketergantungan terhadap *primary node* dan meningkatkan risiko gangguan layanan ketika *node* utama mengalami kegagalan. Selain itu, proses replikasi yang tidak selalu berjalan secara serempak berpotensi menimbulkan keterlambatan sinkronisasi data antar *node* [3] [4].

Galera Cluster menjadi alternatif yang relevan karena mendukung *synchronous multi-master replication*. Melalui pendekatan ini, setiap *node* dapat menerima operasi baca dan tulis, kemudian perubahan data direplikasi secara sinkron ke *node* lain. Mekanisme tersebut dapat mengurangi ketergantungan terhadap satu *node* utama serta membantu menjaga konsistensi data pada lingkungan sistem terdistribusi [5] [6]. Meskipun demikian, performa *Galera Cluster* tetap perlu dibandingkan dengan *InnoDB Cluster* agar karakteristik keduanya dapat dipahami secara lebih objektif. Pada konteks *UPA TIK* Undiksha, kajian mengenai *early warning* kebakaran dan smart lock pusat data menunjukkan bahwa pengelolaan pusat data membutuhkan pemantauan serta pengamanan yang berkelanjutan [7], [8]. Penelitian ini bertujuan menyimulasikan implementasi *Galera Cluster* dan membandingkannya dengan *InnoDB Cluster* berdasarkan metrik sinkronisasi, performa, dan *high availability*. Dengan demikian, hasil artikel berfokus pada rekapitulasi performa dan simpulan perbandingan awal antara kedua arsitektur *cluster*.

2. TINJAUAN PUSTAKA

2.1 Sistem Basis Data dan Replikasi

Sistem basis data adalah sistem yang dirancang untuk menyimpan, mengelola, dan mengambil data secara efisien dengan bantuan *Database Management System*. Dalam konteks layanan digital, basis data menjadi pusat pengelolaan informasi sehingga ketersediaannya harus dijaga [9]. Replikasi basis data digunakan untuk menduplikasi data dari satu *node* ke *node* lain agar sistem tetap tersedia ketika terjadi gangguan. Replikasi dapat dilakukan secara asinkron maupun sinkron. Replikasi asinkron umumnya memiliki respons yang lebih cepat, tetapi memiliki risiko keterlambatan replikasi. Replikasi sinkron memastikan data diterapkan ke *node* lain secara lebih konsisten sebelum transaksi dianggap berhasil [10].

2.2 InnoDB Cluster

InnoDB Cluster merupakan solusi *high availability* pada *MySQL* yang memanfaatkan *MySQL Group Replication*. Sistem ini dapat berjalan pada mode *single-primary* maupun *multi-primary*. Pada mode *single-primary*, satu *node* berperan sebagai *primary node* yang menangani operasi tulis, sedangkan *node* lainnya berperan sebagai *secondary node*. *InnoDB Cluster* mendukung mekanisme *failover* ketika *primary node* mengalami gangguan, tetapi pada konfigurasi *single-primary* proses penulisan tetap bergantung pada satu *node* utama [11], [12].

2.3 Galera Cluster

Galera Cluster merupakan teknologi *database cluster* berbasis replikasi sinkron *multi-master*. Setiap *node* dalam *cluster* dapat berperan sebagai *master* dan menerima transaksi penulisan. Perubahan data yang terjadi pada satu *node* akan disebarkan ke *node* lain menggunakan mekanisme *write-set replication*. *Galera Cluster* juga mendukung quorum untuk mencegah split-brain scenario, sehingga cocok digunakan pada sistem yang membutuhkan konsistensi dan ketersediaan tinggi [13].

2.4 High Availability dan Sysbench

High availability merupakan kemampuan sistem untuk mempertahankan layanan ketika terjadi kegagalan pada salah satu komponen. Pada sistem basis data, *high availability* dapat diamati melalui waktu pemulihan layanan dan kemampuan *node* untuk kembali bergabung ke *cluster*. *Sysbench* digunakan sebagai alat benchmark untuk menghasilkan *workload OLTP* dan mengukur performa *database* berdasarkan *throughput* dan *latency* [14] [15]. Penelitian sebelumnya juga

menunjukkan bahwa perbandingan metode komputasi dapat dilakukan melalui metrik kinerja yang terukur untuk mengetahui kelebihan dan kekurangan masing-masing pendekatan [16]. Kajian terkait di lingkungan Undiksha juga menunjukkan bahwa *high availability* dan *load balancing* pada SIAK Undiksha dapat dievaluasi menggunakan metrik *MTTR*, *availability, throughput, latency* [17] [18]. Evaluasi performa aplikasi menggunakan *load testing* juga dapat memanfaatkan metrik *throughput*, *latency*, CPU, dan memori [19]. Pengelolaan jaringan dan otomasi bandwidth juga berkaitan dengan kestabilan layanan infrastruktur *TIK* [20], sedangkan evaluasi jaringan menggunakan *QoS* serta *failover* dapat digunakan untuk menilai kestabilan layanan berdasarkan *throughput, delay, jitter, packet loss,* dan ketersediaan koneksi [21], [22]. Pengujian performa *database* pada lingkungan *cloud* juga menggunakan metrik kinerja untuk membandingkan karakteristik beberapa sistem basis data [23].

3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan eksperimen melalui simulasi *virtual*. Metode pengembangan yang digunakan adalah *Network Development Life Cycle* yang dibatasi pada tahap *analysis, design, simulation prototyping, implementation,* dan *monitoring.* Objek pengujian terdiri dari dua arsitektur, yaitu *InnoDB Cluster* sebagai *baseline* dan *Galera Cluster* sebagai sistem usulan. Seluruh pengujian dilakukan pada lingkungan *virtual* menggunakan tiga *node Ubuntu Server* dengan spesifikasi yang dibuat sama agar perbandingan tidak dipengaruhi oleh perbedaan sumber daya perangkat.

Alur penelitian disusun mengikuti pendekatan *Network Development Life Cycle (NDLC)* agar proses analisis kebutuhan, perancangan, pembangunan prototipe, implementasi, dan monitoring dapat dilakukan secara runtut. Pada artikel ini, tahapan NDLC digunakan sampai tahap *monitoring* seperti ditunjukkan pada Gambar 1.

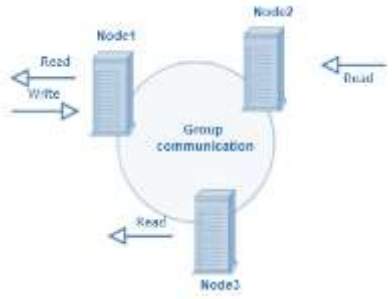
Gambar 1 Tahapan NDLC yang digunakan dalam penelitian

3.1 Analysis

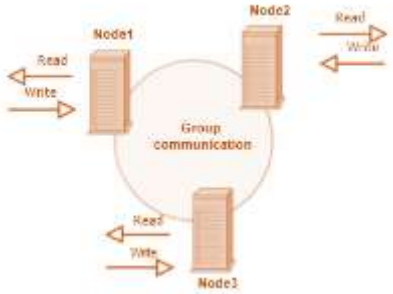
Tahap *analysis* dilakukan melalui observasi terhadap kebutuhan sistem basis data di *UPA TIK* Universitas Pendidikan Ganesha serta studi literatur mengenai replikasi basis data, *high availability*, *InnoDB Cluster*, dan *Galera Cluster*. Hasil analisis menunjukkan bahwa sistem pembanding menggunakan pendekatan *single-primary*, sehingga operasi tulis masih bergantung pada satu *node* utama. Kondisi tersebut menjadi dasar pemilihan *Galera Cluster* sebagai rancangan pembanding karena mendukung replikasi sinkron dan *multi-master*.

3.2 Design

Tahap *design* dilakukan dengan menyusun rancangan dua lingkungan pengujian yang setara, yaitu *InnoDB Cluster* sebagai *baseline* dan *Galera Cluster* sebagai sistem usulan. Keduanya menggunakan tiga *node virtual* dengan spesifikasi yang sama agar hasil pengujian tidak dipengaruhi oleh perbedaan sumber daya perangkat. Perbedaan utama diletakkan pada mekanisme *cluster*, yaitu *single-primary* pada *InnoDB Cluster* dan *synchronous multi-master replication* pada *Galera Cluster*. Rancangan topologi *InnoDB Cluster* sebagai sistem *baseline* ditunjukkan pada Gambar 2, sedangkan rancangan topologi *Galera Cluster* sebagai sistem usulan ditunjukkan pada Gambar 3.



Gambar 2 Topologi InnoDB Cluster sebagai sistem baseline



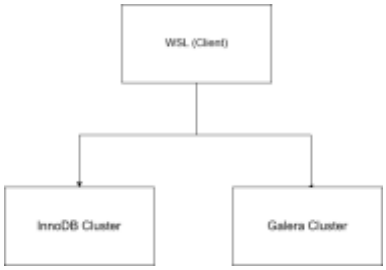
Gambar 3 Topologi Galera Cluster sebagai sistem usulan

3.3 Simulation Prototyping

Tahap *simulation prototyping* dilakukan dengan membangun lingkungan uji menggunakan *VirtualBox*. Setiap *node* menggunakan *Ubuntu Server* dan diberikan alamat *IP* statis dalam jaringan *host-only*. Pada tahap ini, *InnoDB Cluster* dibangun terlebih dahulu sebagai *baseline*, kemudian *Galera Cluster* dibangun sebagai rancangan sistem dengan konfigurasi replikasi sinkron antar-*node*. Pengujian konektivitas dilakukan untuk memastikan seluruh *node* dapat berkomunikasi sebelum masuk ke tahap implementasi.

3.4 Implementation

Tahap implementation difokuskan pada pelaksanaan pengujian. Beban transaksi tidak dijalankan dari *node cluster*, tetapi dari *Windows Subsystem for Linux (WSL)* sebagai client eksternal. Pendekatan ini dipilih agar simulasi mendekati kondisi nyata, yaitu *database cluster* menerima permintaan dari client atau aplikasi di luar *node database*. *Sysbench* digunakan untuk menghasilkan *workload OLTP read-write* pada kondisi tiga *node* aktif dan kondisi *degraded* dua *node* aktif. Arsitektur pengujian menggunakan WSL sebagai *client eksternal* ditunjukkan pada Gambar 4.



Gambar 4 Arsitektur pengujian menggunakan WSL sebagai client eksternal

3.5 Monitoring

Tahap monitoring dilakukan dengan mengumpulkan hasil pengujian dari *Sysbench*, pengamatan *resource* melalui *Grafana* atau Prometheus, dan *script failover*. Data yang diperoleh kemudian direkap berdasarkan metrik *throughput, latency,* penggunaan *CPU*, penggunaan memori, waktu replikasi, integritas data, *RTO*, dan *MTTR*. Konfigurasi *IP node* pengujian ditunjukkan pada Tabel 1, sedangkan spesifikasi *node* pengujian disajikan pada Tabel 2.

Tabel 1. Konfigurasi IP node pengujian

Node	*IP Address*	Keterangan
Node 1	192.168.56.10	*Database Server 1*
Node 2	192.168.56.11	*Database Server 2*
Node 3	192.168.56.12	*Database Server 3*

Tabel 2. Spesifikasi node pengujian

Node	*CPU*	*RAM*	*Storage*
Node 1	*2 Core*	4 GB	20 GB
Node 2	*2 Core*	4 GB	20 GB
Node 3	*2 Core*	4 GB	20 GB

Pengujian dilakukan menggunakan *Windows Subsystem for Linux* sebagai client eksternal. *Sysbench* digunakan untuk menghasilkan *workload OLTP read-write*. Pengujian dilakukan pada dua kondisi, yaitu kondisi tiga *node* aktif dan kondisi *degraded* dua *node* aktif. Pada kondisi tiga *node* aktif digunakan skala 1000, 5000, dan 10000 data. Pada kondisi *degraded* digunakan skala 700, 3500, dan 7500 data. Setiap skenario dilakukan sebanyak 10

iterasi. Rincian skenario pengujian ditunjukkan pada Tabel 3.

Tabel 3. Skenario pengujian

Kondisi	Skala Data	Iterasi
3 *node* aktif	1000; 5000; 10000	10
2 *node* aktif	700; 3500; 7500	10
Failover	*RTO* dan *MTTR*	5

Metrik performa yang diamati meliputi *throughput*, *latency*, penggunaan *CPU*, penggunaan memori, waktu replikasi, dan integritas data. Metrik *high availability* meliputi *Recovery Time Objective* (*RTO*) dan *Mean Time To Recovery* (*MTTR*). Analisis data dilakukan secara deskriptif dengan menghitung nilai rata-rata setiap metrik pada masing-masing *cluster*.

4. HASIL DAN PEMBAHASAN

4.1 Hasil Implementasi

Lingkungan simulasi berhasil dibangun menggunakan tiga *node database* untuk masing-masing arsitektur. Pada *InnoDB Cluster*, *node* bekerja dengan pendekatan *single-primary*, yaitu satu *node* menangani operasi tulis dan *node* lainnya berperan sebagai *secondary*. Pada *Galera Cluster*, seluruh *node* dapat berperan sebagai *master* sehingga transaksi dapat diterima oleh *node* aktif mana pun. Keduanya berhasil berjalan dalam jaringan lokal yang sama dan dapat diakses dari client eksternal melalui *WSL*. Pengujian awal menunjukkan bahwa seluruh *node* dapat saling berkomunikasi dan layanan *database* aktif. Pada *Galera Cluster*, pembuatan *database* pada salah satu *node* dapat terlihat pada *node* lain, yang menunjukkan bahwa replikasi sinkron berjalan. Pada *InnoDB Cluster*, replikasi juga berhasil dilakukan dari *primary node* menuju *secondary node*. Hasil ini menjadi dasar untuk melanjutkan pengujian performa dan *high availability*. Gambar 5 menunjukkan status *node* pada lingkungan *cluster*.

Gambar 5 Status node pada lingkungan cluster

Gambar 5 memperlihatkan status *cluster* pada salah satu proses pengujian. Pemeriksaan status dilakukan untuk memastikan *node* berada pada kondisi aktif dan siap menerima beban transaksi sebelum skenario pengujian dijalankan.

4.2 Rekapitulasi Throughput dan Latency

Rekapitulasi rata-rata *throughput* dan *latency* pada setiap skenario pengujian ditunjukkan pada Tabel 4.

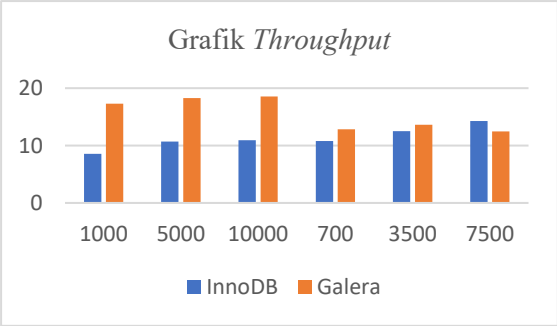
Tabel 4. Rekapitulasi rata-rata throughput dan latency

Kond.	Data	*Cluster*	*TPS*	Latency
3 Node	1000	*InnoDB*	8.54	4936.67
3 Node	1000	*Galera*	17.30	1955.70
3 Node	5000	*InnoDB*	10.67	6354.42
3 Node	5000	*Galera*	18.26	2100.59
3 Node	10000	*InnoDB*	10.91	5764.16
3 Node	10000	*Galera*	18.56	1965.67
2 Node	700	*InnoDB*	10.78	3771.96
2 Node	700	*Galera*	12.83	2600.92
2 Node	3500	*InnoDB*	12.49	5141.16
2 Node	3500	*Galera*	13.61	2142.29
2 Node	7500	*InnoDB*	14.28	3463.92
2 Node	7500	*Galera*	12.46	2971.00

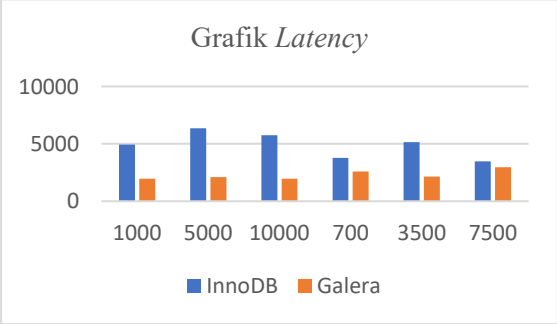
Berdasarkan Tabel 4, *Galera Cluster* menghasilkan *throughput* lebih tinggi pada seluruh skenario tiga *node* aktif. Pada skala 1000 data, rata-rata *throughput Galera* mencapai 17,30 *TPS*, sedangkan *InnoDB Cluster* 8,54 *TPS*. Pada skala 5000 data, *Galera* mencapai 18,26 *TPS*, sedangkan *InnoDB* 10,67 *TPS*. Pada skala 10000 data, *Galera* mencapai 18,56 *TPS*, sedangkan *InnoDB* 10,91 *TPS*. Hal ini menunjukkan bahwa mekanisme *multi-master* pada *Galera* membantu meningkatkan

kemampuan pemrosesan transaksi pada kondisi normal.

Dari sisi *latency*, *Galera Cluster* juga menghasilkan nilai lebih rendah pada seluruh skenario tiga *node* aktif. Rata-rata *latency Galera* berada pada kisaran 1955,70 ms sampai 2100,59 ms, sedangkan *InnoDB* berada pada kisaran 4936,67 ms sampai 6354,42 ms. Pada kondisi *degraded* dua *node* aktif, *Galera* tetap menghasilkan *latency* lebih rendah pada skala 700 dan 3500 data, sedangkan *InnoDB* memiliki *latency* lebih rendah pada skala 7500 data. Secara umum, *Galera* lebih unggul pada metrik *throughput* dan *latency*. Perbandingan *throughput* rata-rata ditunjukkan pada Gambar 6, sedangkan perbandingan *latency* rata-rata ditunjukkan pada Gambar 7.



Gambar 6 Grafik perbandingan throughput rata-rata



Gambar 7 Grafik perbandingan latency rata-rata

Pada Gambar 6 terlihat bahwa *Galera Cluster* menghasilkan *throughput* lebih tinggi pada seluruh skenario tiga *node* aktif dan masih kompetitif pada kondisi *degraded*. Sementara itu, Gambar 7 menunjukkan bahwa *Galera Cluster* memberikan *latency* yang lebih rendah dibandingkan *InnoDB Cluster* pada seluruh skenario, sehingga respons sistem lebih cepat ketika menerima beban transaksi.

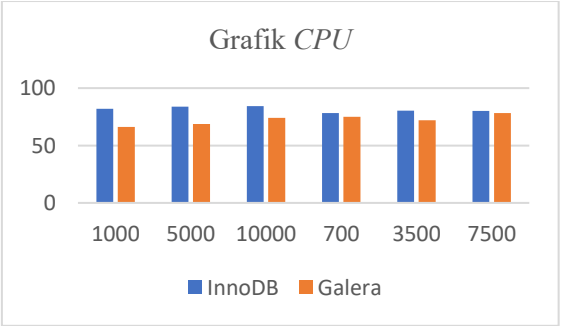
4.3 Rekapitulasi CPU dan Memori

Rekapitulasi rata-rata penggunaan *CPU* dan memori pada setiap skenario pengujian ditunjukkan pada Tabel 5.

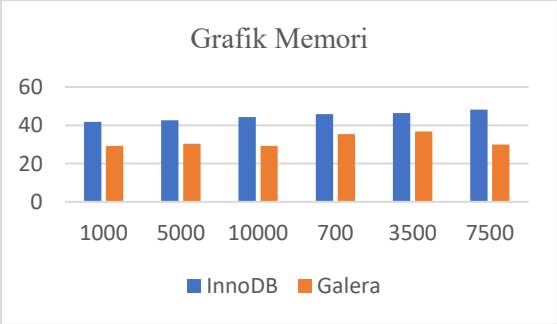
Tabel 5. Rekapitulasi rata-rata penggunaan CPU dan memori

Kond.	Data	*Cluster*	*CPU*	*Memory*
3 Node	1000	*InnoDB*	82.10	41.80
3 Node	1000	*Galera*	66.25	29.28
3 Node	5000	*InnoDB*	83.82	42.68
3 Node	5000	*Galera*	68.85	30.35
3 Node	10000	*InnoDB*	84.39	44.33
3 Node	10000	*Galera*	74.06	29.30
2 Node	700	*InnoDB*	78.30	45.80
2 Node	700	*Galera*	75.01	35.44
2 Node	3500	*InnoDB*	80.44	46.38
2 Node	3500	*Galera*	71.98	36.78
2 Node	7500	*InnoDB*	80.15	48.27
2 Node	7500	*Galera*	78.36	29.95

Berdasarkan Tabel 5, *Galera Cluster* menggunakan *CPU* lebih rendah pada skenario tiga *node* aktif dengan skala 1000 dan 5000 data, sedangkan pada skala 10000 data penggunaan *CPU Galera* tetap lebih rendah dibandingkan *InnoDB*. Pada kondisi *degraded*, penggunaan *CPU Galera* sedikit lebih tinggi pada skala 700 data, tetapi lebih rendah pada skala 3500 dan 7500 data. Dari sisi memori, *Galera Cluster* menunjukkan penggunaan memori yang lebih efisien pada seluruh skenario. Rata-rata penggunaan memori *Galera* berada pada kisaran 29,28% sampai 36,78%, sedangkan *InnoDB* berada pada kisaran 41,80% sampai 48,27%. Perbandingan penggunaan *CPU* rata-rata ditunjukkan pada Gambar 8, sedangkan perbandingan penggunaan memori rata-rata ditunjukkan pada Gambar 9.



Gambar 8 Grafik perbandingan penggunaan CPU rata-rata



Gambar 9 Grafik penggunaan Memori rata-rata

Gambar 8 dan Gambar 9 memperlihatkan bahwa *Galera Cluster* secara umum menggunakan sumber daya yang lebih efisien, terutama pada penggunaan memori yang konsisten lebih rendah dibandingkan *InnoDB Cluster*. Pada penggunaan *CPU*, *Galera* juga lebih rendah pada sebagian besar skenario, meskipun pada kondisi *degraded* selisih keduanya tidak terlalu besar.

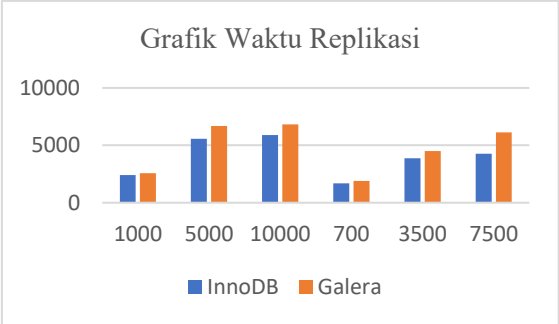
4.4 Rekapitulasi Waktu Replikasi dan Integritas Data

Rekapitulasi rata-rata waktu replikasi dan integritas data pada setiap skenario pengujian ditunjukkan pada Tabel 6.

Tabel 6. Rekapitulasi rata-rata waktu replikasi dan integritas data

Kondisi	Data	*Cluster*	Replikasi	Integritas
3 Node	1000	*InnoDB*	2396.50	Sesuai
3 Node	1000	*Galera*	2578.90	Sesuai
3 Node	5000	*InnoDB*	5573.00	Sesuai
3 Node	5000	*Galera*	6690.30	Sesuai
3 Node	10000	*InnoDB*	5888.00	Sesuai
3 Node	10000	*Galera*	6813.00	Sesuai
2 Node	700	*InnoDB*	1675.90	Sesuai
2 Node	700	*Galera*	1902.70	Sesuai
2 Node	3500	*InnoDB*	3865.20	Sesuai
2 Node	3500	*Galera*	4495.20	Sesuai
2 Node	7500	*InnoDB*	4264.80	Sesuai
2 Node	7500	*Galera*	6127.30	Sesuai

Tabel 6 menunjukkan bahwa hasil integritas data pada seluruh skenario berada dalam kondisi sesuai. Artinya, jumlah data dan hasil verifikasi antar *node* menunjukkan bahwa data berhasil direplikasi tanpa perbedaan jumlah akhir. Pada metrik waktu replikasi, hasil pengujian menunjukkan karakteristik yang berbeda. *InnoDB* lebih cepat pada skala 5000 dan 10000 data dalam kondisi tiga *node* aktif, sedangkan *Galera* lebih cepat pada skala 700 data dalam kondisi *degraded*. Perbedaan ini menunjukkan bahwa replikasi sinkron *Galera* memiliki konsekuensi waktu tertentu karena transaksi perlu dikonfirmasi antar *node*, tetapi memberikan keuntungan pada konsistensi dan kemampuan *multi-master*. Perbandingan waktu replikasi rata-rata ditunjukkan pada Gambar 10.



Gambar 10 Grafik perbandingan waktu replikasi rata-rata

Berdasarkan Gambar 10, waktu replikasi rata-rata *InnoDB Cluster* cenderung lebih rendah pada seluruh skenario pengujian. Hal ini menunjukkan bahwa pada lingkungan simulasi ini *InnoDB* menyelesaikan propagasi data lebih cepat. Meskipun demikian, kedua *cluster* tetap menunjukkan integritas data yang sesuai pada seluruh skenario, sehingga konsistensi data tetap terjaga.

4.5 Rekapitulasi High Availability

Rekapitulasi rata-rata *RTO* dan *MTTR* pada pengujian *high availability* ditunjukkan pada Tabel 7.

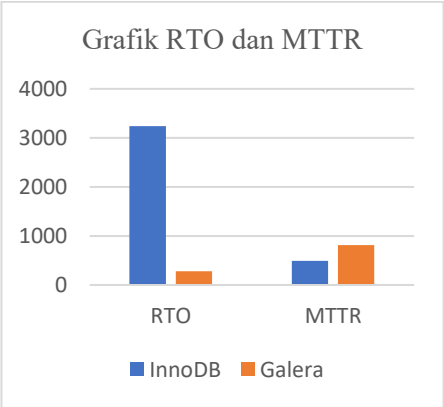
Tabel 7. Rekapitulasi rata-rata RTO dan MTTR

Cluster	*RTO* (ms)	*MTTR* (ms)
InnoDB	3239.60	280.80
Galera	495.00	815.20

Berdasarkan Tabel 7, *Galera Cluster* memiliki *RTO* rata-rata 495,00 ms, jauh lebih rendah dibandingkan *InnoDB Cluster* dengan *RTO*

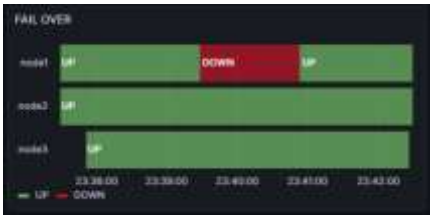
rata-rata 3239,60 ms. Hasil ini menunjukkan bahwa *Galera* lebih cepat mempertahankan ketersediaan layanan ketika salah satu *node* dinonaktifkan. Hal ini disebabkan oleh arsitektur *multi-master Galera* yang tidak bergantung pada satu *primary node*. Selama *cluster* masih memenuhi quorum, *node* lain tetap dapat melayani transaksi.

Pada metrik *MTTR*, *InnoDB Cluster* menunjukkan waktu *recovery* lebih rendah, yaitu 280,80 ms, sedangkan *Galera Cluster* 815,20 ms. Hal ini menunjukkan bahwa proses *node* kembali bergabung ke *InnoDB Cluster* dalam pengujian ini lebih cepat. Namun, dari sisi ketersediaan layanan ketika gangguan terjadi, *Galera* lebih unggul karena waktu *RTO* lebih kecil. Dengan demikian, *Galera Cluster* lebih baik untuk menjaga kontinuitas layanan, sedangkan *InnoDB Cluster* lebih cepat pada pemulihan *node* setelah *node* kembali aktif. Perbandingan rata-rata *RTO* dan *MTTR* ditunjukkan pada Gambar 11.



Gambar 11 Grafik perbandingan rata-rata RTO dan MTTR

Gambar 11 menunjukkan bahwa *Galera Cluster* memiliki nilai *RTO* yang jauh lebih rendah dibandingkan *InnoDB Cluster*, sehingga layanan lebih cepat tetap tersedia ketika salah satu *node* mengalami gangguan. Namun, nilai *MTTR Galera* lebih tinggi, yang menunjukkan bahwa proses *node* kembali sinkron ke *cluster* membutuhkan waktu lebih lama dibandingkan *InnoDB Cluster*. *Monitoring failover* melalui *dashboard Grafana* ditunjukkan pada Gambar 12.



Gambar 12 Monitoring failover pada Grafana

Gambar 12 menunjukkan *dashboard monitoring* yang digunakan untuk melihat perubahan status *node* ketika salah satu *node* dinonaktifkan. Informasi tersebut mendukung pencatatan *RTO* dan *MTTR* karena memperlihatkan kapan *node* mengalami gangguan, kapan layanan tetap tersedia, dan kapan *node* kembali aktif.

4.6 Pembahasan Perbandingan Metrik

Secara keseluruhan, *Galera Cluster* menunjukkan keunggulan pada *throughput*, *latency*, penggunaan memori, dan *RTO*. Keunggulan *throughput* dan *latency* menunjukkan bahwa *Galera* lebih responsif dalam menangani beban transaksi pada sebagian besar skenario. Penggunaan memori yang lebih rendah juga menunjukkan bahwa *Galera* lebih efisien dalam memanfaatkan sumber daya pada lingkungan simulasi. Selain itu, *RTO* yang lebih kecil menunjukkan kemampuan *Galera* dalam menjaga layanan tetap tersedia ketika terjadi gangguan *node*.

InnoDB Cluster tetap memiliki keunggulan pada beberapa aspek, terutama *MTTR* dan sebagian waktu replikasi pada skenario tertentu. *MTTR* yang lebih kecil menunjukkan bahwa *node InnoDB* dapat kembali bergabung lebih cepat setelah proses *recovery*. Namun, karena *InnoDB Cluster* menggunakan pendekatan *single-primary*, kegagalan *primary node* tetap membutuhkan proses pemilihan *primary* baru sebelum layanan tulis dapat kembali stabil. Oleh karena itu, pemilihan arsitektur sebaiknya disesuaikan dengan prioritas sistem. Jika prioritas utama adalah kontinuitas layanan dan konsistensi data, *Galera Cluster* lebih sesuai. Jika prioritasnya pemulihan *node* secara cepat pada arsitektur *single-primary*, *InnoDB Cluster* masih dapat dipertimbangkan.

5. KESIMPULAN

a. Implementasi *Galera Cluster* berhasil disimulasikan menggunakan tiga *node virtual* dan dapat menjalankan mekanisme replikasi sinkron *multi-master*. *InnoDB Cluster* juga berhasil dibangun sebagai sistem *baseline* dengan pendekatan *single-primary*. Kedua sistem dapat digunakan untuk pengujian sinkronisasi, performa, dan *high availability*.
b. Berdasarkan rekapitulasi rata-rata metrik performa, *Galera Cluster* lebih unggul pada *throughput*, *latency*, penggunaan memori, dan *RTO*. *Galera* mampu menghasilkan *throughput* lebih tinggi dan *latency* lebih rendah pada sebagian besar skenario pengujian. *Galera* juga memiliki *RTO* rata-rata lebih kecil, yaitu 495,00 ms, dibandingkan *InnoDB Cluster* sebesar 3239,60 ms.
c. *InnoDB Cluster* memiliki keunggulan pada metrik *MTTR* dengan rata-rata 280,80 ms, sedangkan *Galera Cluster* memiliki rata-rata *MTTR* 815,20 ms. Hal ini menunjukkan bahwa pemulihan *node InnoDB* pada pengujian ini lebih cepat, tetapi *Galera* lebih baik dalam menjaga layanan tetap tersedia ketika terjadi gangguan *node*.
d. Secara umum, *Galera Cluster* lebih sesuai digunakan sebagai alternatif sistem basis data terdistribusi yang membutuhkan replikasi sinkron, konsistensi data, dan ketersediaan layanan tinggi. Pengembangan penelitian selanjutnya dapat dilakukan pada *server* fisik, jumlah *node* yang lebih banyak, variasi *workload* yang lebih luas, serta analisis statistik lanjutan setelah data pengujian disetujui.

UCAPAN TERIMA KASIH

Selama proses penelitian berlangsung, banyak pihak yang telah memberikan dukungan, semangat, dan motivasi kepada penulis. Oleh karena itu, penulis mengucapkan terima kasih yang sebesar-besarnya karena tanpa kontribusi tersebut penelitian ini tidak akan dapat diselesaikan hingga tahap akhir.

DAFTAR PUSTAKA

[1] Y. Zhang et al., "Towards cost-effective and elastic cloud *database* deployment via memory disaggregation," Proceedings of the VLDB Endowment, pp. 1900-1912, 2021, doi: 10.14778/3467861.3467877.
[2] S. Shidqi, D. A. Sulistyo, and F. A. Ahda, "Pembuatan Infrastruktur Database Menggunakan Metode Replikasi Untuk Pelanggan Jagoan Hosting," Jurnal Ilmiah Teknologi Informasi Asia, vol. 16, no. 1, 2022.
[3] R. Baharsyah, "Perbandingan Load Balancing Router MySQL dan HAProxy Menggunakan SysBench dan *Cluster* InnoDB Pada Sistem Operasi CentOS," 2025.
[4] R. Shrestha and T. Tandel, "An Evaluation Method and Comparison of Modern *Cluster*-Based Highly Available Database Solutions," in CLOSER, pp. 131-138, 2023.
[5] S. Widiono, "Experiments and Descriptive Analysis in The MariaDB Database *Cluster* System to Prepare Data Availability," International Journal of Engineering Technology and Natural Sciences, vol. 1, no. 1, pp. 42-48, 2019, doi: 10.46923/ijets.v1i1.24.
[6] A. Heryanto and Y. Hartati, "Backup Database Dengan Multi Master Replikasi Pada Kluster Server," Jurnal Ilmiah Ilmu Komputer, vol. 6, no. 1, pp. 7-12, 2022, doi: 10.35329/jiik.v6i1.118.
[7] P. Mandiasa, I. N. S. W. Wijaya, and I. K. R. Arthana, "Perancangan Sistem Early Warning Kebakaran Berbasis IoT pada Data Center *UPA TIK* Undiksha," Jurnal Komputer Teknologi Informasi Sistem Komputer (JUKTISI), vol. 4, no. 3, pp. 2383-2394, 2026, doi: 10.62712/juktisi.v4i3.888.
[8] M. W. Aryadi, I. K. R. Arthana, and P. H. Suputra, "Smart Lock System Berbasis IoT Menggunakan ESP32 untuk Keamanan Akses Pusat Data (Studi Kasus: *UPA TIK* Undiksha)," Jurnal Informatika dan Teknik Elektro Terapan, vol. 14, no. 1, 2026, doi: 10.23960/jitet.v14i1.8826.
[9] D. Muliawan, "Penerapan replikasi synchronous pada aplikasi data KRS online pada Universitas Jabal Ghafur menggunakan PostgreSQL," Future Academia, vol. 1, no. 1, pp. 31-45, 2023.
[10] M. Alham Nushair and K. Yahya, "Penerapan Replikasi Basis Data Terdistribusi Menggunakan Metode Synchronous Pada Favehotel Makassar," Nusantara Hasana Journal, vol. 1, no. 12, 2022.
[11] A. Arnqvist, L. Jiang, and C. Klein, "Evaluating Failover and Recovery of Replicated SQL Databases," 2023.

[12] D. R. Zmaranda, C. I. Moisi, C. A. Gyorodi, R. Gyorodi, and L. Bandici, "An analysis of the performance and configuration features of MySQL Document Store and Elasticsearch as an alternative backend in a data replication solution," Applied Sciences, vol. 11, no. 24, 2021, doi: 10.3390/app112411590.

[13] A. Setiawan and W. M. Kansha, "Pembuatan Sistem Database *Cluster* Menggunakan Aplikasi *Galera Cluster* di Sekolah Vokasi IPB University," Jurnal Sains Terapan, vol. 11, no. 2, pp. 49-59, 2021, doi: 10.29244/jstsv.11.2.49-59.

[14] Z. Li, Y. Tu, and Z. Ma, "A Sample-Aware Database Tuning System With Deep Reinforcement Learning," Journal of Database Management, vol. 35, no. 1, 2023, doi: 10.4018/JDM.333519.

[15] J. Redzepagic, A. Kapulica, N. Malesevic, and V. Dakic, "Empirical Performance and Operational Analysis of Monolithic and Distributed Database Architectures in Kubernetes Environments," Computers, vol. 15, no. 5, 2026, doi: 10.3390/computers15050282.

[16] N. Khairunisa, Carudin, and A. Jamaludin, "Analisis Perbandingan Algoritma CNN dan YOLO dalam Mengidentifikasi Kerusakan Jalan," Jurnal Informatika dan Teknik Elektro Terapan, vol. 12, no. 3, pp. 1756-1768, 2024, doi: 10.23960/jitet.v12i3.4434.

[17] K. S. Pradipta, G. A. J. Saskara, and B. G. K. Yudistira, "Implementasi Kubernetes *Cluster* untuk High Availability dan Load Balancing SIAK Undiksha," RIGGS: Journal of Artificial Intelligence and Digital Business, vol. 4, no. 4, pp. 12831-12840, 2026, doi: 10.31004/riggs.v4i4.5846.

[18] G. A. J. Saskara, I. M. E. Listartha, and I. K. R. Arthana, "Implementasi Proxmox Untuk High Availability Dan Load Balancing Pada Sistem SIAK Undiksha," Algoritme Jurnal Mahasiswa Teknik Informatika, vol. 6, no. 1, pp. 323-334, 2025, doi: 10.35957/algoritme.v6i1.11313.

[19] I. K. D. A. S. Nanda, K. A. Seputra, and K. T. Dermawan, "Perbandingan Performa RESTful API Menggunakan Framework Python Flask dan Go Gin pada Pengembangan Aplikasi Alumni Circle," RIGGS: Journal of Artificial Intelligence and Digital Business, vol. 5, no. 1, pp. 12113-12119, 2026, doi: 10.31004/riggs.v5i1.7664.

[20] D. M. W. Dwipayoga, G. A. J. Saskara, and I. M. G. Sunarya, "Pengembangan Website Network Automation untuk Manajemen Bandwidth Mikrotik dengan Metode Hierarchical Token Bucket (HTB) di SMP Negeri 8 Singaraja," KARMAPATI, vol. 14, no. 2, 2025, doi: 10.23887/karmapati.v14i2.94491.

[21] K. D. K. Prayudi, G. A. J. Saskara, and I. M. G. Sunarya, "Evaluasi Kualitas Jaringan Internet di SMP Negeri 3 Singaraja Menggunakan Metode Quality of Service (QoS)," KARMAPATI, vol. 14, no. 2, pp. 121-134, 2025, doi: 10.23887/karmapati.v14i2.96208.

[22] J. E. Neno, G. A. J. Saskara, and I. M. G. Sunarya, "Optimalisasi Penggunaan 2 ISP Menggunakan Metode PCC Serta Failover di SMP Negeri 4 Singaraja," KARMAPATI, vol. 14, no. 2, pp. 109-120, 2025, doi: 10.23887/karmapati.v14i2.94941.

[23] M. Zboril and V. Svata, "Performance Comparison of Cloud Databases," Procedia Computer Science, pp. 388-395, 2025, doi: 10.1016/j.procs.2025.02.134.