Jurnal Teknik Informatika (JUTIF)
Vol. 4, No. 1, Februari 2023, hlm. 25-32

DOI: https://doi.org/10.20884/1.jutif.2023.4.1.629
p-ISSN: 2723-3863
e-ISSN: 2723-3871

COMPARATION OF DISTRIBUTED DATABASE MODEL BY CLUSTERING METHOD IN E-GOVERNMENT SYSTEM. STUDY AT KEMENKEU RI

Adinda Krida Wicaksono[*1], Fauzie Nurrakhman[2], Samidi[3]

[1,2,3]Magister Ilmu Komputer, Fakultas Teknologi Informasi, Universitas Budi Luhur, Indonesia
Email: [1]2111601874@student.budiluhur.ac.id, [2]2111601940@student.budiluhur.ac.id, [3]samidi@budiluhur.ac.id

(Naskah masuk: 05 Oktober 2022, Revisi : 03 November 2022, diterbitkan: 10 Februari 2023)

Abstract

MySQL is one of the most popular open source database in the world based on its performance, its reliability, and its easiness that has been proved. MySQL is categorized into community version and commercial version. The commercial version has standalone architecture models and clusters.Database Clustering is one of distributed database model. The case study used is Kemenkeu Learning Center (KLC) Application which is an e-government application in Ministry of Finance. KLC application used for interactive learning media for all ASN especially ASN in Ministry of Finance. This makes the KLC application being categorized into a very high critical application, so it takes a good response time and has high availability. One of the most important factors to fulfill this need is by deciding the appropriate MySQL database architecture for KLC application. This research was conducted by creating 3 (three) database architecture models according to best practice from MySQL, namely the Standalone architecture model, InnoDB Cluster and NDB Cluster. Then tested with several different scenarios to get results in the form of response time and percentage of errors that indicate high availability capabilities. The tools used in testing are Apache JMeter. From the test results, it is found that the MySQL NDB Cluster is the right database architecture model in an effort to realize the need for database services with the best response time and high availability capabilities.

Keywords: *Apache Jmeter, Clustering Database, Database, Distributed Database, InnoDB Cluster, MySQL, NDB cluster, Standalone Database.*

PENGUJIAN MODEL DISTRIBUTED DATABASE DENGAN METODE CLUSTERING PADA SISTEM E-GOVERNMENT. STUDI PADA KEMENKEU RI

Abstrak

MySQL adalah salah satu *database open source* paling populer di dunia. Dengan kinerja, keandalan, dan kemudahan penggunaan yang telah terbukti. MySQL terbagi atas versi *community* dan *commercial*. Versi *commercial* memiliki model arsitektur *standalone* dan *cluster*. *Database Clustering* merupakan salah satu bentuk model *distributed database*. Studi kasus yang digunakan adalah Aplikasi Kemenkeu Learning Centre (KLC) yang merupakan salah satu aplikasi *e-government* di Kementerian Keuangan. Aplikasi KLC ini digunakan sebagai media pembelajaran interaktif jarak jauh bagi ASN di seluruh Indonesia khususnya pegawai Kementerian Keuangan. Hal ini menjadikan aplikasi KLC termasuk dalam kategori aplikasi dengan kritikalitas yang sangat tinggi sehingga dibutuhkan response time yang baik dan memiliki kemampuan *high availability*. Salah satu faktor penting dalam memenuhi kebutuhan tersebut adalah dengan menentukan model arsitektur *database* MySQL yang tepat untuk aplikasi KLC. Penelitian ini dilakukan dengan membuat 3 (tiga) model arsitektur database sesuai *best practice* dari MySQL, yakni model arsitektur *Standalone*, InnoDB *Cluster* dan NDB *Cluster*. Kemudian dilakukan pengujian dengan beberapa skenario berbeda untuk mendapatkan hasil berupa *response time* dan persentase *error* yang menunjukkan kemampuan *high availability*. *Tools* yang digunakan dalam melakukan pengujian adalah Apache JMeter. Dari hasil pengujian, diperoleh bahwa MySQL NDB *Cluster* adalah model arsitektur *database* yang tepat dalam upaya mewujudkan kebutuhan layanan *database* dengan *response time* dan kemampuan *high availability* terbaik.

Kata kunci: *Apache Jmeter, Database, Database Clustering, Distributed Database, InnoDB Cluster, NDB Cluster, Standalone Database.*

1. PENDAHULUAN

Data memiliki peranan yang sangat penting dalam proses bisnis modern. Dengan adanya transformasi digital, data terus bertumbuh, dan semakin banyak data yang perlu disimpan dan diakses. Tren pertumbuhan data ini tidak terlepas dari pemanfaatan teknologi di berbagai aspek dan sektor, salah satunya adalah di pemerintahan. *E-government* merupakan penggunaan dan pemanfaatan teknologi informasi oleh pemerintah agar tercipta komunikasi antara pemerintah, masyarakat, dunia bisnis dan pihak-pihak lain yang berkepentingan untuk memberikan pelayanan secara cepat dan tepat [1].

Sejalan dengan hal tersebut, di era kemudahan akses informasi dan komunikasi saat ini, dituntut juga ketersediaan aplikasi yang dapat diakses secara terus menerus. Tidak hanya pemilik proses bisnis yang menentukan kritikalitas suatu layanan, bahkan pengguna biasa pun seringkali memiliki harapan layanan akan selalu tersedia. Hal ini akhirnya berujung pada kebutuhan akses data yang dapat beroperasi secara terus menerus dan andal. Jika tidak direncanakan dan dikelola dengan baik maka saat terjadi gangguan atau kegagalan dapat berpotensi pada terhentinya layanan. Sehingga waktu henti menjadi perhatian khusus karena terhentinya suatu layanan dapat menimbulkan kerugian bagi organisasi.

Kementerian Keuangan mulai mengintegrasikan inisiatif transformasi ke dalam konteks yang lebih modern dengan menerapkan aspek digitalisasi sebagai respon terhadap perubahan zaman, dan pesatnya perkembangan teknologi informasi dan komunikasi. Beberapa contoh *e-government* di internal Kemenkeu antara lain *Activity-Based Workplace* (ABW), *Flexible Working Place* (FWS), e-Kemenkeu (*office automation*), Kemenkeu e-Learning Center (KLC). Terdapat pula pelayanan digital Kemenkeu lainnya yang diperuntukkan bagi masyarakat dan *stakeholder* ekternal lainnya. Dalam keberlangsungannya tentu dibutuhkan infrastruktur dan model arsitektur yang tepat agar layanan yang ada dapat selalu tersedia dan dapat diandalkan. Salah satu area yang menjadi fokus untuk mewujudkan keberlangsungan layanan tersebut yakni area *database*.

Database didefinisikan sebagai pengumpulan data yang terorganisir. selain itu, basis data juga dapat merujuk ke sistem datastore atau juga disebut kumpulan dataset yang disimpan dalam sistem datastore [2]. Kondisi saat ini Kementerian Keuangan mengelola lebih dari 603 *database*, dimana dari jumlah tersebut sebanyak 326 *database* menggunakan *database* MySQL. Hal ini menunjukkan penggunaan *database* MySQL yang dominan dalam sistem Kemenkeu. Dengan populernya penggunaan *database* MySQL dan semakin banyaknya kebutuhan organisasi dalam membuat sistem baru, maka diperlukan solusi untuk memaksimalkan penggunaan *database* MySQL tersebut.

Database clustering muncul sebagai salah satu solusi yang dapat digunakan untuk menjamin ketersediaan layanan. Database clustering merupakan salah satu bentuk model *distributed database* yang merupakan teknologi untuk menyimpan dan membaca data dari beberapa lokasi dengan tetap menjaga ketergantungan dan keterjangkauan data [3]. *Database clustering* mengacu pada kemampuan mengkombinasikan beberapa *server* atau *instance* untuk terhubung ke *database* [4]. *Instance* adalah kumpulan *memory* dan proses yang berinteraksi langsung dengan *database*. *Database clustering* menawarkan beberapa keuntungan utama yakni yang pertama adalah toleransi kegagalan (*fault tolerance*) karena menyediakan lebih dari satu *server* atau *instance* bagi pengguna untuk terhubung sehingga meyediakan alternatif jika terjadi kegagalan pada salah satu *server*. Kemudian keutamaan yang kedua adalah pembagian beban/*load balancing*, karena fitur *database cluster* yang memungkinkan pengguna secara otomatis teralokasi ke *server* yang memiliki beban lebih rendah.

Berdasarkan pada penjelasan pada bagian pendahuluan dan latar belakang diatas, rumusan masalah yang dapat diambil sebagai berikut: Bagaimana hasil pengujian yang dilakukan dan perbandingan hasil atas masing-masing model arsitektur MySQL dan model arsitektur MySQL mana yang sesuai dengan kebutuhan.

Mengingat luasnya bahasan mengenai model *database clustering* dan ketersediaan produknya yang beragam, pada penelitian ini hanya membahas 3 model arsitektur *database* dari MySQL yakni Standalone, InnoDB *Cluster* dan NDB *Cluster*. Objek penelitian yang digunakan adalah *database* aplikasi Kemenkeu e-Learning Center (KLC). Parameter yang digunakan adalah response time dan persentase error.

Tujuan dari penelitian ini adalah menguji beberapa model arsitektur *database* MySQL agar mendapatkan hasil yang dapat menggambarkan perbedaan performa dari beberapa model arsitektur *database* tersebut. Manfaat dari peneliltian ini yakni dapat membantu organisasi dalam memilih model arsitektur *database* yang tepat dalam upaya mewujudkan kebutuhan layanan database dengan response time dan kemampuan high availability terbaik.

2. METODE PENELITIAN

2.1. Tinjauan Studi

High Availaiblity (HA) mengacu pada kemampuan *server* atau service untuk dapat tersedia secara terus-menerus. Konsep HA ini dapat dicapai dengan penerapan beberapan prinsip, yakni: menghilangkan titik tunggal kegagalan (*single point of failure*), menambahkan pemulihan melalui redundancy, dan mengimplementasikan *fault torelance* [5]. *Fault tolerant* merupakan kemampuan untuk melakukan pemulihan dari kondisi fault atau

eror yang terdeteksi didalam grup tanpa kehilangan data atau fungsionalitas. *Redundancy* berarti memiliki dua atau lebih komponen yang menjalankan perang yang sama dalam sistem. *Failover* merupakan sebuah peristiwa yang memungkinkan dalam suatu grup untuk melakukan pemulihan dari kesalahan yang terjadi pada komponen *primary*, secara otomatis memilih *primary* yang baru.

MySQL merupakan salah satu produk open source *database* system yang paling popular. *Open source* berarti siapapun dapat menggunakannya untuk berbagai macam kebutuhan secara gratis. Kemudian tersedia di banyak platform yang membuatnya mudah untuk didapatkan dan diinstall [6]. MySQL juga merupakan sistem manajemen *database* relasional (RDBMS) berbasis *structured query language* (SQL) dengan model arsitektur *client-server*.

Instance merupakan *server* MySQL yang sedang berjalan, biasanya merujuk pada satu atau lebih *server* MySQL yang berjalan pada mesin yang sama [5]. Hal ini berbeda dengan *server* MySQL yang merujuk pada suatu set *hardware* dan eksekusi MySQL. Sedangkan *cluster* pada MySQL merupakan suatu kumpulan *node* atau *server* yang didalamnya terdiri dari *node* data, *node management*, dan *node* API [7].

Ada tiga model arsitektur yang dilakukan pengujian pada penelitian ini yaitu model arsitektur MySQL Standalone, model arsitektur MySQL InnoDB Cluster dan model arsitektur MySQL NDB Cluster.

2.2. Penelitian Terdahulu

Penelitian terdahulu yang dilakukan oleh Samidi, Fadly, Yusuf Virmansyah, Ronal Yulyanto Suladi dan Ario Bambang Lesmana [8] membahas tentang "Optimasi *Database* dengan Metode *Index* dan Partisi Tabel *Database* Postgresql pada Aplikasi *E-Commerce* Studi pada Aplikasi Tokopintar". Dalam penelitian tersebut membahas bahwa optimasi *database* dengan metode *index* dan *partition table*. Pengujian dilakukan dengan membandingkan waktu *response time query* dengan menggunakan beberapa model data *table*.

Penelitian terkait *clustering* dari Afirda Desember Riawati, M Irfan, Khaeruddin dan Amrul Faruq [9] membahas tentang "*High Availability Dynamic Sharding Database Server* Dengan Metode *Fail Over* Dan *Clustering*". Dalam penelitian tersebut membahas teknik *clustering database* pada *database* mongoDB dimana sistem *sharding database* berbasis noSQL menggunakan mongoDB berhasil diimplementasikan pada *server* yang diset dengan *High Avaibility*.

Adapun Suyud Widiono [10] membahas tentang "*Experiments And Descriptive Analysis In The Mariadb Database Cluster System To Prepare Data Availability*". Penelitian membahas desain dan implementasikan sistem *cluster* pada *database* MariaDB.

Mahendra Data, Gilang Ramadhan dan Kasyful Amron [11] membahas tentang "Analisis Availabilitas dan Reabilitas Multi-Master *Database Server* dengan *State Snapshot Transfers* (SST) Jenis RSync Pada MariaDB Galera Cluster". Penelitian tersebut membahas desain *cluster database* pada *database* MariaDB.

Ravie Kurnia Laday, Heru Sukoco dan Yani Nurhadryani [12] membahas tentang "*Distributed System and Multimaster Replication Model on Reliability Optimation Database*". Penelitian ini tersebut membahas tentang perancangan *database* MySQL dengan metode *multi master replication* yang menghasilkan dapat menjadi solusi untuk menjaga ketersediaan data selama koneksi antar *server* terjaga.

Hayat Z. dan Soomro T.R. [13] dalam penelitian "*Implementation of oracle real application cluster*" membahas tentang implementasi teknologi *clustering* dari RDBMS Oracle yang mampu memberikan kemampuan *High Availability* untuk menjamin kestabilan dalam proses pembacaan atau akses data bahkan ketika terjadi kegagalan pada *node server* anggota *cluster* serta kemampuan skalabilitas naik dan turun tergantung dari proses bisnis.

Intan Putri Andhikha [14] telah melakukan penelitian "Implementasi Dan Analisis Kinerja Mysql Cluster Menggunakan Metode *Load Balancing*" yang membahas tentang fungsionalitas MySQL Cluster yang berhasil bekerja dengan baik untuk fungsi high availability dan performa dengan mengkolaborasikan dengan sistem lain yang mendukung kinerja dari *clustering database* pada RDBMS MySQL yaitu menggunakan *Load Balancer*.

Juanda Hakim Lubis [15] dalam penelitian "Analisa Performansi Query Pada Database Smell" melakukan pengujian tentang optimasi *database* dengan menggunakan partisi. Dimana hasilnya memiliki waktu dan cost pemrosesan *query* atau *response time query* yang lebih baik.

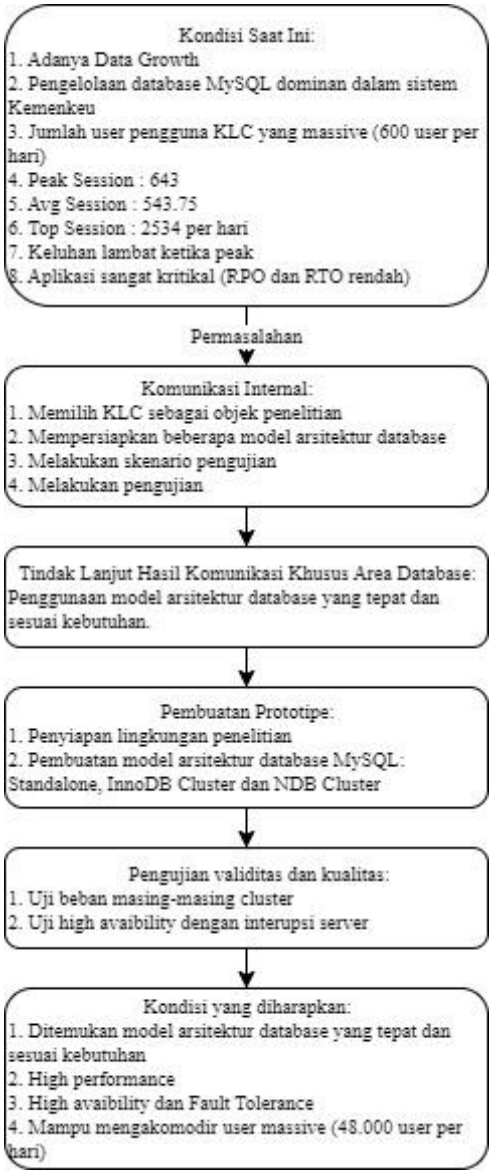
Kai Xiang [16] dalam penelitian "*An improvement in MySQL cluster in e-commerce scenarios*" membahas tentang pengujian sistem MEC dalam menguji kemampuan *high availability* dan *high concurrency* pada arsitektur yang menggunakan gabungan dari MyCAT, Percona XtraDB dan replikasi MySQL. Dimana dalam metode ini operasi pembacaan dan penulisan data dapat dipisah ke dalam *node server database* yang berbeda. Pada penelitian tersebut digunakan dua *node server* MyCat, tiga *node server* Percona XtraDB Cluster serta tujuh *node server* MySQL *Replication* atau MySQL *Single Node*.

Pada hasil penelitian sebelumnya telah mencoba optimasi *database* dengan menganalisa respon time query dan juga teknik *high avaibility* pada beberapa produk *database*. Selain itu, *node database* yang digunakan pada rancangan skenario penelitian

sebelumya yakni sejumlah 3 *node* atau sejumlah minimum rekomendasi dalam rancangan *cluster*. Dalam penelitian ini peneliti mencoba melakukan eksperimen yang berbeda dibandingkan penelitan sebelumnya, dimana peneliti mencoba membandingkan menguji beberapa model arsitektur *database* MySQL dengan jumlah *node database* yang lebih luas, yakni paling sedikit 1 (satu) *node* dan paling banyak 6 (enam *node*) *database* agar mendapatkan hasil yang lebih lengkap dan dapat menggambarkan perbedaan performa dari beberapa model arsitektur *database* tersebut dalam upaya mewujudkan layanan *database* yang andal dengan ketersediaan yang tinggi.

2.3. Kerangka Konsep Penelitian

IBerdasarkan pemaparan di atas, maka kerangka konsep penelitian yang penulis lakukan adalah sebagai berikut.



Gambar 1 Kerangka Konsep Penelitian.

2.4. Metode Penelitian

Dalam penelitian ini digunakan metode eksperimental dimana *database* yang digunakan adalah *database* KLC dengan ukuran sebesar 6.8 GB. Data yang dijadikan penelitan adalah data per tanggal 1 Mei 2022. Pengujian dilakukan dengan mengambil hasil eksekusi kueri dan *response time* menggunakan *tools* Apache JMeter.

```
+---------+--------------+
| DB Name | DB Size in GB |
+---------+--------------+
| klc     |          6.8 |
+---------+--------------+
1 row in set (0.00 sec)
```
Gambar 2 Ukuran Database Penelitian

Adapun langkah-langkah mulai dari persiapan lingkungan penelitian sampai dengan pengujian adalah sebagai berikut:

a. Instalasi sistem operasi *server* virtual untuk masing-masing model arsitektur *database*
b. Instalasi *database* pada masing-masing model arsitektur *database*
c. Restore *database* dari hasil backup
d. Konfigurasi Apache JMeter
e. Pengujian dilakukan dengan 2 skenario sebagai berikut:
    1. Pengujian beban menggunakan query select dan insert tanpa interupsi server.
    2. Pengujian beban menggunakan query select dan insert dengan interupsi server.

Kemudian pengujian kedua dilakukan dengan metode yang sama dengan pengujian pertama tetapi dikombinasikan dengan interupsi server dengan mematikan service database MySQL. Pengujian interupsi server ini dilakukan untuk mengukur persentase error yang menunjukkan kemampuan high availability.

3. HASIL DAN PEMBAHASAN

3.1. Persiapan Lingkungan

Lingkungan yang digunakan berupa *server* virtual pada hyper-v. Semua *server virtual* berada dalam 1 (satu) *server* fisik/host dan 1 (satu) segmen *network* yang sama 10.242.93.0/24. Masing-masing *server* virtual memiliki spesifikasi sebagai berikut:

a. Sistem Operasi Oracle Linux 8.5
b. 2 Virtual Processor
c. 8192 MB Virtual Memory
d. 200 GB Disk

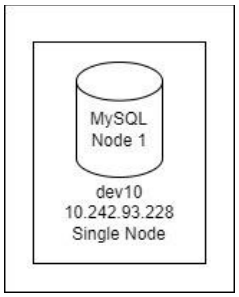
Konfigurasi Apache Jmeter menggunakan *File Installer* Apache Jmeter Version 5.4.3. Penelitian dilakukan dengan membuat *test plan* pada Apache Jmeter menggunakan *variable thread group* 1000 *user*/10 menit. Hasil pengujian diambil dari hasil *listener report*: *View Results Tree*, *Summary Report* dan *Response Time Graph*. Untuk *field* dengan nilai *auto increment* digunakan *Config Element Counter*, dengan nilai *increment* 1 (satu). Masing-masing

model arsitektur *database* dilakukan pengujian dengan *query insert* dan *select* yang sama.

Model arsitektur yang dipersiapkan mengacu pada 3 model best practice MySQL sebagai berikut:

1. Model Arsitektur MySQL Standalone

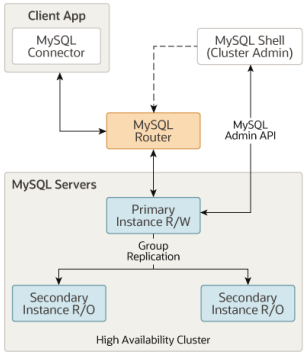
Best practice model arsitektur standalone menggunakan 1 (satu) server untuk mengelola instance database MySQL. Lingkungan yang dipersiapkan adalah dengan 1 (satu) server seperti yang ditunjukkan pada Gambar 3. *File Installer* yang digunakan adalah MySQL Commercial Server 8.0.27 RPM for Oracle Linux / RHEL 8 x86 (64bit), *Database* version: 8.0.27-commercial, *Server* version: MySQL Enterprise Server – Commercial.



Gambar 3 Topologi Pengujian MySQL Standalone

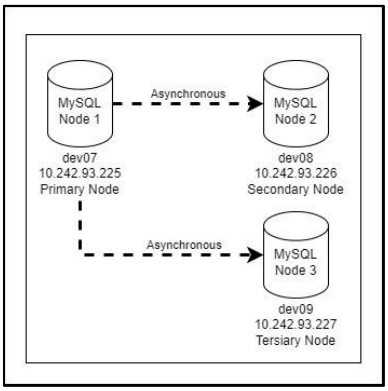
2. Model Arsitektur MySQL InnoDB Cluster

Best practice model arsitektur InnoDB Cluster menyediakan solusi *high availability* untuk MySQL dengan menggunakan adminAPI, yang sudah termasuk didalam MySQL *shell*, berisi paling sedikit 3 (tiga) MySQL *server instance* yang berfungsi sebagai InnoDB *cluster* sesuai yang ditunjukkan pada Gambar 4.



Gambar 4 Best Practice Topologi MySQL InnoDB *Cluster*

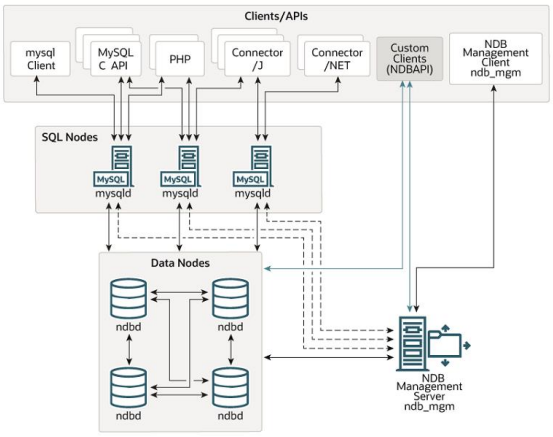
Lingkungan yang dipersiapkan adalah 3 (tiga) server virtual terdiri atas 1 server sebagai Primary Node dan 2 server sebagai Secondary Node seperti yang ditunjukkan pada Gambar 5. File Installer yang digunakan adalah MySQL Commercial Server 8.0.27 RPM for Oracle Linux / RHEL 8 x86 (64bit), MySQL Shell 8.0.27 RPM for Oracle Linux / RHEL 8 x86 (64bit), Database version: 8.0.27-commercial , Server version: MySQL Enterprise Server – Commercial



Gambar 5 Topologi Pengujian MySQL InnoDB Cluster

3. Model Arsitektur MySQL NDB Cluster

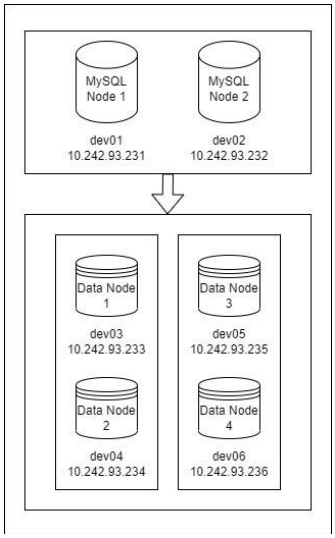
Best practice model arsitektur NDB *cluster* menawarkan fitur *high-availability* dan *data-persistence* yang dapat dikonfigurasi dengan berbagai opsi *failover* dan *load-balance* yang berisi satu set data lengkap sesuai yang ditunjukkan pada Gambar 6.



Gambar 6 Best Practice Topologi MySQL NDB Cluster

Syarat konfigurasi sebuah *cluster* adalah minimal 3 (tiga) *node*, yang masing-masing berfungsi sebagai *Node Management* untuk mengelola *node* lain dalam NDB Cluster, *Node* Data sebagai *node* untuk menyimpan data *cluster*, dan SQL *Node* yang berperan untuk mengakses data *cluster* [8].

Lingkungan yang dipersiapkan adalah Cluster 6 *server* virtual terdiri atas 2 *server* virtual sebagai Management dan MySQL Node serta 4 *server* virtual sebagai Data Node seperti yang ditunjukkan pada Gambar 7. File instaler yang digunakan adalah MySQL Cluster 8.0.27 RPM for Oracle Linux / RHEL 8 x86 (64bit), Database version : 8.0.27-*cluster*, Server version: MySQL Cluster Enterprise Server – Commercial



Gambar 7 Topologi Pengujian MySQL NDB Cluster.

3.2. Pelaksanaan Pengujian

Pengujian skenario pertama dilakukan dengan menjalankan *query insert* dan *select* tanpa interupsi server, kemudian membandingkan *response time* pada tiap model arsitektur *database* Standalone, InnoDB Cluster, dan NDB Cluster. Pengujian query select dan insert ini dilakukan untuk mengukur response time dari masing-masing model arsitektur database.

Berdasarkan hasil pengujian skenario pertama dengan query select tanpa melakukan interupsi server ditunjukkan pada Tabel 1, didapatkan hasil response time Min terbaik dapat dicapai dengan model arsitektur NDB Cluster dengan nilai 65ms, sementara hasil response time Max terbaik didapat pada arsitektur NDB Cluster dengan nilai 871ms, dan hasil response time rata-rata terbaik didapatkan pada model arsitektur NDB Cluster dengan nilai 146ms.

Tabel 1 Pengujian Beban *Insert*

Arsitektur	Response Time (dalam ms)			
	Min	Max	Avg	Error %
Standalone	67	990	150	0.00%
InnoDB Cluster	66	929	160	0.00%
NDB Cluster	**65**	**871**	**146**	0.00%

Berdasarkan hasil pengujian skenario pertama dengan menjalankan query insert tanpa melakukan interupsi server ditunjukkan pada Tabel 2, didapatkan hasil response time Min terbaik dapat dicapai dengan model arsitektur NDB Cluster dengan nilai 65ms, sementara hasil response time Max terbaik didapat pada arsitektur standalone dengan nilai 903ms, dan hasil response time rata-rata terbaik didapatkan pada model arsitektur InnoDB Cluster dengan nilai 148ms.

Tabel 2 Pengujian Beban *Select*

Arsitektur	Response Time (dalam ms)			
	Min	Max	Avg	Error %
Standalone	69	**903**	153	0.00%
InnoDB Cluster	69	1073	**148**	0.00%
NDB Cluster	**65**	1059	152	0.00%

Pengujian skenario kedua dilakukan dengan menjalankan *query insert* dan *select* disertai interupsi server dengan mematikan service database MySQL seperti pada Tabel 3 berikut:

Tabel 3 Skenario Interupsi Server

Menit ke-	Standalone	InnoDB Cluster	NDB Cluster
0	Server 1 ON	Server 1 ON Server 2 ON Server 3 ON	Server 1 ON Server 2 ON
1	Server 1 **OFF**	Server 1 **OFF** Server 2 ON Server 3 ON	Server 1 **OFF** Server 2 ON
2	Server 1 **OFF**	Server 1 **OFF** Server 2 ON Server 3 ON	Server 1 **OFF** Server 2 ON
3	Server 1 ON	Server 1 ON Server 2 ON Server 3 ON	Server 1 ON Server 2 ON
4	Server 1 ON	Server 1 ON Server 2 ON Server 3 ON	Server 1 ON Server 2 ON
5	Server 1 ON	Server 1 ON Server 2 **OFF** Server 3 ON	Server 1 ON Server 2 **OFF**
6	Server 1 ON	Server 1 ON Server 2 **OFF** Server 3 ON	Server 1 ON Server 2 **OFF**
7	Server 1 ON	Server 1 ON Server 2 ON Server 3 ON	Server 1 ON Server 2 ON
8	Server 1 ON	Server 1 ON Server 2 ON Server 3 ON	Server 1 ON Server 2 ON
9	Server 1 ON	Server 1 ON Server 2 ON Server 3 ON	Server 1 ON Server 2 ON
10	Server 1 ON	Server 1 ON Server 2 ON Server 3 ON	Server 1 ON Server 2 ON

Kemudian membandingkan *persentase error* pada tiap model arsitektur *database* Standalone, InnoDB Cluster, dan NDB Cluster. Pengujian disertai interupsi server ini dilakukan untuk mengukur kemampuan high availability dari masing-masing model arsitektur database.

Berdasarkan hasil pengujian skenario kedua dengan eksekusi query select disertai interupsi server ditunjukkan pada Tabel 4, hasil persentase error terendah didapatkan pada model arsitektur NDB Cluster dengan nilai 0.10%, diikuti dengan model arsitektur InnoDB Cluster dengan nilai 5.70%, dan Standalone dengan nilai 20.30%.

Tabel 4 Pengujian query select dengan interupsi server

Arsitektur	Error % (timeout)
Standalone	20.30%
InnoDB Cluster	5.70%
NDB Cluster	**0.10%**

Sedangkan hasil pengujian skenario kedua dengan eksekusi query insert disertai interupsi server ditunjukkan pada Tabel 5, hasil persentase error terendah didapatkan pada model arsitektur NDB Cluster dengan nilai 0.00%, diikuti dengan model

arsitektur InnoDB Cluster dengan nilai 1.50%, dan Standalone dengan nilai 19.60%.

Tabel 5 Pengujian query insert dengan interupsi server

Arsitektur	Persentase Error (%)
Standalone	19.60%
InnoDB Cluster	1.50%
NDB Cluster	**0.00%**

4. DISKUSI

Hasil observasi pengujian pertama yakni pengujian dengan menjalankan *query insert* dan *select* tanpa melakukan interupsi pada *server*, didapatkan hasil *response time* yang mencerminkan performa ketiga model arsitektur tersebut, dimana hasil pengujian pada model arsitektur NDB Cluster secara keseluruhan didapatkan hasil *response time* terbaik yang dibuktikan dengan response time Min tercepat pada pengujian query insert, dan reponse time Min tercepat, Max terendah, dan Avg terendah pada pengujian query select.

Sementara itu hasil observasi pengujian kedua yakni pengujian dengan menjalankan *query insert* dan *select*, dengan melakukan interupsi pada *server*, didapatkan pula hasil yang konsisten, dimana hasil pengujian pada model arsitektur NDB Cluster secara keseluruhan didapatkan hasil dengan persentase error terendah baik pada pengujian select maupun insert. Hal ini menunjukkan bahwa model arsitektur NDB Cluster memiliki kemampuan *high availability* terbaik, sehingga layanan dapat tersedia secara terus menerus dengan potensi interupsi yang minim.

5. KESIMPULAN

Berdasarkan hasil penelitian yang dilakukan pada studi kasus *database* KLC, dapat disimpulkan bahwa NDB Cluster adalah arsitektur database yang tepat dalam upaya mewujudkan kebutuhan layanan database dengan response time dan kemampuan high availability terbaik pada system e-government.

Dalam penelitian ini masih didapatkan banyak hal yang dapat dikembangkan lagi, saran dari tim peneliti kepada peneliti berikutnya yang ingin melanjutkan penelitian ini adalah melakukan penelitian terkait penggunaan *load balance* pada *cluster database* MySQL dan melakukan optimasi *query* pada *database* KLC.

DAFTAR PUSTAKA

[1] H. Atthahara, "Inovasi pelayanan publik berbasis e-government: studi kasus aplikasi Ogan Lopian Dinas Komunikasi dan Informatika di Kabupaten Purwakarta," *Jurnal Politikom Indonesiana*, vol. 3, no. 1, pp. 66-66, 2018.

[2] A. Fadli, M. I. Zulfa, A. W. W. Nugraha, A. Taryana, and M. S. Aliim, "Analisis Perbandingan Unjuk Kerja Database SQL dan Database NoSQL Untuk Mendukung Era Big Data," *Jurnal Nasional Teknik Elektro*, pp. 154-158, 2020.

[3] M. S. Rana, M. K. Sohel, and M. S. Arman, "Distributed Database Problems Approaches and Solutions-A Study," *International Journal of Machine Learning and Computing (IJMLC)*, 2018.

[4] A. Misbullah, N. Nazaruddin, R. Rasudin, Z. Zulfan, "ANALISA PROSES MIGRASI MYSQL NON-CLUSTER KE CLUSTER DALAM MENANGANI FAIL-OVER SISTEM AKADEMIK UNIVERSITAS SYIAH KUALA,"*Cyberspace: Jurnal Pendidikan Teknologi Informasi*, vol. 4, no. 1, pp. 40-49, 2020.

[5] C. Bell, *Introducing the MySQL 8 Document Store*, 2018.

[6] C. Bell, *Introducing InnoDB Cluster*, 2018.

[7] W. Krogh, and M. Okuno, *Pro MySQL NDB Cluster*, 2017.

[8] S. Samidi, F. Fadly, Y. Virmansyah, R. Y. Suladi, and A. B. Lesmana, "Optimasi Database dengan Metode Index dan Partisi Tabel Database Postgresql pada Aplikasi E-Commerce. Studi pada Aplikasi Tokopintar," *Jurnal Pendidikan Tambusai*, vol. 6, no. 1, pp. 2094-2102, 2022.

[9] A. D. Riawati, M. Irfan, K. Khaeruddin, and A. Faruq, "High Availability Dynamic Sharding Database Server Dengan Metode Fail Over Dan Clustering," *Jurnal Manajemen Informatika dan Sistem Informasi*, vol. 5, no. 1, pp. 1-10, 2022.

[10] 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.

[11] G. Ramadhan, and K. Amron, "Analisis Availabilitas dan Reliabilitas Multi-Master Database Server Dengan State Snapshot Transfers (SST) Jenis Rsync Pada MariaDB Galera Cluster," *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN*, *2355*, 7699, 2017.

[12] R. K. Laday, H.Sukoco, and Y. Nurhadryani, "Distributed System and Multimaster Replication Model on Reliability Optimation Database," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 13, no. 3, pp. 529-536, 2015.

[13] Z. Hayat, and T. R. Soomro, "Implementation of oracle real application *cluster*", *Review of Computer Engineering Studies*, vol. 3, no. 4, pp. 81-85, 2016.

[14] I. P. Andhikha, "Implementasi Dan Analisis

Kinerja Mysql Cluster Menggunakan Metode Load Balancing," *Jurnal Manajemen Informatika*, vol. 8, no. 1, 2017.

[15] J. H. Lubis, "Analisa Performansi Query Pada Database Smell," *Jurnal Mantik Penusa*, vol. 21, no. 1, 2017.

[16] K. Xiang, "An improvement in MySQL *cluster* in e-commerce scenarios," In *Proceedings of 2nd International Conference on Information Technology and Electronic Commerce*, IEEE, pp. 286-289, 2014.