Jurnal ICT : Information Communication & Technology
Vol. 23, N0.1, Juli 2023, pp. 297-303
p-ISSN: 2302-0261, e-ISSN: 2303-3363

Optimalisasi *Open Source Relational Database* Menggunakan Metode *Replication* Dan Metode *Cluster*

Arief Ginanjar[1*], Mokhamad Hendayun[2], Wahyu Purnama Sari[3]
[1,2,3] Program Studi Teknik Informatika, Universitas Langlangbuana, Bandung, Indonesia
Email: [1] arief.ginanjar@unla.ac.id, [2] hendayun@unla.ac.id, [3] wahyu.alyapurnama@gmail.com

INFORMASI ARTIKEL

Histori artikel:

Naskah masuk, 06 Mei 2023
Direvisi, 03 Agustus 2023
Diiterima, 07 Agustus 2023

Kata Kunci:

Clustering
Replication
Database
Virtual Machine
Optimize Database Server

ABSTRAK

Abstract- *Information technology are developed very fast in the present days. Along with the development of time, the amount of data and users will also increase which results in process of accessed and process workload will become slower. The ability to provide fast retrieve information is highly demanded to be more efficient and effective at these time. To overcome this problem there are technologies called Cluster and Replication that can help ensure high service availability by utilizing several computer resources that connected and work together. If any of these server experiences failure, the system will not be immediately disrupted because other servers will continue functioning and replace the task of main server. Cluster capabilities allow a database to stay alive for a long time. Replications are techniques to copy and distribute database and objects from one to another database and synchronise between them. And data consistency can be keepted. Testing is worked with the simultaneous access technique using a script loop. Testing results with clusters tend to be faster, and testing between double VMs and triple VMs looks superior to triple VMs so it can make new alternative of choice in implementing database programming.*

Abstrak- Dengan perkembangan teknologi informasi yang sangat cepat beriringan dengan perkembangan jumlah data informasi serta pengguna yang mengakibatkan proses akses serta beban kerja sistem akan lebih berat. Sedangkan kemampuan dalam ketersediaan informasi sangat dituntut untuk bisa lebih efisien dan efektif setiap saat. Adapun solusi untuk mengatasi masalah tersebut menggunakan teknologi *clustering* dan *replication* yang dapat membantu menjamin ketersediaan layanan yang tinggi dengan memanfaatkan beberapa komputer atau *machine server* yang saling terkoneksi. Jika salah satu *server* mengalami gagal akses maka sistem tidak langsung terganggu, hal tersebut disebabkan *server* lain yang sudah terkoneksi akan tetap berfungsi dan menggantikan kerja *server* yang terganggu tersebut. Dengan metode *clustering* memungkinkan *database* server tetap berfungsi melayani pengguna. Sedangkan *replication* merupakan teknik untuk melakukan *clone* serta distribusi data serta objek dari satu *server database* ke *server database* lain dan melaksanakan sinkron antara *server database* yang menyebabkan konsistensi data tetap terjaga. Pengujian dilakukan dengan teknik simultan akses menggunakan *loop script*. Hasil pengujian menggunakan *cluster* cenderung menghasilkan respon lebih cepat antara dua VM dan tiga VM, sehingga tiga VM terlihat lebih unggul sehingga dapat menjadikan alternatif pilihan baru dalam teknik implementasi *database*.

Penulis Korespondensi:

Arief Ginanjar
Program Studi Teknik Informatika,
Universitas Langlangbuana
Jl. Karapitan No.114, Bandung, Indonesia
Email: arief.ginanjar@unla.ac.id

1. Pendahuluan

Database merupakan kumpulan informasi yang tersimpan dalam *storage* dan tersusun secara sistematis supaya dapat digunakan oleh komputer atau pengguna untuk memperoleh informasi dari *database* tersebut. Manfaat lain yang dari *database* bagi pengguna adalah proses pengambilan data informasi yang cepat dan mudah, supaya dapat memberikan layanan yang terbaik kepada pengguna [1]. Dengan menggunakan *database* maka perolehan informasi serta layanan, cepat dan efisien. Salah satu *database* yang sering digunakan adalah RDBMS atau *Relational Database Management System*, model ini paling populer digunakan karena dinilai efisien dengan model yang mudah sehingga sederhana dalam memahami dan mudah dalam penggunaan [2]. Model *database* dengan tabel dua dimensi yang terdiri dari baris dan kolom serta model penamaan yang unik. Dengan metode penamaan unik maka antara *data relational* yang digunakan, dengan nama yang unik akan menjaga relasi nilai yang satu dan lainnya. Setiap baris pada tabel harus mempunyai identitas unik yang disebut *primary key*, serta baris di antara tabel dapat dibuat saling terkait menggunakan *foreign key* [3].

Metode yang dapat digunakan untuk solus terhadap masalah terkait ketersediaan sumber data yang ingin diakses adalah dengan menggunakan *replication database*. Hal tersebut merupakan teknologi yang digunakan untuk *clone data* dan distribusi data dari *server database* terhadap beberapa *server database* lain yang sudah saling terkoneksi [4]. Tanpa *replication, backup* akan menghambat kinerja sistem dan berpotensi mengalami data tidak konsisten, yang disebabkan satu tabel berubah sementara tabel lain yang mempunyai relasi tidak berubah karena sedang dalam proses *backup*. *Shutdown server* dari layanan *user* akan menjamin data yang konsisten, namun hal ini berarti mengganggu layanan terhadap pengguna dan hal ini sangat tidak diharapkan terjadi pada penyedia layanan[5].

Solusi lain yang mungkin dapat dilakukan untuk memenuhi sumber daya dari data yang dibutuhkan yaitu dengan menambah *server* dan menerapkan metode *cluster*, dimana kumpulan *server* yang melayani *client* secara merata sehingga jumlah *concurrent connection* dapat meningkat dan ketersediaan akses *server* lebih tinggi. *Cluster* adalah sekelompok mesin yang bertindak sebagai entitas tunggal untuk menyediakan *resource* dan *service network*. Pada metode *clustering server* ini terdapat metode unggulan yaitu sebagai *failover cluster* dan *load balancing cluster*[6]. Fungsi *failover* bekerja ketika salah satu *server* yang terkoneksi bagian dari *cluster* (*node*) mengalami gagal akses atau *shutdown* maka akan tergantikan oleh *server* lain secara otomatis, supaya layanan tidak mengalami gangguan. Sedangkan fungsi *load balancer cluster* bekerja ketika *server* menangani permintaan dari *user*, maka permintaan akan dibagi secara merata ke semua *node* pada *cluster* sehingga *server* tidak akan mengalami *overload* [7].

Adapun tujuan *replication* dan *cluster* pada dasarnya mempunyai fungsi yang sama, namun mempunyai konfigurasi yang berbeda. Sehingga dengan penelitian ini diharapkan dapat menjawab pertanyaan terkait apakah perbedaan konfigurasi dapat memecahkan masalah dengan pembacaan data yang sama cepat atau akan memiliki perbedaan hasil.

Tujuan penelitian yang ingin dicapai ialah mengetahui waktu akses respon dari setiap spesimen seperti yang tercantum dalam Tabel 1. Dengan proses pengujian pembacaan satu *set* data *raw random* nama kecamatan dan kelurahan di Indonesia.

Dengan serangkaian pengujian tersebut diharapkan diketahui tingkat kecepatan dan akurasi dari masing-masing pengujian spesimen seperti yang tercantum di Tabel 1. Diharapkan dapat menjadi sumber referensi baru dalam pengambilan keputusan, untuk para *programmer* ketika harus memilih jenis *scalable* RDMS yang dapat diterapkan dalam implementasi aplikasi.

Tabel 1. Rencana spesimen yang akan diuji.

Spesimen	*Scalable* RDBMS	Perancangan *Hardware*
1	*Replication*	*Double VM*
2	*Replication*	*Triple VM*
3	*Cluster*	*Double VM*
4	*Cluster*	*Triple VM*

2. Metode Penulisan

Penelitian ini bersifat eksploratif yang artinya dilakukan dengan cara menggali informasi dari subjek penelitian. Metode yang digunakan dalam penelitian ini adalah model *prototype*. Menurut Pressman model *prototype* atau *incremental process* model merupakan metodologi bersifat sistematis yang terdiri dari *communication, planning, modeling, construction* dan *deployment* kemudian dilengkapi filosofi *iterative*, ada tahapan yang dapat dilakukan berulang-ulang hingga kebutuhan dalam tahapan tersebut terpenuhi [8].

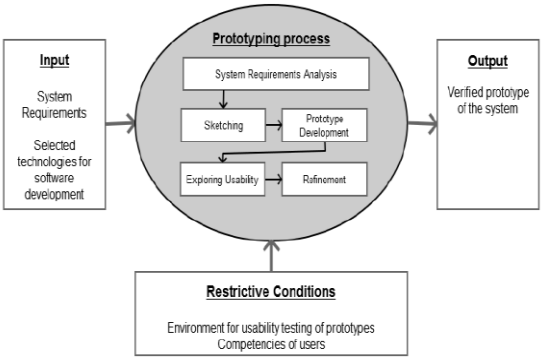
Metode yang dirujuk pada penyusunan laporan, menggunakan metode *prototype* dengan

modifikasi *evolutionary* dengan penambahan proses uji performa *software*:

1. Mempelajari Literatur
   Mempelajari literatur yang digunakan sebagai sumber penelitian. Jenis pustaka dapat berupa book chapter, *paper* atau *webpage* yang berkaitan materi dengan *MariaDB, Cluster* dan *Replication*.
2. Analisis
   Proses melakukan analisis terhadap perancangan perangkat lunak dan teknik pemrograman serta bagaimana mengkombinasikan teknologi tersebut.
3. Perancangan Perangkat Lunak
   Proses perancangan *software* digabungkan dengan *framework* yang akan dikembangkan berdasarkan hasil analisis. Dokumen hasil analisis tersebut harus memenuhi scenario pengujian yang telah ditetapkan [9].
4. Implementasi Perangkat Lunak
   Proses penyusunan perangkat lunak yang dibangun berdasarkan hasil perancangan. Proses ini akan menghasilkan produk perangkat lunak yang berisi *framework* dengan variable dan parameter telah ditentukan dalam dokumen analisis [10].
5. Evaluasi
   Melakukan proses uji hasil terhadap software yang telah dikembangkan serta melakukan evaluasi dari setiap *scenario* uji yang didefinisikan.
6. Pengulangan rancangan dan *development software* dilanjutkan dengan Evaluasi.
   Melakukan proses uji terhadap data dan sofware yang telah dibangun ketika hasil uji belum memenuhi variabel yang direncanakan maka perlu pengulangan rancangan dan *development* [11].
7. Pengujian Unjuk Kerja *Software*
   Proses uji performa software terhadap *stres* dengan metode *loop loading url* dilaksanakan berulang terhadap data pengujian menggunakan metode *url looping script* dengan *script code*.

2.1 Evolutionary Prototype Modeling

Tahapan yang telah dilakukan sebagai bagian dari pelaksanaan penelitian yang merujuk kepada metode *evolutionary prototype model,* ilustrasi dari metode tersebut terlihat pada Gambar 1. Proses *evolutionary prototype* terdiri dari beberapa proses yaitu masukan, *prototype process*, dan luaran serta dibatasi dengan *restrictive*, setiap fungsi yang ada dalam sistem harus memenuhi variable dan parameter yang telah ditentukan dalam *system requirement* [12].



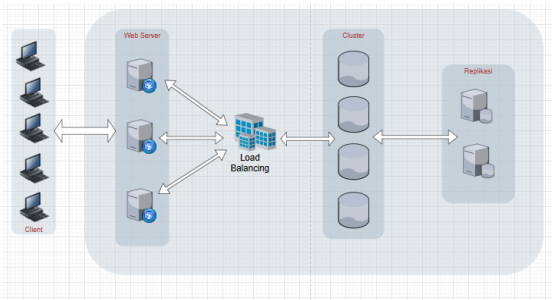
Gambar 1. Metode *Evolutionary Prototyping* [12].

Adapun tahapan yang harus dilakukan sesuai dengan metode *evolutionary prototype* [13]:

1. Input merupakan proses *System Requirement*, berupa proses mengumpulkan informasi sistem yang diperlukan serta tujuan dan gambaran komponen yang dinginkan.
2. *Prototyping Process*, merancang *software* yang terdiri dari lima bagian:
   a. *System Requirement Analysis*.
   b. *Sketching*.
   c. *Prototype Development*.
   d. *Exploring Usability*.
3. *Refinement*, evaluasi dan perbaikan proses kerja dari masing-masing skenario uji yang dilakukan dengan *restrictive conditions* [14].
4. *Output*, dalam langkah ini dilakukan verifikasi hasil dari langkah yang telah dilakukan sebelumnya terhadap kebutuhan sistem.

2.2 Arsitektur Spesimen

Dalam sebuah teknologi untuk melakukan transfer data dan membagi beban trafik agar dapat berjalan secara optimal dapat dilakukan dengan *replication* atau *cluster database*[6]. Dengan persamaan fungsi *replication* dan *cluster* dapat digambarkan bentuk arsitektur spesimen sebagai berikut:



Gambar 2. Arsitektur Spesimen

3. Proses Penelitian

3.1 Konfigurasi Spesimen

Di dalam pengujian spesimen yang telah dilakukan terdapat aspek teknis yang diterapkan

yaitu; pengujian yang dilakukan terdiri dari beberapa *layer* sistem yaitu, *layer database*, *layer application container*, *layer logic programming* serta *layer script* uji [15]. Kemudian dalam masing-masing *layer* terdapat beberapa konfigurasi yang digunakan sistem untuk saling berinteraksi dengan *layer* lain.

Tabel 2. *Code* konfigurasi setiap lapisan

No	Lapisan	Teknologi	Konfigurasi
1	*Database*	*MariaDB*	*Default*
2	*Application Container*	*Apache PHP*	*Default*
3	*Logic Programming*	*PHP Framework*	Spesifik
4	*Script* Pengujian	*PHP Programming*	Spesifik

Dari aplikasi yang dibangun yang berfungsi untuk melakukan pengujian pembacaan data kelurahan dan kecamatan secara *random*, menggunakan *MariaDB cluster* atau *MariaDB replication* yang disimpan dibelakang *load balancer* menggunakan *HAProxy*, maka kita akan mengetahui spesifikasi konfigurasi dengan *script* yang dapat dilihat pada Gambar 4.

```
try {
    PriatyMain pic = new PriatyMain(args[0], args[1], args[2]);
    ExecutorService es = Executors.newCachedThreadPool();

    pic.pfd.preDigFilling(pic.pfd.fdrPathStr, encodedStr, finalString, pic.waktuAntara);
    pic.pfd.cekPreDigFilling(); pic.pfd.digFiling(encodedStr, finalString, pic.waktuAntara, es);

    while (true) {
} catch (Exception e) {
    e.printStackTrace();
}
```

Gambar 4. Kode sumber *startup load* kecamatan dan kelurahan.

```
while (true) {
    int jmlAwal = pic.pfd.tmpList.size();
    int jmlAkhir = pic.dic.getCountDataByteArrayFingerPrint();
    if (jmlAwal == jmlAkhir) {
        pic.pe.getPreElitsm(2);
        pic.pe.getElitsm(pic.waktuAntara);

        while (true) {
            int jmlElAwal = pic.pe.elPreList.size();
            int jmlElAkhir = pic.dic.getDataCountElitsmPre();
            if (jmlElAwal == jmlElAkhir) {
                pic.ps.getCrossOver(pic.waktuAntara);

                while (true) {
                    int jmlCro = pic.ps.jmlHasilKawin;
                    int jmlCroAkhir = pic.dic.getDataCountByteArrayCrossOver();
                    if (jmlCro == jmlCroAkhir) {
                        pic.pm.cekPreMutationTwo();
                        pic.pm.getMutationTwo(pic.waktuAntara); break;
                    } else { }
                } break;
            } else { }
        } break;
    } else { }
}
```

Gambar 5. Kode sumber seleksi kondisi *load* data kecamatan dan kelurahan.

```
private Properties getProperties() {
    if (properties == null) {
        properties = new Properties();
        properties.setProperty("user", USERNAME);
        properties.setProperty("password", PASSWORD);
        properties.setProperty("MaxPooledStatements", MAX_POOL);
    }
    return properties;
}

public Connection connect() throws SQLException {
    if (connection == null) {
        try {
            Class.forName(DATABASE_DRIVER);
            connection = DriverManager.getConnection(DATABASE_URL, getProperties());
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
    }
    return connection;
}
```

Gambar 6. Kode sumber konfigurasi koneksi *database*.

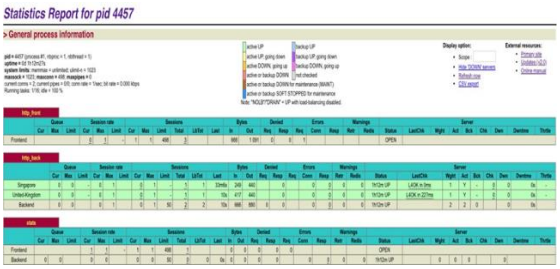
Setelah konfigurasi aplikasi sesuai dengan spesimen data di dalam database seperti yang terlihat pada Tabel 1. Serta sudah *install* dalam lingkungan *Apache* – PHP - *MariaDB* seperti yang terlihat pada Tabel 2. Untuk kode sumber yang terlihat pada gambar 4 merupakan kode *startup* dari aplikasi dimana aplikasi akan mulai *running* menggunakan *script* tersebut, sedangkan untuk gambar 5 merupakan proses awal akuisisi data kelurahan dan kecamatan secara *random*, diakhiri dengan kode untuk melakukan koneksi terhadap *database* dapat terlihat pada gambar 6.

```
defaults
  mode http
  timeout client 10s
  timeout connect 5s
  timeout server 10s
  timeout http-request 10s

frontend myfrontend
  bind 127.0.0.1:80
  default_backend myservers

backend myservers
  server server1 127.0.0.1:8000
  server server2 127.0.0.1:8001
  server server3 127.0.0.1:8002
```

Gambar 7. Kode sumber konfigurasi untuk koneksi *load balancer* menggunakan konsep *replication*.

Pada Gambar 7 terlihat *script* konfigurasi menghubungkan tiga database server dengan brand *MariaDB* menggunakan *HAProxy*, terdapat tiga konfigurasi utama yaitu; *defaults* yang berfungsi untuk mengatur prilaku sistem *load balancer* secara umum; *frontend* yang mengatur akses internet terhadap *HAproxy*, kemudian *backend* yang menyambungkan *HAProxy* dengan *MariaDB replication* yang sudah disimpan dalam VM dengan alamat IP *address* yang sudah ditentukan.

Sedangkan pada gambar 8 terlihat hasil dari konfigurasi *load balancer* dengan konsep *replication* menggunakan HAProxy, dengan trafik yang terlihat pada gambar yaitu *http_front*, *http_back dan total load*.



Gambar 8. Hasil konfigurasi *load balancer* dengan konsep *replication* menggunakan *HAProxy*.

```
$ sudo vim /etc/mysql/mariadb.conf.d/50-server.cnf
[galera]
wsrep_on                = ON
wsrep_cluster_name      = "MariaDB Galera Cluster"
wsrep_provider          = /usr/lib/galera/libgalera_smm.so
wsrep_cluster_address   = "gcomm://node1,node2,node3"
binlog_format           = row
default_storage_engine  = InnoDB
innodb_autoinc_lock_mode = 2
bind-address = 0.0.0.0
wsrep_node_address="node1"
```

Gambar 9. Kode sumber konfigurasi *load balancer* dengan konsep *cluster* menggunakan ProxySQL untuk *node* 1.

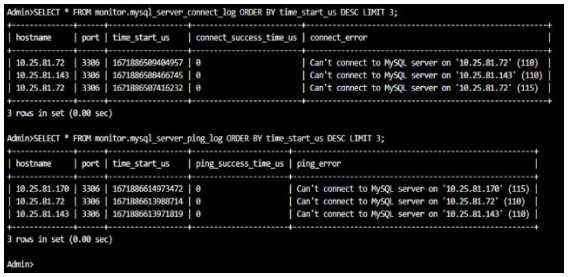
Pada gambar 9, gambar 10 dan gambar 11 merupakan konfigurasi MariaDB *scalable* menggunakan konsep *cluster*, dimana proses *synchronize* dari masing-masing *node* 1, *node* 2 dan *node* 3 ada di tingkatan *atomic*.

```
$ sudo vim /etc/mysql/mariadb.conf.d/50-server.cnf
[galera]
wsrep_on                = ON
wsrep_cluster_name      = "MariaDB Galera Cluster"
wsrep_provider          = /usr/lib/galera/libgalera_smm.so
wsrep_cluster_address   = "gcomm://node1,node2,node3"
binlog_format           = row
default_storage_engine  = InnoDB
innodb_autoinc_lock_mode = 2
bind-address = 0.0.0.0
wsrep_node_address="node2"
```

Gambar 10. Kode sumber konfigurasi *load balancer* dengan konsep *cluster* menggunakan *ProxySQL* untuk *node* 2.

```
$ sudo vim /etc/mysql/mariadb.conf.d/50-server.cnf
[galera]
wsrep_on                = ON
wsrep_cluster_name      = "MariaDB Galera Cluster"
wsrep_provider          = /usr/lib/galera/libgalera_smm.so
wsrep_cluster_address   = "gcomm://node1,node2,node3"
binlog_format           = row
default_storage_engine  = InnoDB
innodb_autoinc_lock_mode = 2
bind-address = 0.0.0.0
wsrep_node_address="node3"
```

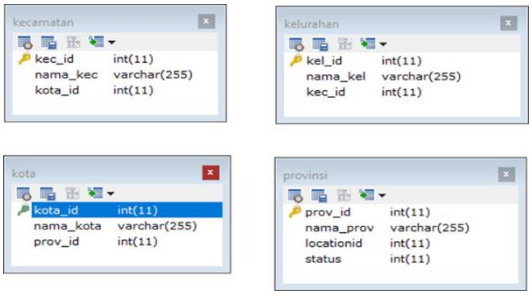
Gambar 11. Kode sumber konfigurasi *load balancer* dengan konsep *cluster* menggunakan ProxySQL untuk *node* 3.



```
Admin>SELECT * FROM monitor.mysql_server_connect_log ORDER BY time_start_us DESC LIMIT 3;
+-----------+------+------------------+-------------------------+---------------------------------------------------+
| hostname  | port | time_start_us    | connect_success_time_us | connect_error                                     |
+-----------+------+------------------+-------------------------+---------------------------------------------------+
| 10.25.81.72  | 3306 | 1671886580404957 | 0                       | Can't connect to MySQL server on '10.25.81.72' (110) |
| 10.25.81.143 | 3306 | 1671886580466745 | 0                       | Can't connect to MySQL server on '10.25.81.143' (110) |
| 10.25.81.72  | 3306 | 1671886587416232 | 0                       | Can't connect to MySQL server on '10.25.81.72' (115) |
+-----------+------+------------------+-------------------------+---------------------------------------------------+
3 rows in set (0.00 sec)

Admin>SELECT * FROM monitor.mysql_server_ping_log ORDER BY time_start_us DESC LIMIT 3;
+-----------+------+------------------+-------------------+---------------------------------------------------+
| hostname  | port | time_start_us    | ping_success_time_us | ping_error                                     |
+-----------+------+------------------+-------------------+---------------------------------------------------+
| 10.25.81.170 | 3306 | 1671886614973472 | 0                 | Can't connect to MySQL server on '10.25.81.170' (115) |
| 10.25.81.72  | 3306 | 1671886613988714 | 0                 | Can't connect to MySQL server on '10.25.81.72' (110) |
| 10.25.81.143 | 3306 | 1671886613971819 | 0                 | Can't connect to MySQL server on '10.25.81.143' (110) |
+-----------+------+------------------+-------------------+---------------------------------------------------+
3 rows in set (0.00 sec)

Admin>
```

Gambar 12. Hasil konfigurasi *load balancer* dengan konsep *cluster* menggunakan ProxySQL

Keunggulan dari proses *atomic* adalah proses *synchronize* data tidak menyebabkan proses *transaction* data harus terhenti karena ada proses *table locking* atau *row locking*. Sedangkan pada gambar 12 terlihat hasil dari konfigurasi tiga node MariaDB yang disambungkan menggunakan jaringan *local* terhadap *load balancer* dengan *brand ProxySQL* menggunakan konsep *cluster*. Dengan kolom status *connects* dan status *ping* yang menandakan masing-masing *node server* saling terkoneksi.



Gambar 13. Struktur data tabel kecamatan dan kelurahan di Indonesia.

Setelah seluruh konfigurasi berhasil diimplementasikan maka dengan menggunakan struktur tabel data yang terlihat pada gambar 13 kemudian peneliti mencoba menguji eksekusi *query* data menggunakan *query script* yang terlihat pada gambar 14.

```
SELECT nama_prov, nama_kota, nama_kec, nama_kel
FROM provinsi a, kota b, kecamatan c, kelurahan d
WHERE a.prov_id = b.prov_id AND b.kota_id = c.kota_id AND c.kec_id = d.kec_id
ORDER BY a.prov_id, b.kota_id, c.kec_id, d.kel_id
```

Gambar 14. *Syntax query* untuk *load* data kecamatan dan kelurahan seluruh Indonesia.

nama_prov	nama_kota	nama_kec	nama_kel
ACEH	PIDIE JAYA	BANDAR BARU	ABAH LUENG
ACEH	PIDIE JAYA	BANDAR BARU	AKI NEUNGOH
ACEH	PIDIE JAYA	BANDAR BARU	ALUE
ACEH	PIDIE JAYA	BANDAR BARU	ARA
ACEH	PIDIE JAYA	BANDAR BARU	BALEE
ACEH	PIDIE JAYA	BANDAR BARU	BAROH COT
ACEH	PIDIE JAYA	BANDAR BARU	BAROH LAMCOK
ACEH	PIDIE JAYA	BANDAR BARU	BAROH MUDA
ACEH	PIDIE JAYA	BANDAR BARU	BEURANDEH
ACEH	PIDIE JAYA	BANDAR BARU	BLANG BABO
ACEH	PIDIE JAYA	BANDAR BARU	BLANG GLONG
ACEH	PIDIE JAYA	BANDAR BARU	BLANG IBOIH
ACEH	PIDIE JAYA	BANDAR BARU	BLANG MEUENG
ACEH	PIDIE JAYA	BANDAR BARU	BLANG SUNOM
ACEH	PIDIE JAYA	BANDAR BARU	COT LAMOIEN
ACEH	PIDIE JAYA	BANDAR BARU	DAROIH
ACEH	PIDIE JAYA	BANDAR BARU	DAYAH LAMOIEN

Gambar 15. Hasil *query* untuk *load* data kecamatan dan kelurahan seluruh Indonesia.

Gambar 15 merupakan tampilan hasil *query* dari data kecamatan dan kelurahan seluruh

Indonesia, yang akan menjadi langkah awal mengeksekusi pengujian performa antara *cluster* dan *replication* yang berasal dari *syntax query* pada Gambar 14.

3.2 Hasil Penelitian

Dari hasil pengujian berdasarkan spesimen yang telah ditetapkan di awal. Ditemukan kondisi bahwa spesimen *cluster* menunjukan hasil pengujian yang cukup signifikan tinggi dalam waktu respon. Hal tersebut menunjukan *cluster mariaDB* mempunyai kemampuan lebih dalam mengelola data *cache* dari *node* dengan respon yang tinggi dibandingkan dengan *replication*[16]. Hal tersebut menjadikan MariaDB *cluster* unggul dibandingkan dengan MariaDB *replication*. Seperti terlihat pada tabel 4 dan tabel 5.

Tabel 4. Hasil pengujian *replication*.

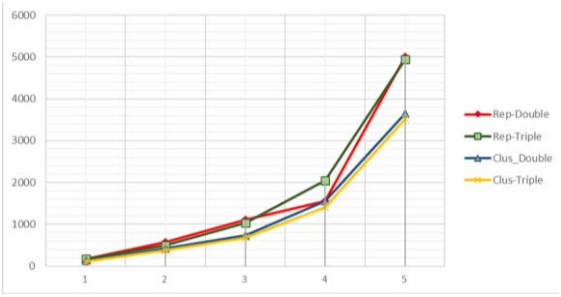
No. Pengujian		Konfigurasi *Default*	Kecepatan Respon *Replication*	
			Dua VM	Tiga VM
1	100 hit	152,796	165	178
2	500 hit	40.612,5	572	492
3	1000 hit	81.225	1.102	1.040
4	2000 hit	162.450	1.548	2.052
5	5000 hit	406.125	5.005	4.942

Selain dari sisi respon, kelemahan dari *replication* adalah ketika terjadi kendala disaat transaksi *create, update* dan *delete* data, maka butuh *effort* manual dari administrator *database* untuk mengatasi masalah *synchronize* data. Sedangkan *MariaDB cluster* hal tersebut dapat dilakukan *resolve* secara otomatis[17].

Tabel 5. Hasil pengujian *cluster*.

No. Pengujian		Konfigurasi *Default*	Kecepatan Respon *Cluster*	
			Dua VM	Tiga VM
1	100 hit	152,796	133	113
2	500 hit	40.612,5	432	385
3	1000 hit	81.225	739	679
4	2000 hit	162.450	1.56	1.401
5	5000 hit	406.125	3.659	3.516

Dari data hasil pengujian yang terdapat pada tabel 4 dan tabel 5 terlihat hasil uji dari *query syntax* yang terdapat pada gambar 15 dan menghasilkan data yang terlihat pada gambar 16 dengan total data yang dihasilkan sebanyak 81.225 baris data. Kemudian *syntax* tersebut dilakukan uji load data sebanyak jumlah pengujian seperti yang terlihat pada tabel 4 dan tabel 5 kolom pengujian, dimana terdapat pengujian sebanyak 100 *hit*, 500 *hit*, 1000 *hit*, 2000 *hit* dan 5000 *hit*. Dari data tersebut kemudian transformasi menjadi diagram garis maka akan terlihat performa setiap spesimen terlihat pada gambar 16 seperti berikut.



Gambar 16. Hasil pengujian jumlah *hit* dibanding dengan waktu proses.

Hasil pengujian performa antara proses data menggunakan *replication* dengan *cluster* menunjukkan pada spesimen *cluster* menggunakan tiga VM mampu memproses data lebih cepat dibandingkan spesimen lain. Adapun data yang diperbandingkan pada gambar 16 yaitu antara jumlah pengujian yang diwakili menggunakan angka 1, 2, 3, 4 hingga 5 yaitu angka urut pengujian seperti yang terlihat pada tabel 4 dan tabel 5 kemudian dibandingkan dengan waktu yang dibutuhkan untuk setiap pengujian seperti yang terlihat pada diagram garis pada sumbu y yaitu variabel waktu dengan satuan *milisecond*. Hal tersebut terlihat pada gambar 16.

Tabel 6 Nilai pengujian *replication* dan *cluster*.

No	Pengujian	Perbandingan (Persen)	
		Replication vs Cluster Dua VM	*Replication vs Cluster* Tiga VM
1	100 hit	24,06	57,52
2	500 hit	31,49	27,79
3	1000 hit	49,12	53,17
4	2000 hit	2,79	46,47
5	5000 hit	36,79	40,56
Rata-rata		28,85	45,10
Rata-rata Total		36,98	

Menggunakan rumus perbandingan nilai antara waktu proses diantara spesimen, yaitu perbandingan spesimen satu dengan tiga dan perbandingan spesimen dua dengan empat, menggunakan rumus berikut.

$$Nilai\ Presentase = \left(\frac{Nilai\ Awal - Nilai\ Akhir}{Nilai\ Akhir}\right) \quad (1)$$

Maka didapat nilai perbandingan yang terlihat pada Tabel 6. Dari tabel tersebut diketahui penggunaan *cluster* dapat meningkatkan kecepatan proses rata-rata sebesar 28.85% untuk dua VM dan 45.10% untuk tiga VM, dengan rata-rata total sebesar 36.98%.

4. Kesimpulan

Dengan penelitian yang telah dilakukan tentang tentang hasil pengujian kecepatan pembacaan 81.225 baris data menggunakan *scalable replication* dan *scalable cluster*, terdapat beberapa hasil yang dapat diketahui yaitu; Hasil

pengujian menggunakan *cluster* cenderung lebih cepat dengan rata-rata total mencapai 36.98%, Ketika melihat hasil pengujian antara dua VM dan tiga VM terlihat tiga VM lebih unggul, disebabkan penggunaan *resource* yang lebih banyak sehingga beban kerja menjadi lebih ringan antara VM dibanding dengan dua VM.

Kesimpulan diatas dapat dijadikan referensi baru tentang cara mengoptimalkan kemampuan *server database* dalam memproses data besar, sehingga dapat menjadikan alternatif pilihan baru dalam *pemrograman database*. Adapun saran yang dapat disampaikan untuk penelitian selanjutnya adalah dengan mengaktifkan *object cache* pada level database untuk *query join*.

Daftar Pustaka

[1] W. G. E. Bratha, "Literature Review Komponen Sistem Informasi Manajemen: Software, Database Dan Brainware," *J. Ekon. Manaj. Sist. Inf.*, vol. 3, no. 3, pp. 344–360, 2022, doi: 10.31933/jemsi.v3i3.824.

[2] Sugiyatno, "Perancangan Clustering Database Server Untuk Meningkatkan Unjuk Kerja Server Dan Menjamin Ketersediaan Layanan," *J. Cendikia*, vol. XVIII, pp. 281–289, 2019.

[3] W. S. Prasetya, "Perancangan Model Database Relasional Dengan Metode Database Life Cycle," *Pros. Semin. Nas. Inform. 2015*, vol. 1, no. 1, pp. 91–98, 2015.

[4] H. Maulana, "Analisis Dan Perancangan Sistem Replication Database Mysql Dengan Menggunakan Vmware Pada Sistem Operasi Open Source," *InfoTekJar (Jurnal Nas. Inform. dan Teknol. Jaringan)*, vol. 1, no. 1, pp. 32–37, 2016, doi: 10.30743/infotekjar.v1i1.37.

[5] B. Raharjo, *Membuat Database Menggunakan MySQL*. Bandung: Informatika, 2011.

[6] A. Lentz and M. Ab, "Introduction to MySQL Cluster: Architecture and Use," 2006.

[7] T. M. Connolly and C. E. Begg, *Database System: A Practical Aproach to Design, Implementation and Management*, 3nd Editio. England: Pearson Education, 2005.

[8] R. S. Pressman, *A Manager's Guide to Software Engineering*. New York: McGraw-Hill, Inc., 1996.

[9] B. Ilman and A. Ginanjar, "Rancang Bangun Web Service-JSON Menggunakan Kombinasi Spring dan MyBatis Framework dalam lingkungan Java Platform," *J. Teknol.*, vol. 9, no. 1, 2019.

[10] R. S. Pressman, *Rekayasa Perangkat Lunak Pendekatan Praktisi*, Buku Satu. Yogyakarta: Andi, 2002.

[11] M. Bolung and H. R. K. Tampangela, "Analisa Penggunaan Metodologi Pengembangan Perangkat Lunak," *J. ELTIKOM*, vol. 1, no. 1, pp. 1–10, 2017, doi: 10.31961/eltikom.v1i1.1.

[12] R. Nacheva, "Prototyping Approach in User Interface Development," in *Second Conference on Innovative Teaching Methods (ITM 2017) 28-29 June*, 2017, p. 78.

[13] D. Susandi, W. Nugraha, and S. F. Rodiyansyah, "Perancangan Smart Parking System Pada Prototype Smart Office Berbasis Internet Of Things," in *Prosiding Semnastek*, 2017, no. November, pp. 1–7.

[14] Abadi, Iwan, and Arief Ginanjar. "Implementasi Analisis Sistem Online Kawasan Keselamatan Operasinal Penerbangan Menggunakan Web Service Berbasis NodeJS dan Android." *Jurnal ICT: Information Communication & Technology* 20.1 (2021): 1-8.

[15] Ginanjar, Arief, and Kusmaya Kusmaya. "Reliability Comparison of High Performance Computing between Single Thread Loop and Multiple Thread Loop using Java-Based Programming at Fingerprint Data Processing." *JURNAL SISFOTEK GLOBAL* 12.1 (2022): 11-17.

[16] B. Schwartz, P. Zaitsev, and V. Tkachenko, *High Performance Mysql: Optimization, Backups and Replication*, 2nd Edition. California: O'Reilly Media, Inc, 2012.

[17] K. Christiono and H. Sama, "Studi Komparasi Database Management System Antara Maria DB Dan Postgresql Terhadap Efisiensi Penggunaan Sumber Daya Komputer," *Conf. Business, Soc. Sci. Innov. Technol.*, vol. 1, no. 1, pp. 573–579, 2020.