Teknika 19 (1): 47-58

JURNAL TEKNIKA
ISSN: 0854-3143
e-ISSN: 2622-3481
Journal homepage: http://jurnal.polsri.ac.id/index.php/teknika
Journal Email: teknika@polsri.ac.id

Implementasi *Master-Slave* Sebagai Koneksi Baca Dan Tulis Database Dengan Metode *Replication*

Akhlak Kamiswara*[1], Alek Wijaya[2]

[*1, 2]Teknik Informatika, Universitas Bina Darma, Palembang, Indonesia
*Email Penulis Korespondensi: akamiswara@gmail.com

Abstrak

Kemajuan dalam komputasi, internet of things, data yang dihasilkan mesin, volume data kini meledak. Memiliki data saja tidak cukup. Seseorang perlu memiliki kemampuan untuk melakukan manajemen database melibatkan keterampilan dan pengetahuan yang diperlukan untuk merancang, mengimplementasikan, dan memelihara sistem database. Bagaimana penerapan database dengan metode master-slave replication sebagai koneksi tulis dan baca sehingga dapat membantu untuk memisahkan proses koneksi saat melakukan penulisan dan pembacaan data sebagai pembagi beban kerja server dan meningkatkan execute time pada client. Sehingga dapat mengetahui hasil dari pengujian metode master-slave replication sebagai koneksi baca dan tulis database dan membandingkan hasil tersebut dengan metode master only sebagai koneksi baca dan tulis database. Pada bentuk master-slave penulis membuat master sebagai server yang aktif melakukan segala bentuk penulisan data metode HTTP POST dan slave yang standby akan mendapatkan data yang didapatkan hasil dari replikasi master digunakan untuk proses pembacaan data metode HTTP GET ini membuat proses penulisan data kehadiran tidak akan terganggu oleh proses pembacaan data ke database sehingga CPU Usage dan Memory Usage server akan terjaga dan meningkatkan execution time dan total request. Dengan adanya metode master-slave replication ini akan membantu mengurangi persaingan untuk resource dan memungkinkan aplikasi untuk melakukan lebih banyak operasi baca dan tulis secara paralel.

Kata kunci*—Database, MariaDB, Master-Slave Replication, Execution Time, Total Request*

Abstract

The rapid advancements in computing, IoT, machine-generated data, and overall data volumes have created an unprecedented demand for efficient data management. Merely possessing data is insufficient; the capability to effectively manage databases, encompassing skills and knowledge in design, implementation, and maintenance, is crucial. This study investigates how implementing a master-slave replication method as a write and read connection can enhance database performance. By separating write and read operations, this method aims to balance the workload, reduce server load, and improve client-side execution time. The research compares the performance of the master-slave replication method with a traditional master-only approach. In the master-slave setup, the master server handles all write operations using HTTP POST, while the slave server, which receives replicated data from the master, manages read operations using HTTP GET. This separation ensures that write operations are not hindered by

Online Available: September 29, 2024

read operations, optimizing CPU and memory usage, and consequently improving execution time and overall request handling. Ultimately, master-slave replication mitigates resource contention, enabling applications to execute more read and write operations concurrently.

Keywords—*Database, MariaDB, Master-Slave Replication, Execution Time, Total Request*

1. PENDAHULUAN

Data dalam istilah *the next new oil* merupakan sebuah aset yang berharga. Ibarat sumber daya minyak, bagi mereka yang melihat nilai fundamental data dan belajar mengekstrak serta menggunakannya ada imbalan yang sangat besar. Kita berada dalam ekonomi digital di mana data menjadi lebih berharga. Namun jika tidak diolah, maka data tersebut tidak dapat digunakan.

Jadi, data harus dipecah, dianalisis agar memiliki nilai. Karena kemajuan dalam komputasi, *internet of things*, data yang dihasilkan mesin, volume data kini meledak. Memiliki data saja tidak cukup. Seseorang perlu memiliki kemampuan untuk melakukan manajemen *database* melibatkan keterampilan dan pengetahuan yang diperlukan untuk merancang, mengimplementasikan, dan memelihara sistem *database* [1],[2]. Dalam melakukan proses penulisan dan pembacaan data ke *database* diperlukan metode sarana untuk melakukan proses tersebut seperti menggunakan metode protokol *HTTP POST* untuk mengirimkan data dan metode *protocol HTTP GET* untuk membaca data antara *client* dan *server* [3].

Master-slave replication merupakan sebuah metode penduplikasi basis data dimana jika terdapat data yang ditulis ke *master database*, data tersebut juga akan masuk ke *database slave* yang *standby*. Pada bentuk *master-slave* penulis membuat *master* sebagai *server* yang aktif dan akan melayani proses penulisan data sedangkan *slave* yang *standby* akan mendapatkan data dari replikasi *master* yang digunakan untuk melayani proses pembacaan data. Ini yang membuat proses penulisan data kehadiran tidak akan terganggu oleh proses pembacaan data ke *database* sehingga dapat meningkatkan *total request* dan *execution time* [4].

Hal ini yang membuat penulis menggunakan teknologi replikasi data dan melakukan pengujian pada *database* yang menggunakan metode *master only* dan *master-slave* lalu membandingkannya. Beberapa penelitian terdahulu yang menerapkan implementasi replikasi pada *database* diantaranya, membuat *auto promote server slave* menjadi *master* di saat *server master* mengalami *error* [5]. Kemudian penelitian [6] melakukan *load testing* untuk mendapatkan perbandingan performa antara arsitektur *monolithic* dan *microservices*. Penelitian lainnya oleh [7] Implementasi *Zabbix* untuk *monitong resource* perangkat komputer dan jaringan. Berdasarkan dari penelitian terdahulu maka dalam penelitian ini akan dilakukan implementasi *master-slave replication* untuk membagi beban proses pembacaan dan penulisan data ke *database*.

2. METODE PENELITIAN

2.1 Tahapan Penelitian

Adapun tahapan penelitian data yang digunakan penulis yaitu:

1) Analisis

Tahap awal ini dilakukan analisa kebutuhan, serta permasalahan yang muncul sehingga dilakukan tahap menganalisa untuk melakukan perancangan.

2) Design

Tahap desain ini akan membuat gambar desain topologi *server database* yang akan dibangun dan perancangan *database*.

3) Implementation

Tahapan ini akan lebih lama dari tahapan sebelumnya, dikarenakan penulis menerapkan yang telah dianalisa dan dirancang sebelumnya. Pada tahapan inilah terlihat bagaimana metode *master-slave replication database* yang akan dibangun akan memberikan perubahan dengan metode yang hanya menggunakan *database master only*.

4) Testing dan Monitoring

Tahapan ini akan dilakukan pengujian sistem berupa uji coba sistem yang telah dibangun, pengujian akan dilakukan pada metode *master-slave replication database* dimana *master* akan digunakan untuk koneksi tulis sedangkan *slave* akan digunakan untuk koneksi baca dan *master database only* akan digunakan untuk koneksi baca dan koneksi tulis. *Database* akan dilakukan *load testing* menggunakan aplikasi *locust* dan aplikasi absensi yang bisa melakukan penulisan data dan menampilkan hasil absensi menggunakan bahasa pemrograman *PHP* [8], [9]. Tahapan ini akan mengamati *database* yang menggunakan *master-slave replication* dan *master only* untuk melihat sumber daya *server database* seperti *CPU usage* dan *memory usage* menggunakan aplikasi *zabbix* dan melihat *response tim*e serta banyaknya *request* menggunakan aplikasi *locust*, sehingga dapat dibandingkan hasil dari implementasi tersebut [10].

2. 2 Analisis

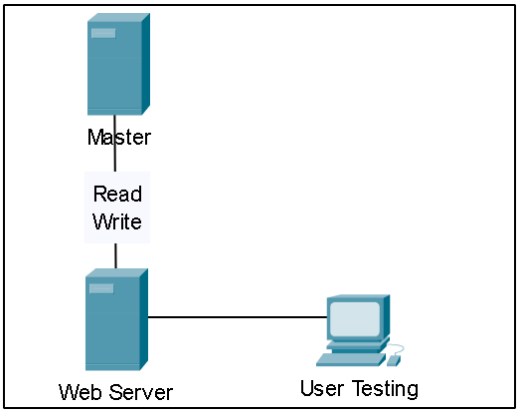
Berdasarkan pengamatan yang didapatkan, proses kegiatan absensi kehadiran biasanya memiliki waktu maksimal mengabsen sehingga absen tepat waktu menjadi sangat penting untuk catatan kehadiran seseorang. Biasanya absensi menggunakan identitas seperti nama dan nomor induk, ketepatan waktu sangat penting sehingga menyebabkan orang-orang melakukan kegiatan absensi kehadiran secara bersamaan dengan masif sehingga mempengaruhi kinerja *database*, yang dapat menyebabkan turunnya performa *database* dan terjadinya kegagalan dalam pencatatan data absensi.

2. 3 Desain

Desain merupakan gambaran dari proses kerja pengujian *database* yang akan dibuat, selain itu juga sebagai gambaran secara menyeluruh terhadap pembuatan untuk pengujian *database master only* dan *master-slave*. Tujuan dari desain ini adalah suatu desain sistem yang dilakukan dalam mengerjakan suatu sistem yang akan dijalankan dan diimplementasikan dalam suatu bentuk konsep awal pengujian *database master only* dan *master-slave*. Dengan adanya desain *database* yang terkonsep maka pengujian *database master only* dan *master-slave* ini akan terarah.

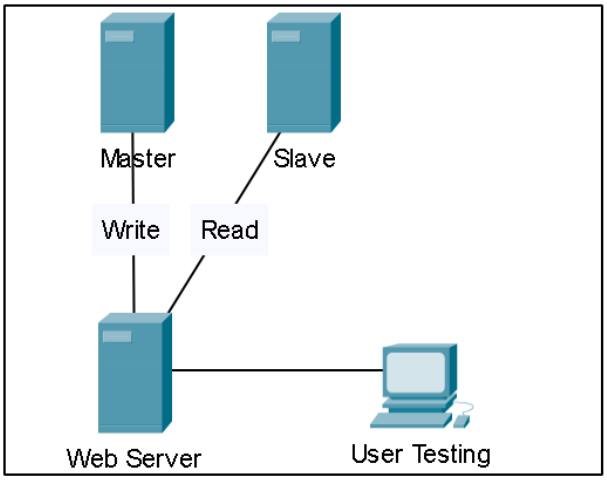
2. 3.1 Desain Topologi Server

Desain topologi mengambarkan bagaimana proses data *get* dan *post* terjadi dengan menggunakan metode *master only* dan *master-slave*, hal ini dapat dilihat pada Gambar 1.



Gambar 1 Topologi *Master Only*

Pada *database master only* proses pembacaan data dan penulisan data terjadi di *database master*, dimana rancangan artsitekturnya dapat dilihat pada Gambar 2. Pada *database master-slave* proses penulisan data terjadi di *database master* dan pembacaan data terjadi di *database slave*.



Gambar 2 Topologi *Master-slave*

2. 3.2 Desain Database Desain

Menurut [11], proses desain *database* terdiri dari 3 tahap yaitu model konseptual, model logikal, dan fisik. Penulis menggunakan 3 tahap tersebut untuk mendesain *database* sesuai dengan kegiatan absensi kehadiran.

a. Model Konseptual

Penulis membuat desain model konseptual *database* berdasarkan kegiatan absensi kehadiran pada umumnya. Desain model konseptual terdiri dari entitas dan relasi. Sebelum menentukan entitas maka diperlukan identifikasi kegiatan absensi kehadiran. Penulis hanya menggunakan 1 *table* untuk pengujian *database* metode *master only* dan *master-slave* sehingga penulis tidak membuat relasi untuk pengujian ini. Entitas dapat dilihat pada Tabel 1.

Tabel 1 Tabel Entitas

Nama Entitas	Keterangan
Kehadiran	Berisi Informasi mengenai data kehadiran

b. Desain Model Logikal

Tahapan ini merupakan tahapan lanjutan setelah desain model konseptual dibuat. Hasil dari identifikasi pada desain model konseptual menghasilkan rincian entitas *domain*. Adapun hasil dari identifikasi tersebut menghasilkan *table* kehadiran dengan rincian entitas dapat dilihat pada Tabel 2.

Tabel 2 Rancangan Tabel Kehadiran

Nama Tabel	Atribut, *Primary Key* (*)
Kehadiran	*Id_kehadiran, badge, finger_node, waktu

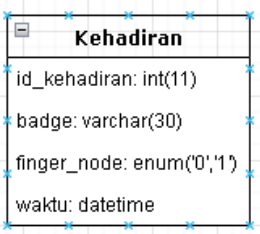
Setelah melakukan proses normalisasi, selanjutnya penulis mengidentifikasi atribut, *domain* dan *primary key* pada entitas tabel kehadiran. Hasil analisis identifikasi atribut, *domain* dan *primary key* pada *table* kehadiran dapat dilihat pada Tabel 3.

Tabel 3 Identifikasi Atribut, Domain dan Primary Key

Entitas	Atribut	*Domain*	*Primary Key*
Kehadiran	id_kehadiran	*Int*	id_kehadiran
	badge	*Varchar*	
	finger_node	*Enum*(‘0’,’1’)	
	waktu	*datetime*	

c. Desain Model Fisik

Tahap akhir dari desain *database* adalah membuat rancangan model fisik yang bertujuan untuk mengambarkan struktur *database*. Desain dapat dilihat pada Gambar 3 model fisik dari *database* untuk *table* kehadiran.



Gambar 3 Model Fisik *Database*

2. 4 Implementation

Untuk melakukan pengujian *database* metode *master only* dan *master-slave* ini penulis menggunakan *operating system VMware vSphere Hypervisor* sebagai *server* virtualisasi dan *Ubuntu 22.04 LTS 64bit* sebagai *server virtual machine*, sedangkan pada sisi *database* penulis menggunakan *Mariadb server* sebagai *database server* [12]. Pengujian *database* ini menggunakan *PHP* sebagai bahasa pemrograman yang digunakan.

2. 4.1 Implementation API

Pada implementasi *API* ini, penulis membuat *script coding* menggunakan bahasa pemrograman *PHP* yang akan dihosting di *web server* menggunakan *Apache2* [13]. *Script* koding *PHP* yang telah dibuat ditempatkan pada *folder /var/www/html* untuk *web server apache2* dapat membaca *script coding* tersebut sehingga dapat di*hosting* [14].

a. Script coding get method

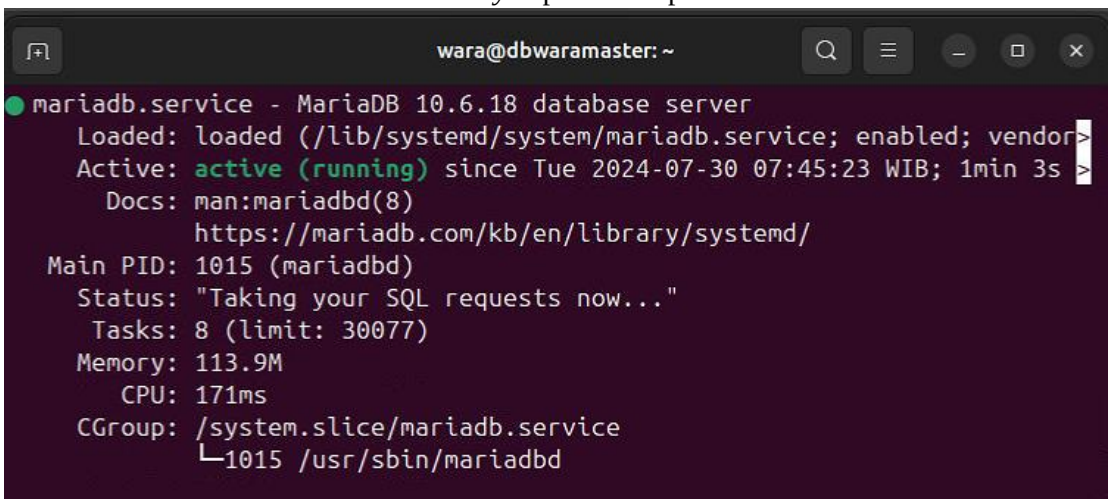
Script ini berfungsi untuk melakukan pembacaan data ke *database* dengan membaca tabel kehadiran dengan *limit* 30 *rows*.

b. Script coding post method

Script ini berfungsi untuk melakukan penulisan data ke *database* dan data masuk ke *table* kehadiran dengan menggunakan *syntax transaction* dan *mysqli_rollback API* ini dapat mengulangi proses penulisan data jika terjadi kegagalan dalam penulisan data ke tabel.

2. 5 Implementasi Master Only

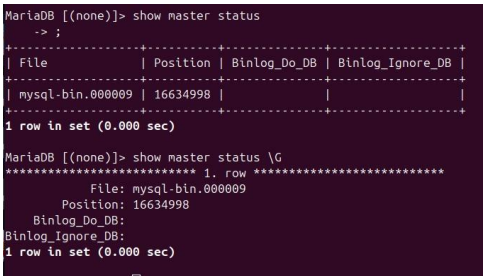
Hasil instalasi *database master only* dapat dilihat pada Gambar 4.



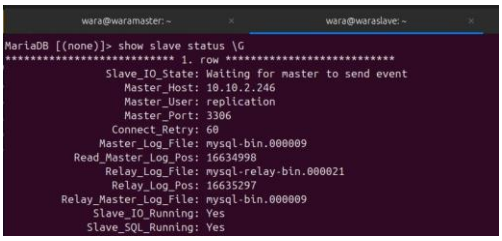
Gambar 4 *Status MariaDB Master Only*

2. 6 Implementasi Master-Slave

Hasil instalasi *database master-slave replication* dapat dilihat pada Gambar 5 dan Gambar 6.



Gambar 5 *Status Database Master*



Gambar 6 *Status Database Slave*

2. 7 Implementasi Locust

Implementasi *locust* menggunakan *script* dan *script* ini berfungsi untuk melakukan pembacaan *API* mana yang akan diuji. *Script coding* koneksi ini diberi nama *locustfile.py*. Setelah dibuat *script*nya dapat dijalankan untuk melakukan *testing*.

3. HASIL DAN PEMBAHASAN

Hasil dari *master-slave replication* membuat *master* sebagai *server* yang aktif melakukan segala bentuk penulisan data metode *HTTP POST* dan *slave* yang *standby* akan mendapatkan data yang didapatkan hasil dari replikasi *master* digunakan untuk proses pembacaan data metode *HTTP GET* ini membuat proses penulisan data kehadiran tidak akan terganggu oleh proses pembacaan data ke *database* sehingga *CPU Usage* dan *Memory Usage server* akan terjaga dan meningkatkan *execution time* dari sisi *client*, sehingga membantu mengurangi persaingan untuk sumber daya dan memungkinkan aplikasi untuk melakukan lebih banyak operasi baca dan tulis secara *parallel* [15], [16].

Pada penelitian ini juga dilakukan perbandingan pengujian menggunakan metode *master-slave replication* sebagai koneksi baca dan tulis *database* dengan metode *master only* sebagai koneksi baca dan tulis *database*.

3. 1 Testing dan Monitoring

Pengujian ini menggunakan 2 juta baris *record* dengan melakukan perbandingan metode pengujian menggunakan metode *master-slave replication* dan metode *master only*. Pengujian ini juga melibatkan aplikasi *Locust* sebagai aplikasi yang digunakan untuk melakukan *load testing* tetapi dengan *API* yang berbeda.

3. 1.1 Testing Master Only

Pengujian *master only* dilakukan dengan menggunakan 2 juta baris *record* pada *table* dengan menggunakan *API* untuk melakukan pengujian.

a. Struktur Tabel Master Only

Struktur *table master only* akan menampilkan *field record table* yang akan digunakan pada *master only*. Struktur *table master only* dapat dilihat pada Gambar 7.

```
MariaDB [absensi]> desc kehadiran;
+-------------+-------------------------+------+-----+-------------------+----------------+
| Field       | Type                    | Null | Key | Default           | Extra          |
+-------------+-------------------------+------+-----+-------------------+----------------+
| id_kehadiran | int(11) unsigned zerofill | NO   | PRI | NULL              | auto_increment |
| badge       | varchar(30)             | NO   |     | NULL              |                |
| finger_node | enum('0','1')           | NO   |     | NULL              |                |
| waktu       | datetime                | NO   |     | current_timestamp() |              |
+-------------+-------------------------+------+-----+-------------------+----------------+
4 rows in set (0.002 sec)

MariaDB [absensi]>
```

Gambar 7 Struktur Tabel

b. *Record Table Master Only*

Record table master only yang digunakan pada *master only* menampilkan data *record* dengan menggunakan 2 Juta baris *record*, seperti dapat dilihat pada Gambar 8.

```
Database changed
MariaDB [absensi]> SELECT COUNT(*) FROM kehadiran;
+----------+
| COUNT(*) |
+----------+
|  2000000 |
+----------+
1 row in set (0.734 sec)

MariaDB [absensi]>
```

Gambar 8 *Record Table Master Only*

c. Pengujian Menggunakan *Locust*

Pengujian dilakukan dengan menggunakan browser dengan *URL localhost*:8089. Pengujian dilakukan menggunakan 512 pengguna selama 30 menit yang akan diarahkan ke *server API master only*.

d. Hasil Pengujian

Hasil Pengujian yang telah dilakukan menampilkan hasil *total request* sebanyak 57517 dengan rata-rata *respon time* 12388,24 ms. Hasil dari pengujian dapat dilihat pada Gambar 9.

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)
GET	/skripsi/get.php	28933	0	4	19237.02	2	1390229
POST	/skripsi/insert_begin.php	28584	0	4	5455.84	2	1175189
	Aggregated	57517	0	4	12388.24	2	1390229

Gambar 9 Hasil Pengujian *Master Only*

e. *CPU Master Only*

Penggunaan *CPU* pada *server master only* selama 30 menit menampilkan rata-rata 98,9462 % dengan maksimal pengunaan 100 % dan minimal penggunaan 68,3853 %. Hasil grafik pengunaan *CPU Usage* dapat dilihat pada Gambar 10.

Gambar 10 Grafik *CPU Usage Master Only*

f. *Memory Master Only*

Penggunaan *memory* pada *server master only* selama 30 menit menampilkan rata-rata 17,3016 % dengan maksimal pengunaan 18,1289 % dan minimal penggunaan 16,352 %. Hasil grafik pengunaan *memory usage* dapat dilihat pada Gambar 11.

Gambar 11 Grafik *Memory Usage Master Only*

3. 1.2 Pengujian Master-Slave Replication

Pengujian *master-slave replication* dilakukan dengan menggunakan 2 juta baris *record* pada *table* dengan menggunakan *API* untuk melakukan pengujian.

a. Struktur Table Master-Slave

Struktur *table master-slave* akan menampilkan *field record table* yang akan digunakan pada *master-slave*, seperti dapat dilihat pada Gambar 12.

```
MariaDB [absensi]> desc kehadiran;
+-------------+-------------------------+------+-----+-------------------+----------------+
| Field       | Type                    | Null | Key | Default           | Extra          |
+-------------+-------------------------+------+-----+-------------------+----------------+
| id_kehadiran | int(11) unsigned zerofill | NO   | PRI | NULL              | auto_increment |
| badge       | varchar(30)             | NO   |     | NULL              |                |
| finger_node | enum('0','1')           | NO   |     | NULL              |                |
| waktu       | datetime                | NO   |     | current_timestamp() |                |
+-------------+-------------------------+------+-----+-------------------+----------------+
4 rows in set (0.002 sec)

MariaDB [absensi]>
```

Gambar 12 Struktur Tabel *Master-slave*

b. Record Table Master-Slave

Record table master-slave yang digunakan pada *master-slave* menampilkan data *record* dengan menggunakan 2 Juta baris *record* seperti dapat dilihat pada Gambar 13.

```
Database changed
MariaDB [absensi]> SELECT COUNT(*) FROM kehadiran;
+----------+
| COUNT(*) |
+----------+
|  2000000 |
+----------+
1 row in set (0.734 sec)

MariaDB [absensi]>
```

Gambar 13 Record *Table Master-Slave*

c. Pengujian Menggunakan Locust

Pengujian dilakukan dengan menggunakan *browser* dengan *URL localhost*:8089. Pengujian dilakukan menggunakan 512 pengguna selama 30 menit yang akan diarahkan ke *server API master-slave*.

d. Hasil Pengujian

Hasil pengujian yang telah dilakukan menampilkan hasil *total request* sebanyak 148576 dengan rata-rata respon *time* 5648,33 ms. Hasil dari pengujian dapat dilihat pada Gambar 14.

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)
GET	/skripsi2/get.php	74459	0	4	8133.14	2	562405
POST	/skripsi2/insert_begin.php	74117	0	6	3152.06	4	497673
	Aggregated	148576	0	5	5648.33	2	562405

Gambar 14 Hasil Pengujian *Master Only*

e. CPU Master metode Master-Slave

Penggunaan *CPU* pada *server master only* selama 30 menit menampilkan rata-rata 1,9373% dengan maksimal pengunaan 2,4285% dan minimal penggunaan 0,1252%. Hasil grafik pengunaan *CPU Usage* dapat dilihat pada Gambar 15.

Gambar 15 Grafik *CPU Usage Master-Slave Node Master*

f. CPU Slave metode Master-Slave

Penggunaan *CPU* pada *server master only* selama 30 menit menampilkan rata-rata 97,0538 % dengan maksimal pengunaan 100 % dan minimal penggunaan 11,6153 %. Hasil grafik pengunaan *CPU Usage* dapat dilihat pada Gambar 16.

Gambar 16 Grafik *CPU Usage Master-Slave Node Slave*

g. Memory Master metode Master-Slave

Penggunaan *memory* pada *server master only* selama 30 menit menampilkan rata-rata 19,1504% dengan maksimal pengunaan 19,2583% dan minimal penggunaan 18,9499%. Hasil grafik pengunaan *memory usage* dapat dilihat pada Gambar 17.

Gambar 17 Grafik *Memory Usage Master-Slave Node Master*

h. Memory Slave Metode Master-Slave

Penggunaan *memory* pada *server master only* selama 30 menit menampilkan rata-rata 21,9014% dengan maksimal pengunaan 22,0583% dan minimal penggunaan 21,8303%. Hasil grafik pengunaan *memory usage* dapat dilihat pada Gambar 18.

Gambar 18 Grafik *Memory Usage Master-Slave Node Slave*

3. 1.3 Perbandingan Master Only dan Master-Slave

Pengujian *master only* dan *master-slave* menghasilkan data yang dapat dibandingkan. Data yang dihasilkan dari pengujian adalah *response time*, *total requests*, *cpu utilization* dan *memory utilization*.

a. Response Time

Perbandingan antara *master only* dan *master-slave* dalam kecepatan melayani *request*. Melihat dari hasil perbandingan pada Tabel 4 didapatkan hasil peningkatan rata-rata nilai *response time* pada saat menggunakan metode *master-slave*, performa kecepatan melayani *request* lebih cepat dibandingkan menggunakan *master only*.

Tabel 4 Perbandingan *Response Time*

Response Time	*Master Only*	*Master-slave*	Peningkatan
Rata-rata *Read*	19237.02 ms	8133.14 ms	236,52 %
Rata-rata *Write*	5455.84 ms	3152.06 ms	173,08 %
Total	12388.24 ms	5648.33 ms	219,32 %

b. Total Request

Perbandingan antara *master only* dan *master-slave* dalam menerima banyaknya *request*. Melihat dari hasil perbandingan pada Tabel 5 didapatkan hasil peningkatan banyaknya *response* yang dilayani pada saat menggunakan metode *master-slave*, performa melayani *request* lebih banyak dibandingkan menggunakan *master only*.

Tabel 5 Perbandingan *Total Requests*

Banyak *Request*	*Master Only*	*Master-slave*	Peningkatan
Read	28933	74459	257,34 %
Write	28584	74117	259,29 %
Total	57517	148576	258,31 %

c. CPU Utilization

Perbandingan antara *master only* dan *master-slave* rata-rata *CPU utilization*. Melihat dari hasil perbandingan pada Tabel 6 didapatkan hasil peningkatan penggunaan *CPU*, saat pengujian pada *master only* terjadi peningkatan *CPU usage* di awal pengujian sebesar 68.88% sedangkan *master-slave* hanya 11.61%. Rata-rata penggunaan *master only* sebesar 98.94% dan *master-slave* sebesar 97.05% terjadi selisih 1.89% pada rata-rata penggunaan *CPU*. Maksimal penggunaan *CPU master only* dan *master-slave* sama-sama 100% ini menandakan bahwa *requests* yang terjadi pada *database* cenderung menggunakan *CPU* untuk melakukan proses penulisan dan pembacaan data.

Tabel 6 Perbandingan *CPU Utilization*

CPU usage	*Master Only*	*Master Master-slave*	*Slave Master-slave*
Min	68.88 %	0.12 %	11.61 %
Max	100 %	2.42%	100 %
Avarage	98.94 %	1.93 %	97.05 %

d. Memory Utilization

Perbandingan antara *master only* dan *master-slave* rata-rata *memory utilization*. Melihat dari hasil perbandingan pada Tabel 7 didapatkan hasil peningkatan penggunaan *memory*. peningkatan penggunaan *memory master only* dan *master-slave* ini menandakan bahwa *requests* yang terjadi pada *database* cenderung menggunakan *memory* lebih sedikit untuk melakukan proses penulisan dan pembacaan data.

Tabel 7 Perbandingan *Memory Utilization*

Memory Usage	*Master Only*	*Master Master-slave*	*Slave Master-slave*
Min	16.35 %	18.94 %	21.83 %
Max	18.12 %	19.25%	22.05 %
Avarage	17.30 %	19.15 %	21.83 %
Peningkatan	1,77 %	0,31 %	0,22 %

4. KESIMPULAN

Adapun kesimpulan dari pengujian *database master only* dan *master-slave*, yaitu:

- Operasional database cenderung menggunakan CPU dibandingkan *memory*.
- Total *request* yang dapat diterima mengunakan *master-slave* meningkat dengan nilai rata-rata 258,31 % dibandingkan menggunakan *master only*.
- Response time proses write dan read yang didapat menggunakan *master-slave* meningkat dengan nilai rata-rata 219,32 % dibandingkan menggunakan *master only*.

5. SARAN

Perkembangan sistem informasi yang pesat dan memerlukan skalabilitas yang lebih besar, pengunaan *database master-slave replication* dapat dikombinasikan dengan *database sharding*. *Sharding* membagi data menjadi beberapa *subset* yang disimpan di *server* yang berbeda, sehingga meningkatkan skalabilitas dan mengurangi beban pada *master server*.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tim Redaksi Jurnal Teknika Politeknik Negeri Sriwijaya yang telah memberi kesempatan, sehingga artikel ilmiah ini dapat diterbitkan.

DAFTAR PUSTAKA

[1] W. Murni Wijaya dan Z. Mutaqin Subekti, “Penerapan Aplikasi Database pada Kegiatan Manajemen Sekolah,” vol. 3, no. 1, hlm. 158–167, 2019.

[2] A. Setiawan dan W. Muthia Kansha, “Development of Cluster Database System Using Galera Cluster Application at Vocational School of IPB University,” vol. 11, no. 2, hlm. 49–59, 2021, doi: 10.29244/jstsv.11.2.49.

[3] F. Huzaeni, I. Gunawan, D. Cahya Purnomo, M. Yanti, dan N. Krisdayanti abcde Teknik Elektro Sekolah Tinggi Teknologi Ronggolawe Cepu Penulis Korenspondensi, “Analisis Keamanan Data Pada Website Dengan Wireshark,” 2021. [Daring]. Tersedia pada: http://sttrcepu.ac.id/siakapt/login

[4] M. Hasbi dan N. R. Saputra, “Analisis Quality Of Service (QOS) Jaringan Internet Kantor Pusat King Bukopin dengan Menggunakan Wireshark,” 2021. [Daring]. Tersedia pada: https://jurnal.umj.ac.id/index.php/just-it/index

[5] E. Asriyar dan T. Sutendi, “Implementasi Sistem Replikasi Data Base Postgresql Master-Slave Repmgr dengan Auto Promote MasterDB,” 2019.

[6] F. S. Redha, R. P. Sari, dan S. Rahmayuda, "Perbandingan Performa Web Services Yang Dibangun Menggunakan Arsitektur Monolithic Dan Microservices Pada Sistem Point of Sales," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 10, 2023.

[7] A. B. Cahyo, T. K. Hariadi, dan Y. Ardiyanto, "Implementasi Zabbix Server untuk Memonitor Kondisi Jaringan Komputer di Dinas Komunikasi dan Informatika Kabupaten Pekalongan," 2020.

[8] D. Intan Permatasari *dkk.*, "Pengujian Aplikasi Menggunakan Metode Load Testing dengan Apache Jmeter pada Sistem Informasi Pertanian," *Jurnal Sistem dan Teknologi Informasi*, vol. 8, 2020.

[9] R. Hermiati, Asnawati, dan I. Kanedi, "Pembuatan E-Commerce Pada Raja Komputer Menggunakan Bahasa Pemrograman PHP dan Database MySQL," *Jurnal Media Infotama*, vol. 17, 2021.

[10] R. Wiji Wahyuningrum dan E. Haerullah, "Analisis Monitoring Sistem Jaringan Komputer Menggunakan Aplikasi Spiceworks," *PROSISKO*, vol. 9, 2022.

[11] R. Nur, D. Aryani, T. Elektro, dan P. Negeri Ujung Pandang, "Pemodelan Basis Data Pada Sistem Informasi Laporan Kinerja Program Studi (LKPS) Berbasis Instrumen Akreditasi Program Studi (IAPS 4.0)," 2020.

[12] K. Christiono dan H. Sama, "Studi Komparasi Database Management System Antara Maria DB Dan Postgresql Terhadap Efisiensi Penggunaan Sumber Daya Komputer," 2020, [Daring]. Tersedia pada: http://journal.uib.ac.id/index.php/cbssit

[13] A. Y. Chandra, "Analisis Performansi Antara Apache & Nginx Web Server Dalam Menangani Client Request," *Jurnal Sistem dan Informatika (JSI)*, vol. 14, no. 1, hlm. 48–56, Nov 2019, doi: 10.30864/jsi.v14i1.248.

[14] N. K. Akmal dan M. N. Dasaprawira, "Rancang Bangun Application Programming Interface (API) Menggunakan Gaya Arsitektur Graphql Untuk Pembuatan Sistem Informasi Pendataan Anggota Unit Kegiatan Mahasiswa (UKM) Studi Kasus UKM Starlabs," *Jurnal Sistem Informasi dan Teknologi (SITECH)*, 2022, [Daring]. Tersedia pada: http://www.jurnal.umk.ac.id/sitech

[15] R. Farah Humainah, A. Fikri Tri Firmansyah, S. Iman Ramadhani, dan D. Aribowo, "Pengaruh Kapasitas Memori RAM (Random Access Memory) Terhadap Kecepatan Memori Pada Laptop," *Jurnal Elektronika dan Teknik Informatika Terapan (JENTIK)*, vol. 1, hlm. 178–186, 2023.

[16] D. Andira Sulaeman, I. Nurul Sya'bani, M. Ashof Azria Azka, dan D. Aribowo, "Ruang Lingkup Organisasi Dan Arsiektur Komputer," *Jurnal Elektronika dan Teknik Informatika Terapan (JENTIK)*, vol. 1, hlm. 164–177, 2023.