# PERANCANGAN CLUSTERING DATABASE SERVER UNTUK MENINGKATKAN UNJUK KERJA SERVER DAN MENJAMIN KETERSEDIAAN LAYANAN

**Sugiyatno**
*Teknik Informatika; Universitas Bhayangkara Jakarta Raya; Jl. Raya Perjuangan, Marga Mulya, Bekasi Utara, Marga Mulya, Bekasi Utara, Kota Bks, Jawa Barat 17121 (021) 88955882; e-mail: sugiyatno@dsn.ubharajaya.ac.id.*
* Korespondensi: e-mail: sugiyatno@dsn.ubharajaya.ac.id

**ABSTRAKS**

*Dalam mengelola infrastruktur Teknologi Informasi (TI) dan menyediakan berbagai layanan terkait TI untuk mendukung operasional organisasi atau perusahaan membutuhkan infrastruktur yang dapat menunjang. Selain infrastruktur jaringan computer di butuhkan infrastruktur pada database server. Kegagalan suatu devices pada sebuah server bisa terjadi kapan saja, jika sebuah database server mati yang disebabkan oleh suatu hal (missal: power supply mati, kebakaran atau yang lainnya), maka pengguna tidak bisa mengakses aplikasi. selain resiko kehilangan data yang ada pada server tersebut. Untuk mengatasi masalah tersebut salah satu teknik Clustering Database menggunakan MySQL Cluster. Metode clustering sangat baik untuk menjamin ketersediaan layanan yang tinggi (High-Availability) dan penanganan kegagalan sistem (failure). Jika ada salah satu server yang mengalami kegagalan/failure maka sistem tidak akan langsung terganggu karena server lain akan tetap berfungsi dan menggantikan kerja server utama. Kemampuan clustering memungkinkan sebuah database tetap hidup dalam waktu yang lama.*

*Kata Kunci: Database server, Clustering Database, MySQL Cluster*

## 1. PENDAHULUAN

### 1.1 Latarbelakang

Pada sebuah organisasi atau perusahaan biasanya ada bagian yang tugasnya mengelola infrastruktur Teknologi Informasi (TI) dan menyediakan layanan terkait TI untuk mendukung aktivitas perusahaan

Untuk menjalankan tugasnya, membutuhkan infrastruktur yang handal seperti infrastruktur jaringan komputer, dan *database server*. *Database server* merupakan tempat untuk menampung semua aktifitas *client* atau *user*. D*atabase* bekerja menyediakan data untuk memperoleh data tanpa mengalami masalah.

Kegagalan *devices* pada *server* bisa terjadi kapan saja, sehingga sistem informasi terganggu, jika masih menggunakan *single database*. Dengan *single database*, jika *database server* mati, maka *client* tidak bisa mengakses sistem informasi dan resiko hilangnya data yang tersimpan dalam *server* tersebut. *Database* dituntut bekerja dengan cepat, mempunyai kehandalan dan ketersediaan layanan.

Teknik *Clustering Database* menggunakan *MySQL Cluster* merupakan kumpulan dari beberapa *server* yang berdiri sendiri yang kemudian bekerjasama sebagai suatu sistem tunggal.. Dengan *Clustering Database*, data dapat terbagi ke beberapa *server* secara otomatis secara *real time*, karena semua *server* dianggap satu kesatuan. Metode *clustering* sangat baik untuk menjamin ketersediaan layanan yang tinggi (*High-Availability*) dan penanganan kegagalan sistem (*failure*) karena kemampuan tiap *server* akan digunakan dan jika ada salah satu *server* yang mengalami kegagalan/*failure* maka sistem tidak akan langsung terganggu karena *server* lain akan tetap berfungsi dan menggantikan kerja *server* utama.

### 1.2 Database

*Database* atau basis data adalah kumpulan *file* / *table* yang saling berelasi (berhubungan) yang disimpan dalam media penyimpanan eletronik dan saling berkaitan sehingga memudahkan aktivitas untuk memperoleh informasi (Atzeni, 2003)
Tujuan utama pengelolaan data dalam *database* adalah agar kita dapat memperoleh data yang kita cari dengan mudah dan cepat. Pemanfaatan *database* dilakukan untuk memenuhi sejumlah tujuan seperti berikut ini:

1. Kecepatan dan kemudahan (*speed*)
2. Efisiensi ruang penyimpanan (*space*)
3. Keakuratan (*accuracy*)
4. Ketersediaan (*availability*)
5. Kelengkapan (*completeness*)
6. Keamanan (*security*)
7. Kebersamaan pemakaian (*sharability*)

Dalam penggunaanya, database memiliki beberapa keuntungan yaitu:

1. Mengurangi kesalahan yang disebabkan oleh faktor manusia. Tugas mekanis lebih baik dilaksanakan oleh mesin.
2. Komputer dapat mengambil dan mengubah data lebih cepat dari manusia.

3. Akurat dan informasi terbaru selalu tersedia setiap saat.
4. Menghemat ruangan karena tidak perlu menyediakan ruangan penyimpanan kertas file yang sangat banyak

### 1.3. *Database Management System*

Menurut (Arbie, 2003), *Database Management System (*DBMS*)* adalah sistem *software* yang memungkinkan pengguna untuk mendefinisikan, membuat, memelihara, dan kontrol akses ke database. *DBMS* adalah *software* yang berinteraksi dengan program aplikasi dan pengguna *database*. Inti dari DBMS sering disebut *database engine*. Mesin ini merespons perintah-perintah khusus untuk membuat struktur *database* kemudian membuat, membaca, memperbarui, dan menghapus record-record pada sebuah *database*. DBMS dapat dibeli dari sebuah vendor teknologi database seperti Oracle, IBM, Microsoft, atau Sybase .(Connolly & Begg, 2010)

### 1.4. *Server*

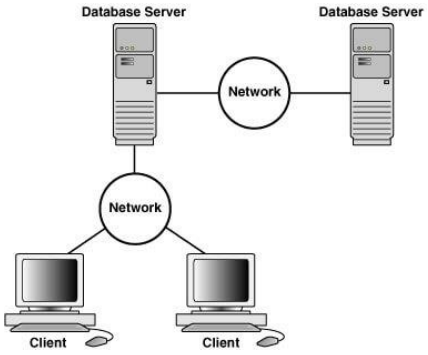
Menurut (O'Brien & James, 2005) lebih spesifik menyatakan bahwa, *Server* adalah komputer yang mendukung aplikasi dan telekomunikasi dalam jaringan, serta pembagian peralatan *software*, dan *database* di antara berbagai terminal kerja dalam jaringan. Fungsi *server* secara umum dilakukan oleh sebuah *server* komputer adalah sebagai berikut:

a. Menyimpan aplikasi dan *database* yang dibutuhkan oleh komputer yang terhubung.
b. Menyediakan fitur keamanan komputer.
c. Melindungi semua komputer terhubung menggunakan *firewall*.
d. Menyediakan *IP address* untuk mesin komputer terhubung.

### 1.5. *Database Server*

Menurut (Hodges, 2007) *Database server* adalah program komputer yang menyediakan layanan data lainnya ke komputer atau program computer lain, *Database server* digunakan untuk menyimpan data baik yang digunakan *client* secara langsung maupun data yang diproses oleh *server* aplikasi. Dalam *server database* tersebut, bisa berisi ratusan ataupun ribuan *database* dari banyak *user*. D*atabase* dikelompokkan atau disimpan per *user* yang memakai layanan *database* tersebut. Agar tidak terjadinya pencurian data. Selain itu *database server* dapat digunakan untuk beberapa kegiatan, seperti analisis data, penyimpanan data, pengarsipan dan lain-lain. Database server menyediakan beberapa manfaat, diantaranya :

a. Semua data untuk organisasi dapat disimpan di satu lokasi.
b. *Database server* menambahkan tingkat keamanan data.
c. Database server menyediakan layanan database management service dimana data disusun dengan cara tertentu sehingga meningkatkan pencarian dan pengambilan data.
d. Beberapa client dapat mengakses data yang disimpan di database server dalam satu waktu tanpa saling menggangu satu sama lain.




Gambar 1. *Database server*
Sumber (O'Brien & James, 2005)

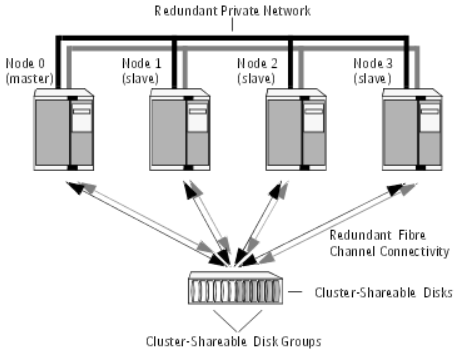
### 1.6. *Clustering Database*

Database clustering adalah kumpulan dari beberapa *server* yang berdiri sendiri yang kemudian bekerjasama sebagai suatu sistem tunggal (Hodges, 2007).. Pada *server database* yang besar dalam pelayanannya tidak menggunakan *server database* tunggal, tetapi dilayani oleh sekelompok *server database*, beberapa buah *server database* dihubungkan menjadi satu pada lingkungan yang sangat kompleks. Berbagai jenis *server* diikat menjadi satu untuk menjadikan suatu pelayanan tunggal (*one stop shopping*).
Contoh arsitektur dari *database cluster*:

*1. Shared Disk Clusters*

Arsitektur *shared disk clusters* menggunakan server-server *independent* dan berbagi sebuah sistem penyimpanan tunggal. Setiap *server* mempunyai prosesor dan memori sendiri, tetapi berbagi *disk resources*. Implementasi utama dari *shared-disk clustering* adalah bukan untuk *scalability*. *Shared-disk clustering* ini diimplementasikan untuk *availability* dan menambah *node* cadangan sebagai *failover node*.

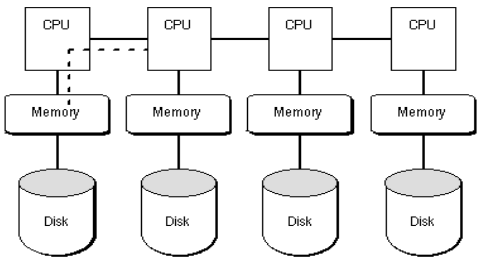
*P-ISSN:0216-9436*
*E-ISSN:2622-6782*




Gambar 2 *Shared Disk Clusters*
Sumber (Hodges, 2007)

2. *Shared Nothing Cluster*

Dalam arsitetur *shared nothing cluster*, tiap *server* dalam *cluster* menangani prosesor, memori, *storage*, *record locks* dan transaksi yang terpisah dan melakukan koordinasi dengan *server* lain melalui jaringan dengan menggunakan *high speed, low-latency interconnect technology*. Dalam proses permintaan data suatu *node* harus mengirimkan pesan ke *node* yang lain yang memiliki data yang diakses. Hal ini juga dilakukan saat koordinasi data yang dilakukan pada *node* yang lain seperti *insert*, *select*, *update* dan *delete*. Berbeda dengan *shared disk*, *shared nothing* didesain untuk *high availability* dan *scalability*.

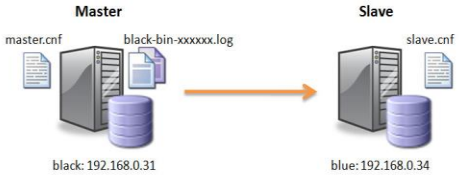



Gambar 3. *Shared nothing cluster*

Sumber : (Hodges, 2007)

**3.** Replikasi

Replikasi adalah suatu teknik untuk melakukan *copy* semua perubahan yang dibuat pada *server* (disebut *master*) ke *server* lain (disebut *slave*) untuk menghindari kehilangan data jika *master* mengalami *crash* dan memiliki salinan dari *server* utama. (Charless Bell, 2010).

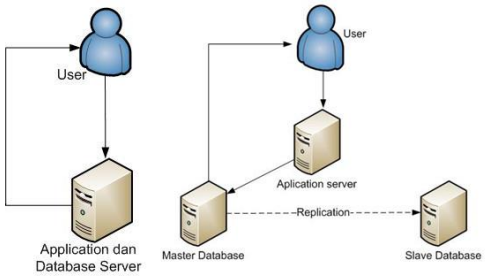



Gambar 4 Skema replikasi

Replikasi antara server di MySQL didasarkan pada mekanisme log biner (*binary log)*. Jika salah satu beroperasi sebagai master menulis pembaruan data dan pembaruan tersebut tercatat ke log biner. Informasi yang ada di log biner disimpan dalam format log yang berbeda sesuai dengan database perubahan yang dicatat. Kemudian *slave* dikonfigurasi untuk membaca log biner dari master dan melakukan perubahan log biner yang ada di *master* ke database pada *slave*

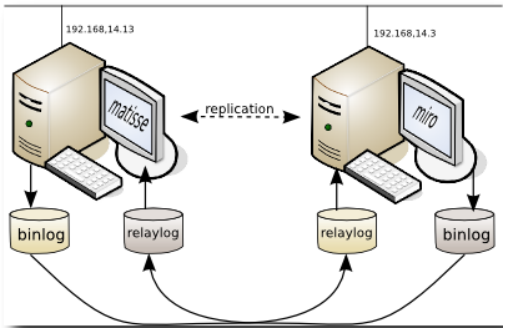
Terdapat beberapa jenis – jenis replikasi diantaranya adalah sebagai berikut:

*a. Snapshot replication*
*b. Transactional replication*
c. *Merge replication*




Gambar 5.Perbedaan skema *database server* biasa dengan replikasi
Sumber (Hodges, 2007)

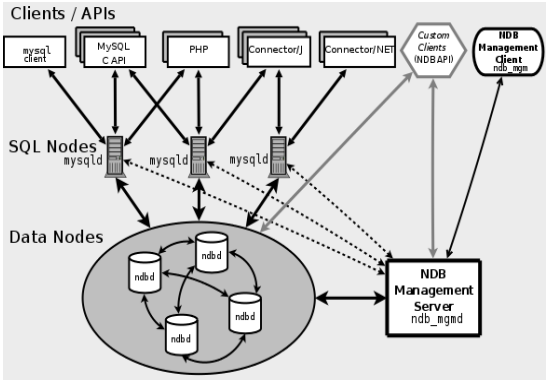
Replikasi master-master merupakan replikasi *mysql* dimana terdapat 2 (dua) server, dimana *server* pertama bertindak sebagai master dan juga *slave* begitu juga dengan *server* kedua sehingga terdapat dua master dan dua *slave*. Dua *server* tersebut akan melakukan sinkronisasi log biner dalam melakukan replikasi.




Gambar 6 Skema replikasi master – master
Sumber : (Hodges, 2007)

### 1.7. *MySQL Clustering*

Menurut (Raharjo & Budi, 2011) *MySQL Cluster* merupakan sebuah tipe basis data (*database*) yang dapat beroperasi dalam ukuran data yang besar. *MySQL Cluster* adalah sebuah teknologi baru untuk memungkinkan *clustering* di dalam *memory database* dalam sebuah sistem *share-nothing*. Arsitektur *share-nothing* mengijinkan sistem dapat bekerja dengan *hardware*/perangkat keras yang sangat murah, dan tidak membutuhkan perangkat keras dan lunak dengan spesifikasi khusus. Arsitektur tersebut juga handal karena masing-masing komponen mempunyai memory dan disk tersendiri. *MySQL cluster* menggabungkan *MySQL server* biasa dengan sebuah mesin penyimpanan in-memory ter-cluster yang dinamakan NDB. NDB berarti bagian dari suatu rangkaian yang dikhususkan sebagai mesin penyimpanan, sedangkan *MySQL cluster* diartikan sebagai kombinasi atau gabungan dari MySQL dan mesin penyimpanan yang baru tersebut.




Gambar 7. *Node* dalam *MySQL Cluster*
Sumber : (Raharjo & Budi, 2011)

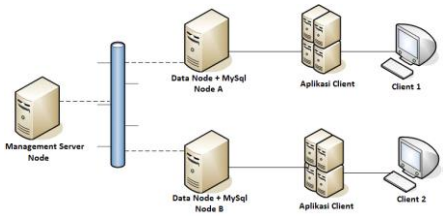
## 2. PEMBAHASAN

### 1. Tahap Desain Sistem

Perancangan *clustering database server* untuk replikasi data menggunakan *Mysql Cluster*. *MySQL Cluster* menggabungkan *MySQL Server* biasa dengan sebuah mesin penyimpanan in-memory tercluster yang dinamakan NDB. NDB berarti bagian dari suatu rangkaian yang dikhususkan sebagai mesin penyimpanan, sedangkan *MySQL Cluster* diartikan sebagai kombinasi atau gabungan dari *MySQL* dan mesin penyimpanan yang baru tersebut. Ada tiga *node* yang menyusun *MySQL Cluster*, yakni:

1. *Data Nodes*, digunakan untuk menyimpan semua data yang menjadi milik *MySQL Cluster*. Semua data direplikasi di node-node ini.
2. *Management Server Node*, digunakan untuk mengendalikan konfigurasi sistem ketika startup. Selain itu, node ini juga dapat digunakan sebagai pengidentifikasi setiap perubahan setting yang terjadi pada *cluster*.
3. *MySQL Server Node*, berfungsi sebagai pintu akses untuk masuk ke dalam node-node data yang ter-cluster.

*Management Server Node* dan *Data Node* dihubungkan tujuannya agar bisa terjadi replikasi data antara kedua *server* tersebut dan juga berfungsi untuk mengatasi kegagalan sistem *database* pada salah satu sisi *server* agar *server* yang lain bisa menggantikan tugas *server* yang lainnya. Agar perancangan itu bisa dilakukan dengan baik digunakan teknologi *MySQL Cluster*. Pada tahapan pembuatan sistem cluster dibagi tiga bagian utama yaitu *Management Server Node*, *Data Node* dan *MySQL Server Node*. Penggambaran sistem secara umum dapat dilihat pada gambar berikut:




Gambar 8 Topologi sistem

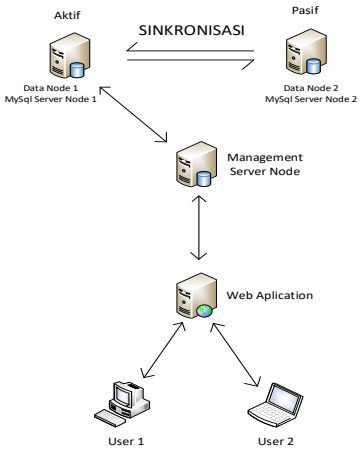
Gambar diatas *Data Node* dan *MySql Node* digabungkan menjadi satu *Node*. Kemudian untuk menghubungkan *Node* A dan *Node* B dibutuhkan *Management Server Node*. Jadi fungsi utama *Management Server Node* adalah menghubungkan kedua *node* tersebut sehingga fungsi dari *MySQL*

*P-ISSN:0216-9436*
*E-ISSN:2622-6782*

*cluster* itu sendiri dapat berjalan dengan fungsi sebagai replikasi data dan *high availability*.

**2. Tahap Simulasi**

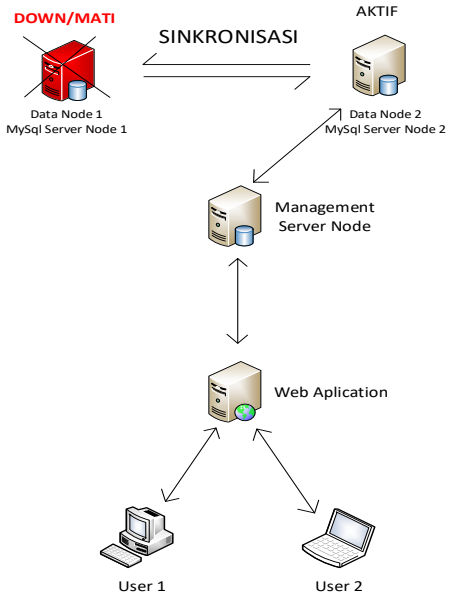
**a. *Prototype*** Sinkronisasi Data *Real Time* Pada *MySql Cluster*




Gambar 9 Simulasi arsitektur *MySql Cluster*

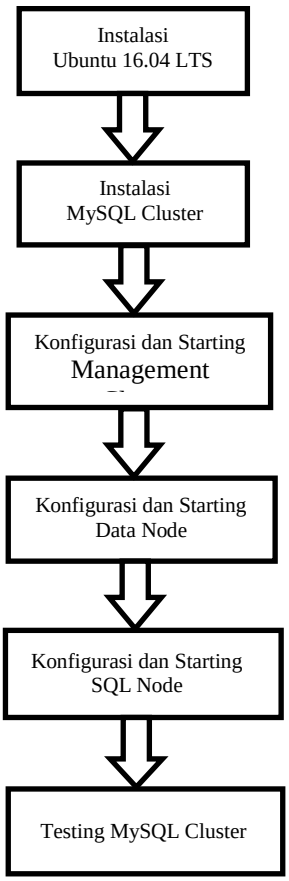
Gambar diatas menunjukan proses kerja *MySql Cluster* pada posisi *default.* User akan mengirim request kepada web aplication dan kemudian web aplication mengirimkannya ke *Management Node Server*. *Management Node Server* sendiri memiliki fungsi sebagai pengatur dan penghubung antar kedua node lainnya. Pada posisi *default, server* yang aktif memberikan *service* adalah *Node 1*. Sedangkan *Node 2* bersifat pasif. Meskipun tidak memberikan *service* kepada *user/client, Node 2* tetap melakukan sinkronisasi secara *real time* dengan *Node 1* sehingga data pada *Node 1* akan tetap sama dengan data yang ada pada *Node 2*.

b. Simulasi *high – Availability* saat terjadi kegagalan (*failure*) pada salah satu server.




Gambar 10 Simulasi *high Availability MySql Cluster*

Gambar diatas menunjukan bagaimana kondisi saat terjadi kegagalan (*failure*) pada *MySql* dan *Data Node 1*. Ketika *server* sedang memberikan *service* kepada client akan tetapi jika tiba – tiba terjadi kegagalan (*failure*) maka secara otomatis *Management Node Server* akan mengalihkan *service* kepada *MySql* dan *Data Node 2*. Sehingga *client* tidak akan menyadari dan terganggu karena adanya masalah atau kegagalan pada servernya




Konfigurasi *Mysql cluster* adalah menginstal *mysql cluster* secara manual, bukan langsung menginstal dari paket yang tersedia di *repository* yang sudah disediakan oleh linux. Tetapi berbeda halnya ketika kita memakai windows, cara menginstal bisa secara manual maupun secara langsung dengan menggunakan .exe nya. Konfigurasi ini sendiri terbagi menjadi tiga, yaitu pengaturan konfigurasi di sisi *Management Server Node*, pengaturan konfigurasi di sisi Node A dan pengaturan di sisi Node B

Gambar 11 Alur simulasi pengujian sistem

## 3. Tahap Implementasi

Untuk membangun *server* ini penulis menggunakan sistem operasi berbasis *opensource* yaitu Proxmox VE sebagai *server* virtualisasi dan linux Ubuntu 16.04 LTS 64 bit sebagai *server* fisik, sedangkan pada sisi *database* penulis menggunakan *MySql server* sebagai infrastruktur *database server*

*Requirements* atau spesifikasi sistem minimum yang direkomendasikan adalah sebagai berikut.

Komputer Server

- *Processor Intel Pentium* 4 – 2,6 GHz
- *Hard Disk* 80 GB 7200 RPM
- *Memory* ( RAM ) 1 GB
- *Monitor*
- *Keyboard* dan *Mouse*

Perangkat keras dan perangkat lunak yang digunakan penulis untuk implementasi dan pengujian *clustering database server* sebagai berikut:

Perangkat Keras

1. Komputer *database server*
   - *Inter Core* i5 – 2450M
   - *Processor* 2.50 GHz
   - *Hard Disk* 500 GB
   - *Memory* (RAM) 4 GB

Perangkat Lunak

- Sistem Operasi Ubuntu Desktop 16.04 LTS
- *MySql Cluster*
- *VirtualBox*
- *Sysbench*

***a.*** **Konfigurasi *Management Node***

*Management Node* adalah komponen pertama yang harus dimulai dalam *cluster* apapun. Diperlukan sebuah *file* konfigurasi yang berfungsi untuk mendeklarasikan arah jalur *node – node* yang dibuat. *File* tersebut sangan penting dalam sistem cluster karena berfungsi mengatur secara keseluruhan MySQL Cluster itu sendiri. *File* tersebut akan diberi nama *file* config.ini yang disimpan pada direktori /var/lib/mysql-cluster/config.ini.

Berikut adalah tahapan konfigurasi *management node*::

1. Membuat direktori mysql-cluster
   # mkdir/var/lib/mysql-cluster
2. Membuat file dan edit config.ini
   # nano /var/lib/mysql-cluster/config.ini
3. Membuat *symbolic link* atau *symlink* folder mysql
   # ln -s /opt/mysql/server-5.6/ /usr/local/mysql
4. Menambahkan PATH baru untuk variabel PATH
   #exportPATH=$PATH:/usr/local/mysql/bin
   #echo“exportPATH=\$PATH:/usr/local/mysql/bin” >> /etc/bash.bashrc
5. Memindahkan file ndb_mgm dan ndb_mgmd ke dalam direktori /usr/local/bin.
   # cp /usr/local/mysql/bin/ndb_mgm* /usr/local/bin/
   # ls /usr/local/bin/ndb_mgm*

***b.*** **Konfigurasi *Data Node* dan *SQL Node***

*MySQL data node* terdiri dari dua komponen (*node*), yaitu *SQL node* (*mysqld process*) dan *data node* (*ndbd process*). *SQL node* (*mysqld process*) berfungsi sebagai pintu akses untuk masuk ke dalam node-node data yang ter-cluster. Sedangkan *data node (ndbd process)* digunakan untuk menyimpan semua data yang menjadi milik *MySQL Cluster*. Semua data direplikasi di node-node ini

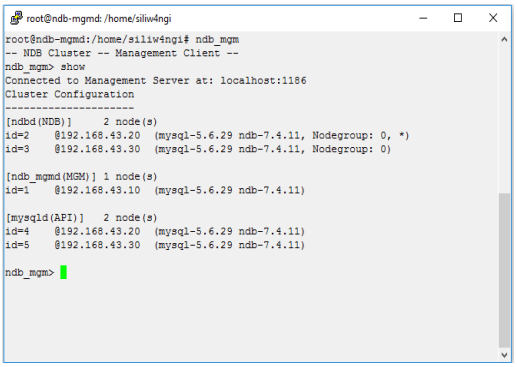
1. Membuat *symbolic link* atau *symlink* folder mysql# ln -s /opt/mysql/server-5.6/ /usr/local/mysql
2. Menambahkan PATH baru untuk variabel PATH
   #export PATH=$PATH:/usr/local/mysql/bin
   # echo “export PATH=\$PATH:/usr/local/mysql/bin” >> /etc/bash.bashrc
3. Konfigurasi *Data Node* dan *SQL Node* Membuat *group* dan *user*
   # groupadd mysql
   # useradd –g mysql mysql
4. Membuat *database default*
   # cd /usr/local/mysql
   # ./scripts/mysql_install_db --user=mysql
5. Mengatur ijin yang diperlukan untuk *MySQL server* dan data direktori
   # chown -R root .
   # chown -R mysql data
   # chgrp -R mysql
6. Konfigurasi *data* dan *SQL node*
   # nano /etc/my.cnf
7. Menjalankan service ndbd pada start-up configuration
   # echo "/usr/local/mysql/bin/ndbd" >> /etc/rc.local
8. Memindahkan file ndb_mgm dan ndb_mgmd ke direktori /usr/local/bin
   # cp /usr/local/mysql/bin/ndb_mgm* /usr/local/bin/
9. Menyalin *script start-up MySQL server* ke direktori baru
   # cp support-files/mysql.server /etc/init.d/mysql
10. Memberi hak akses eksekusi pada *file mysql*
    # chmod +x /etc/init.d/mysql

**c. Pengujian konektivitas**

Pengujian pertama yaitu melakukan starting pada setiap server dan melakukan pengecekan apakah server tersebut sudah terkoneksi satu sama lain atau belum

```
# nano /var/lib/mysql-cluster/config.ini
# ndb_mgmd --initial -f /var/lib/mysql-
    cluster/config.ini --
    configdir=/var/lib/mysql-cluster/
```

- Cek port 1186 (*port default management node*)
  ```
  # netstat –plntu
  ```
- Cek management node telah berjalan
  ```
  # ndb_mgm
  # show
  ```
- *Starting MySQL* dan *Data Node*
  ```
  # nano /etc/my.cnf
  ```
- Starting data node (ndbd)
  ```
  # ndbd –initial
  ```
- *Starting SQL node* (*mysqld*)
  ```
  # systemctl enable mysql
  # systemctl start mysql
  ```
- Cek koneksi pada management node
  ```
  # ndb_mgm
  # show
  ```




Gambar 12 Tampilan informasi koneksi pada *management node*

**d. Pengujian Replikasi Data**

Setelah memastikan bahwa sistem MySQL cluster sudah berjalan dengan pengujian konektivitas yang telah dilakukan sebelumnya, maka selanjutnya adalah tahap pengijian replikasi data. Simulasi pengujian ini yaitu dengan membuat database, tabel dan input data pada node A, dan kemudian akan dilihat apakah data yang dimasukan pada node A secara otomatis akan masuk ke dalam node B juga atau tidak.

- Tahap masuk ke dalam MySQL

```
# mysql –u root –p
mysql> show databases;
mysql> create database test;
mysql> show databases;
mysql> use banisaleh;
mysql>  create table  data_mhs (
npm int(12) not null,
nama varchar(50),
alamat varchar(50) )
engine=ndbcluster;
mysql> show tables;
mysql> insert into data_mhs values
(1001,'deni','bekasi');
mysql> select * from data_mhs;
```

**e. Pengujian *Response Time* dan *Throughput***

Pada proses pengujian ini penulis menggunakan *tools* aplikasi *sysbens* yang berjalan pada mode terminal ubuntu. Parameter yang diukur dalam pengujian ini adalah *throughput* dan *response time* yang diubah jumlah threadsnya Pengukuran tersebut dilakukan berdasarkan mode pengujian berikut:

- *Simple-mode* merupakan pengujian yang dilakukan hanya pada proses *read-only* data atau hanya pembacaan data saja.
- *Complex-mode* merupakan pengujian yang dilakukan dengan proses yang kompleks yang meliputi READ, INSERT, DELETE, dan UPDATE

  Parameter dalam menjalankan pengujian *response time* dan *throughput* yang menjadi batasan dalam pengujian adalah sebagai berikut:

  1. Jumalah record = 100.000
  2. Max. Time = 60 sec.

Pada pengujian ini juga penulis akan membandingkan hasil dari pengujian *response time* dan *throughput* MySQL cluster dengan hasil pengujian *response time* dan *throughput* pada MySQL server default sebagai pembanding dalam uji performa ini. Sebelum melakukan pengujian terlebih dahulu kita siapkan database baru untuk dipakai dalam uji coba ini. Database tersebut diberi nama 'sbtest'. Berikut adalah perintahnya.

```
# mysql –u root –p
mysql> create database sbtest;
```

Berikut adalah data hasil dari pengujian berdasarkan nilai threads yang telah ditentukan:

**A. Skenario *Simple-mode***

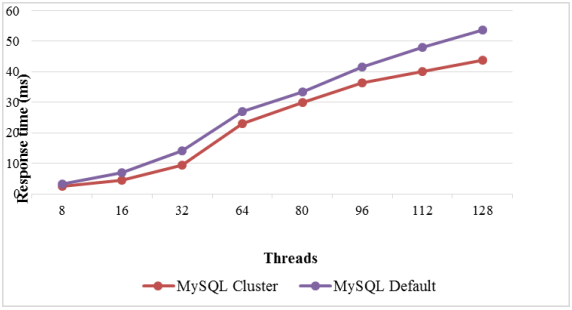
Pada mode ini akan ditampilkan data *throughput* dan *response time* dari hasil transaksi *read-only* yang selengkapnya dapat dilihat pada Tabel 1 merupakan hasil dari pengujian *response time* dan *throughput* pada MySQL cluster dan MySQL server dafault. Dapat dilihat bahwa performa dari MySQL cluster lebih baik dari pada MySQL server default. Dari tabel tersebut diperoleh penurunan jumlah *throughput* mulai dari 8

*P-ISSN:0216-9436*
*E-ISSN:2622-6782*

hingga 128 *threads*. Kondisi ini berpengaruh terhadap parameter *response times* yang semakin lambat akibat penambahan *threads* tersebut.

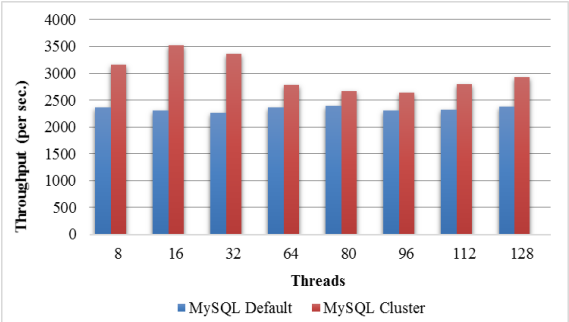
Tabel 1 Rata-rata pengujian MySQL cluster *simple-mode*

| Threads | MySQL Cluster | | MySQL Server (Default) | |
|---|---|---|---|---|
| | Response time (ms) | Throughput (per sec.) | Response time (ms) | Throughput (per sec.) |
| 8 | 2,5 | 3164,4 | 3.33 | 2361.92 |
| 16 | 4,54 | 3518,02 | 6.84 | 2309.35 |
| 32 | 9,5 | 3365,32 | 14.04 | 2269.4 |
| 64 | 22,97 | 2785,01 | 26.99 | 2364.92 |
| 80 | 29,95 | 2669,75 | 33.42 | 2387.11 |
| 96 | 36,34 | 2640,65 | 41.56 | 2304.73 |
| 112 | 39,96 | 2801,86 | 48 | 2328.25 |
| 128 | 43,78 | 2922,67 | 53.59 | 2383.29 |




Gambar 13 Grafik *Response time simple-mode*

*Response time* merupakan rentang waktu dari saat user mengirimkan request ke server hingga server selesai merespon request tersebut. Gambar 4.47 memperlihatkan pengaruh penambahan *threads* terhadap *response time*. Parameter ini akan berdampak pada kecepatan akses yang dibutuhkan suatu *node cluster*. Semakin kecil nilai yang diperoleh menyatakan semakin baik pula kondisi *cluster* tersebut.




Gambar 14 Grafik *Throughput simple-mode*

*Throughput* adalah jumlah *request* yang ditangani dalam satu detik. Gambar 4.49 memperlihatkan pengaruh penambahan *threads* terhadap *throughput*. Parameter ini akan berdampak pada jumlah *request* yang ditangani suatu *node cluster* dalam satu detik. Semakin besar nilai yang diperoleh menyatakan semakin baik pula kondisi *cluster* tersebut.

B. Skenario complex mode

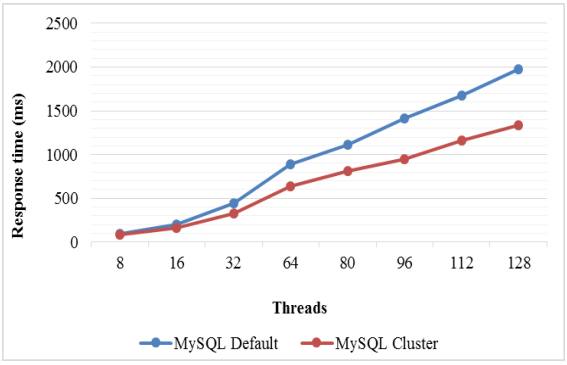
Pada mode ini data yang ditampilkan adalah *throughput* dan *response time* dengan mode *complex-mode*. Terdapat beberapa perbedaan dibandingkan *simple-mode*. *Complex-mode* merupakan pengujian yang dilakukan dengan proses yang kompleks yang meliputi READ, INSERT, DELETE, dan UPDATE. Untuk hasil selengkapnya dapat dilihat pada tabel hasil benchmarking dibawah ini.

Tabel 2 Rata-rata pengujian MySQL *cluster complex-mode*

| Threads | MySQL Cluster | | MySQL Server (Default) | |
|---|---|---|---|---|
| | Response time (ms) | Throughput (per sec.) | Response time (ms) | Throughput (per sec.) |
| 8 | 85.71 | 1672.72 | 95,49 | 1591.01 |
| 16 | 166.06 | 1633.16 | 199,62 | 1521.83 |
| 32 | 325.9 | 1668.63 | 448,6 | 1353.33 |
| 64 | 636.67 | 1626.26 | 894,55 | 1356.11 |
| 80 | 815.97 | 1580.81 | 1110,6 | 1363.55 |
| 96 | 950.34 | 1450.25 | 1417,12 | 1281.18 |
| 112 | 1159.44 | 1415.36 | 1678,99 | 1260.92 |
| 128 | 1335.65 | 1406.27 | 1977,56 | 1222.84 |

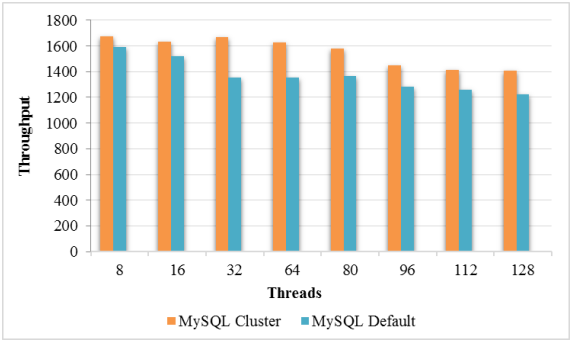
*P-ISSN:0216-9436*
*E-ISSN:2622-6782*

Nilai yang diperoleh pada tabel diatas menunjukan perbedaan dengan percobaan sebelumnya yakni memakai *simple-mode*. Pada pengujian *complex-mode* nilai *response time* lebih besar, begitu juga dengan nilai pada *throughput* lebih kecil. Namun dapat dilihat bahwa performa MySQL cluster masih lebih baik dari pada MySQL server default.




Gambar 15 Grafik *Response time complex-mode*

Perubahan yang terjadi pada gambar 15 menunjukkan nilai yang terbaik pada MySQL cluster adalah 85,71 ms pada saat kondisi *default* pada beban 8 *threads*. Sedangkan nilai terburuknya adalah 1335.65 ms pada saat kondisi *default* diberikan beban 128 *threads*. Nilai tersebut masih berada dibawah nilai MySQL server dafault. Nilai tersebut terlihat sangat signifikan disebabkan pengujian pada *complex-mode* terjadi adanya variasi transaksi yang membutuhkan waktu yang lama untuk menyelesaikannya. Proses tersebut juga dipengaruhi oleh pembagian beban trafik pada sistem *cluster* keduanya.




Gambar 16 Grafik *Throughput complex-mode*

Pada Gambar 16 dapat dilihat nilai *throughput* MySQL cluster yang tertinggi adalah 1672.72 pada beban 8 *threads*. Sedangkan nilai *throughput* terendah adalah 1406.27 pada beban 128 *threads*. Pada MySQL server default nilai *throughput* yang tertinggi adalah 1591.01. Sedangkan nilai *throughput* terendah adalah 1222.84 pada beban 128 *threads*. Artinya performa MySQL cluster lebih baik dibandinkan dengan performa MySQL server default dari sisi pengujian *throughput*.

Nilai *throughput* yang diperoleh pada mode ini lebih kecil dibandingkan pada *simple-mode*. Hal ini dikarenakan pada mode ini terjadi variasi proses transaksi SQL yang meliputi; READ, WRITE, INSERT dan UPDATE. Apabila salah satu dari transaksi tersebut tidak dipenuhi, maka transaksi yang terjadi tidak terhitung. Berbeda dengan *simple-mode*, transaksi yang terjadi hanya READ saja. Jadi kemungkinan transaksi yang terjadi semakin banyak.

## 3. KESIMPULAN

Dari hasil analisa dan pembahasan maka dapat ditarik kesimpulan sebagai berikut:

1. Sistem *clustering database server* menggunakan MySQL *cluster* berhasil dibangun diatas mesin virtual berbasis *open source* yang didalamnya terpasang sistem operasi Ubuntu 16.04 LTE (*Xenial Xerus*).
2. Performa sebuah *database* meningkat jika dibandingkan dengan *database* MySQL yang tidak menggunakan sistem *clustering*.
3. Hasil dari pengukuran *response time* dan *throughput* menunjukan performa MySQL *cluster* cukup baik dibandingkan dengan MySQL server (*default*).

## PUSTAKA

Arbie. (2003). *Manajemen Database dengan MySQL*. Yogyakarta: Andi Offset.

Ariyus, D. (2008). *Pengantar Ilmu Kriptografi*. Yogyakarta: Andi offset.

Atzeni, P. e. (2003). *Database System: Concepts, Languages & Architecture*. Australia: McGraw-Hill.

Connolly, T., & Begg, C. (2010). *Database system: a practical approach to design, implementation and management 5th Edition*. America: Person Education.

Hodges, R. (2007). *Database High Avaibility and Scalability*. America: CTO Continuent Inc.

Munir, R. (2006). *Kriptografi*. Bandung: ITB.

O'Brien, & James, A. (2005). *Pengantar Informatif Perseptif Bisnis dan Manajerial*. Jakarta: Salemba.

Raharjo, & Budi. (2011). *Membuat Database Menggunakan MySQL*. Bandung: Informatika.

Triwahyuni, A. K. (2003). *Pengenalan Teknologi Informasi*. Yogyakarta: Andi Yogyakarta.