IMPLEMENTASI DAN ANALISIS KINERJA MySQL CLUSTER MENGGUANAKAN METODE LOAD BALANCING

Intan Putri Andhikha
D3 Manajemen Informatika, Fakultas Teknik, Universitas Negeri Surabaya, iintanok@gmail.com

Asmunin
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Negeri Surabaya, asmunin@unesa.ac.id

Abstrak

Di era yang sudah berkembang seperti halnya dunia perindustrian pasti membutuhkan sistem kinerja pada sistem informasi dan data yang memiliki ketersediaan data yang tinggi (*high availbility*) dan bekerja dengan efisien contohnya seperti menanggulangi kegagalan dalam transaksi data, sudah menjadi kebutuhan yang sangat penting juga di dunia modern seperti sekarang.

Salah satu solusi untuk mengatasi masalah ini adalah dengan menggunakan teknologi *MySQL Cluster*. Namun dengan adanya *request* data yang terlalu banyak, dan tidak mampu melayaninya, sistem dapat mengalami penurunan performansi. Maka dari itu untuk mengatasi hal tersebut dibutuhkan suatu penelitian untuk membagi beban disetiap *server* yaitu dengan menggunakan sistem *loadbalancing*. Tujuannya adalah menganalisa kinerja dari *Mysql Cluster* dengan melakukan transaksi *database insert, delete, update* dan *select* yang kemudian hasil durasi waktu yang diperoleh akan dianalisis serta menggunakan aplikasi *benchmark sysbench* cara kerjanya dengan memberikan beban *thread* 2-128 dengan menganalisa jumlah *Transaction per Second* dari masing masing kinerja dengan penambahan *loadbalancing* dan *non loadbalancing*. Metode yang digunakan dalam mengerjakan tugas akhir ini yaitu menggunakan teknologi *backup* data sinkronisasi dengan *engine ndb clustering* dalam melakukan sinkronisasi pada *mysql cluster* dan menggunakan teknologi algoritma *round robin* pada teknik pembagian beban sistem *loadbalancing*.

Tujuan dari penelitian ini adalah untuk menganalisa kinerja *Mysql cluster* dalam mempermudah kinerja *database* yang terlalu banyak dalam perbandingan kondisi *loadbalancing* yaitu dengan teknologi haproxy yang dihidupkan, sedangkan pada saat kinerja *non loadbalancing* yaitu dengan me-non aktifkan teknologi haproxy.
Kata Kunci: *Mysql Cluster, Load balancing, failure, high availbility, backup data.*

Abstract

In the developing world as well as the industrial world inevitably requires a system of performance in information systems and data that have high availbility and work efficiently such as tackling failure in data transaction, has become a very important need also in the modern world as it as now.

One solution to solve this problem is to use mysql cluster technology. But with the request too much data, and not able to serve it, the system can experience a decrease in performance. Therefore to overcome this required a study to divide the load on each server that is by using loadbalancing system. The goal is to analyze the performance of the mysql cluster by performing database transactions insert, delete, update and select which then the results of the duration obtained will be analyzed and use benchmark sysbench application, how it works by loading thread 2-128 by analyze transacton per second of each performance with the addition of loadbalancing and non loadbalancing. The method used in this final task is to use technology data backup synchronization with ndb clustering engine in synchronize on mysql cluster and using round robin algorithm technology on loadbalancing system load sharing technique.

The purpose of this study is to analyze the performance of mysql cluster in facilitating the performance of the database is too much in the comparison of loadbalancing conditions that is haproxy technology is turned on, while at the time of non loadbalancing performance that is disable technology haproxy.
Keywords: *Mysql Cluster, Load balancing, failure, high availbility, data backup*

PENDAHULUAN

Di era yang sudah berkembang seperti halnya dunia perindustrian, pendidikan, maupun pemerintahan pasti membutuhkan sistem kinerja pada sistem informasi dan data yang memiliki ketersediaan data yang tinggi dan bekerja dengan efisien contohnya seperti menanggulangi kegagalan dalam transaksi data dan *backup* data yang sudah menjadi kebutuhan sangat penting di dunia *modern* seperti sekarang. Kinerja sistem informasi dan data salah satunya adalah sistem informasi *database* yang harus selalu menyediakan data dengan ketersediaan yang tinggi dan meminimalkan terjadi kegagalan karena salah satu server mati dan tidak ada *backup* data dari server lain. Salah satu solusi untuk mengatasi masalah ini adalah dengan menggunakan

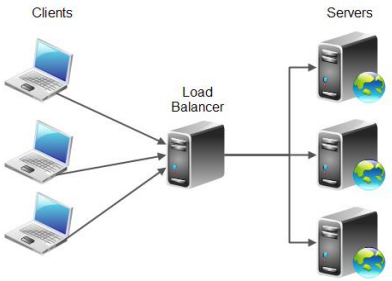
teknologi *mysql cluster* dikarenakan didalam teknologi *mysql cluster* terdapat *backup database* dan juga terdapat sistem yang mampu mengatasi kegagalaan sistem *database*.

Namun dengan adanya *request* data yang terlalu banyak, dan tidak mampu melayaninya, sistem dapat mengalami penurunan kinerja, di sisi lain *database* harus selalu menyediakan data dengan ketersediaan yang tinggi dan secara terus menerus sehingga setiap pengguna dapat memperoleh data tanpa mengalami gangguan atau kegagalan seperti beberapa saat tidak bisa mengakses sistem tersebut atau disebabkan karena server mati dan tidak ada *backup* dari server lain yang langsung menggantikan ketika salah satu server mati. Salah satu solusi untuk mengatasi masalah ini supaya server bekerja secara lebih stabil dan seimbang dengan beban kerja pada beberapa server untuk itu diperlukan penambahan sistem *loadbalancing*. Penelitian ini bertujuan untuk menganalisis kinerja *mysql cluster* dalam kondisi *load balancing* dan *non load balancing (default).*

KAJIAN PUSTAKA

Load Balancing

Menurut Rendra Towidjojo (2013) *load balancing* adalah teknik jaringan komputer yang menggunakan metode pendistribusian terhadap beban trafik untuk membagi beban dalam beberapa jalur koneksi atau *link* secara seimbang. Tujuannya agar tidak ada *link* atau koneksi yang mendapatkan beban yang lebih besar dari *link* atau koneksi lainnya. Dengan membagi beban atau *load* kedalam beberapa *link* tersebut diharapkan dapat mengelola keseimbangan atau *balancing* untuk pengguna *link* tersebut dan menghindari *overload* pada salah satu jalur koneksi jaringan.



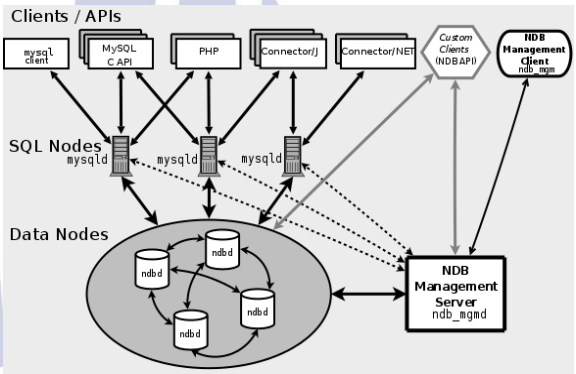
(Sumber: Rendra Towidjojo, 2013)

Gambar 1. Gambaran load balancing

MySQL Cluster

Menurut situs website www.dev.mysql.com. Pada *Mysql Cluster* terbagi atas beberapa teknologi yang digunakan, salah satunya adalah *Cluster NDB (Network Database)* yaitu teknologi yang memungkinkan pengelompokkan *database* dalam memori dan sistem dan bekerja dengan arsitektur *shared-nothing*. *Cluster NDB* dirancang untuk tidak memiliki satu titik pun kegagalan. Pada setiap komponen memiliki memori. *Ndb cluster* sendiri adalah satu kesatuan dari *mysql server* dengan mesin penyimpanan yang tersimpan dalam memori yang disebut *NDB* (yang merupakan singkatan dari "Network DataBase"). Istilah *NDB* mengacu pada bagian dari setup yang spesifik untuk mesin penyimpanan, sedangkan *"MySQL NDB Cluster"* mengacu pada kombinasi satu atau lebih *server MySQL* dengan mesin penyimpanan *NDB.*

Semua program ini bekerja sama untuk membentuk *Cluster NDB*. Bila data disimpan oleh mesin penyimpan *NDB,* tabel disimpan di *node* data. Tabel tersebut dapat diakses secara langsung. Dari semua *server MySQL* lainnya. di *cluster*, sehingga dalam aplikasi penggajian menyimpan data dalam sebuah *cluster*, jika satu aplikasi meng*update* gaji seorang karyawan, semua *server MySQL* lainnya yang menanyakan data ini dapat segera melihat perubahan ini. Data yang tersimpan dalam data *node* untuk *NDB Cluster* dapat dicerminkan, yang artinya cluster dapat menangani kegagalan atau *failure* pada data *node.*



(Sumber: www.dev.mysql.com.)

Gambar 2. Gambaran kinerja mysql cluster

Ndb Cluster adalah mesin penyimpanan dalam memori yang menawarkan ketersediaan tinggi dan ketekunan data. Mesin penyimpanan *ndb cluster* dapat dikonfigurasi dengan berbagai pilihan failover dan *load-balancing*, namun paling mudah untuk memulai dengan mesin penyimpanan di tingkat *cluster*. Mesin penyimpanan *NDB Cluster NDB* berisi satu set data lengkap.

Terdapat tiga tipe cluster node dan dalam konfigurasi *Cluster NDB* minimal terdapat tiga *node* yaitu:

1. *Management node*, Peran tipe *node* ini adalah mengelola *node* lain dalam *Cluster NDB*, melakukan fungsi seperti menyediakan data konfigurasi, mulai *(start)* dan menghentikan *node,* dan menjalankan *backup.* Karena tipe node ini mengelola konfigurasi *node* lain, sebuah node dari tipe ini harus dimulai terlebih dahulu sebelum node lainnya. Sebuah *node MGM* diawali dengan perintah *ndb_mgmd*.
2. *Data node*, jenis *node* ini menyimpan *data cluster* digunakan untuk menyimpan semua data transaksi

pada *MySQL cluster* dan data tersebut direplikasi pada *node* ini.

3. *SQL Node* merupakan interface yang digunakan oleh *clinet* untuk mengakses data atau supaya terkoneksi kedalam database yang terdapat pada *node* sistem *cluster* fungsi lainnya adalah mengembalikan cadangan *cluster*

Database

Dalam buku Sistem Manajemen Basis Data karya Haryanto (2004) *database* atau basis data merupakan sebuah kumpulan berupa data logik yang saling berkaitan atau berhubungan untuk mempresentasikan fakta secara terstruktur dalam *domain* tentunya untuk mendukung aplikasi pada sistem tertentu. *Database* juga mendiskripsikan suatu sistem dan merupakan suatu komponen utama sistem informasi karena semua informasi pengambilan keputusan berasal dari *database* atau basis data.

METODE

Analis Sistem

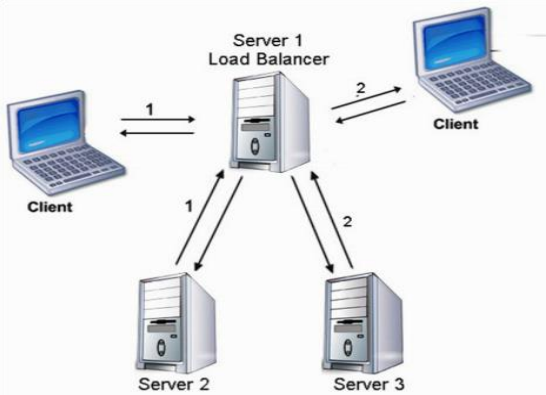
Pada tahap ini dilakukan analisa kebutuhan sistem untuk menjadi dasar menentukan mekanisme sistem yang akan diterapkan pada *mysql cluster* dengan metode *load balancing. Load balancing* adalah teknik pembagian trafik beban supaya menjadi seimbang dan merata.

Salah satu algoritma yang digunakan pada metode *load balancing* untuk membagi beban kerja yaitu salah satunya menggunakan *algoritma round robin*. Pada gambar 3 terdapat gambar mekanisme kerja *load balancing clustering*.

Dari gambar 3 *load balancing* akan menerima sebuah trafik dari *client* selanjutnya trafik tersebut akan dilanjutkan ke beberapa *server* lainnya. Trafik akan diurutkan berdasarkan permintaan *client* dan kondisi *server*. Ketika hanya satu atau salah satu *client* saja yang melakukan *request*, maka proses akan terjadi pada salah satu *server* saja, apabila *request* melebihi, *maka load balancer* akan bekerja yakni membagi proses tersebut secara seimbang untuk masing-masing *server*.

Dengan menggunakan algoritma ini beban akan terbagi secara merata pada semua server yang saling berhubungan. Setiap proses baru yang ditugaskan pada *server* akan masuk antrian dan urutannya disusun berdasarkan proses yang sedang berlangsung maka proses yang berlangsung akan seimbang.

Untuk itu teknik *load balancing* ini akan diterapkan atau diimplementasikan pada *MySQL Cluster*. Teknologi *software* yang menyediakan ketersediaan tinggi. *MySQL Cluster* memiliki 3 komponen yaitu *Management Server Node, SQL Node*, yang langsung terhubung dengan *Data Node*.



Gambar 3. Gambaran load balancing

Pada analisis sistem teknologi *Mysql Cluster* ini, menggunakan *NDB Cluster*. Sistem kinerja dari *NDB Mysql* ini nantinya data yang tersimpan dalam data *node* untuk *NDB Cluster* dapat dicerminkan, dari pencerminan ini *ndb cluster* bersifat *replikasi Multi-Master* yaitu setiap *node* data dapat menerima operasi *write*. Ditambah dengan *auto-sharding*, ini memberikan *skalabilitas write* yang sangat tinggi, yang artinya *cluster* dapat menangani kegagalan atau *failure* pada data *node.*

Ndb Cluster adalah mesin penyimpanan dalam memori yang menawarkan ketersediaan tinggi dan ketekunan data. Mesin penyimpanan ndb cluster dapat dikonfigurasi dengan berbagai pilihan *failover* dan *load-balancing*.

Dan nantinya untuk pengukuran analisa kinerja performansi antara uji coba *MySQL Cluster load balancing* dan *MySQL Cluster* secara *non load balancing* antara lain:

1. Uji *coba high availability* sekaligus uji coba sinkronisasi dari sisi kinerja *mysql cluster* dengan melakukan seperti *insert, delete, update* serta uji coba failure.
2. Uji coba pembagian beban atau sisi *loadbalancing* yaitu pengujian dari sisi jumlah *user* atau *client* yang mengakases nantinya akan dibagi beban pada *server* secara adil dan merata.

Pengujian menggunakan *sample database* . Pengujian kedua sistem *load balancing* dan *non loadbalancing* menggunakan topologi yang sama. Jalannya sistem yaitu ketika beberapa *clinet* atau *user* masuk akan dieksekusi oleh server *loadbalancing* dan dibagi bebannya sama rata untuk tiap tiap server setelah itu dari *loadbalancing client* akan melakukan transaksi pada *database* maka akan diterima oleh server data *node*. Pada transaksi data nantinya *mysql cluster* akan menjalankan fungsinya yaitu sebagai *backup* data. Apabila ada transaksi *database* yang dilakukan di data *node* 1 maka secara otomatis sistem *mysql* bekerja yaitu data transaksi juga akan otomatis masuk di data *node* 2.

Desain Topologi

Pada gambar 4 menunjukkan ilustrasi sederhana mengenai sistem ini. Untuk penjelasannya sebagai berikut:

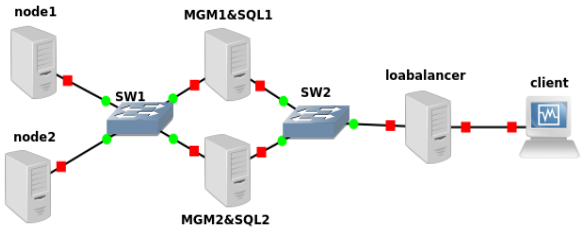
Uji coba akan dilakukan dua kali yaitu:

1. Secara *load balancing*
2. Secara *non load balancing*.

Namun tetap menggunakan satu *topologi* dan dilakukan secara bergantingan dan pada saat uji coba *load balancing*, nantinya sistem *loadbalancing* akan dihidupkan begitu juga saat uji coba *non load balancing* sistem *loadbalancing* akan di matikan.

Pada gambar desain sistem yang digambarkan yaitu perancangan *MySQL Cluster* menggunakan 5 *server* yaitu:

1. *Server* pertama sebagai *data node 1*
2. *Server* kedua sebagai *data node 2*
3. *Server* ketiga dan keempat sebagai *management node dan SQL Node*
4. Server ke 5 sebagai *loadbalancing*



Gambar 4. Gambaran Umum Perancangan Sistem

Dijelaskan pada desain sistem diatas bahwa *node1* dan *node2* adalah *server database* keduanya fungsinya sama yaitu bekerja sebagai server database yang berada di belakang *(back end node*), lalu *server1* dan *server2* adalah *server management node* sekaligus berperan sebagai SQL *node* bekerja sebagai *server* yang berada didepan *(front end)*, Selanjutnya *server loadbalancing* adalah *server* yang bertugas membagi beban antara kedua *server1* dan *server2*.

Perancangan Sistem

Pada gambar 5 terdapat alur perencanaan pengujian *MySQL Cluster* yang digunakan dapat dilihat pada gambar 4

Untuk penjelasan dari gambar 4 sebagai berikut:

a. Analisa sistem merupakan tahap awal dalam pembuatan tugas akhir ini, dengan menganalisis sistem dapat diketahui bagaimana nantinya projek ini akan berjalan.

b. Menyiapkan *OS* dan *tools* yang digunakan meliputi *linux*, *virtualbox* beserta isinya, *GNS3*, *MySQL Cluster* (tutorialnya), *Ha-proxy*, *Mysql Workbench* , *Sysbench.*

c. Membuat *topologi* jaringan di *GNS3* yaitu membuat topologi yang nantinya akan digunakan dalam uji coba.

d. *Install* dan konfigurasi *MySQL Cluster* dan *Ha-prox* yaitu menyiapkan *linux* lalu meng-*install MySQL Cluster* dan *Ha-proxy* beserta konfigurasinya pada *server* yang digunakan.

e. Konfigurasi *node server management* dan *SQL* pada *MySQL Cluster* yang sudah *terinstall*.

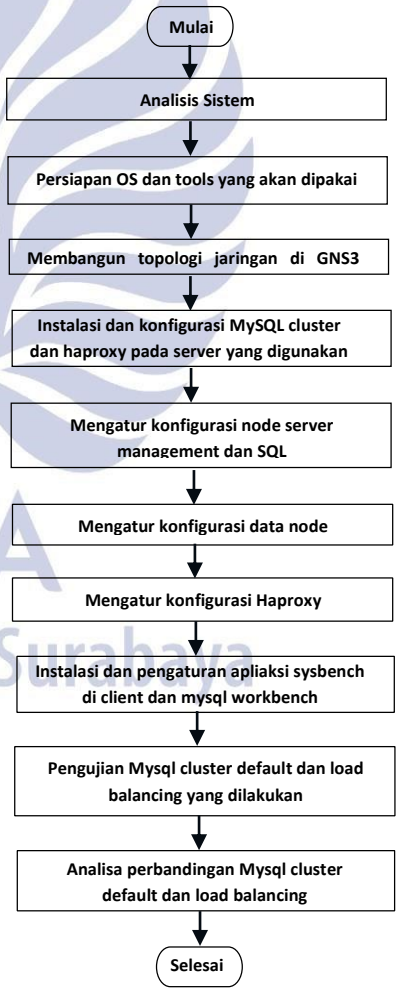
f. Konfigurasi *data node* pada *MySQL Cluster* yang sudah *terinstall*.

g. Konfigurasi *loadbalancing* pada *Ha-proxy* yang sudah *terinstall.*

h. *Install* dan konfigurasi aplikasi *sysbench* di *client* dan install *mysql workbench* yaitu menyiapkan *linux virtual* lalu *meng-install sysbench* dan *mysql workbench* tersebut sebagai alat uji coba tugas akhir ini.

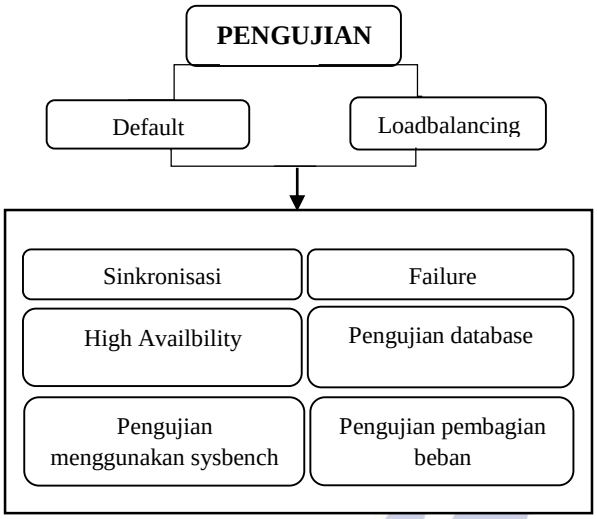
i. Uji coba transaksi *database MySQL Cluster* secara *loadbalancing* dan *MySQL Cluster* menggunakan *non load balancing*.

j. Pengamatan implementasi *MySQL Cluster* secara *loadbalancing* dan *MySQL Cluster* secara *non loadbalancing* yaitu bagaimana perbandingan analisa kinerja berdasarkan uji coba tersebut.



Gambar 5. Alur Perencanaan Pengujian

Skenario Pengujian



Gambar 6. Skenario Pengujian

Skenario pengujian *MySQL Cluster* dengan metode *load balancer* yang akan dilakukan sebagai tahap menganalisa hasil dari uji coba. Tahapan pengujian penelitian sebagai berikut:

1) Melakukan tes koneksi antar *server* yang masing masing terhubung dalam sistem *cluster* dengan melakukan *ping*.
2) Melakukan tes apakah sistem *mysql cluster* dan *haproxy* sudah berjalan.
3) Selanjutnya melakukan uji percobaan *high availbility* dengan *sinkronisasi* yaitu berupa melakukan transaksi database seperti *query insert, delete,* dan *update* beberapa data pada *database,* nantinya akan dilihat apakah sudah sama tersinkronisasi antara kedua *database* maupun jumlah data pada tabel *database* yang kita ubah, kemudian pengujian *failure* yaitu mematikan salah satu node untuk uji coba *server* berhasil digantikan dan tetap berjalan apabila terdapat salah satu *node* yang mati.
4) Melakukan percobaan dari sisi *user* yaitu pengecekan pembagian beban yaitu dengan menambahkan *user* atau *clinet* yang mengakses *server* dan nanti dapat diketahui pembagian beban *user* atau *client* yang sudah masuk mengakses yaitu terbagi antara *server1* dan *server2* dan dapat diketahui beban trafik yang terbagi sama rata.
5) Pengujian melakukan transaksi *CRUD* yaitu *select, insert delete* dan *update* dengan lebih dari satu tabel dan disimpulkan durasi waktu yang didapatkan saat transaksi dengan perbandingan percobaan secara *loadbalancing* dan non *loadbalancing*.
6) Selanjutnya dari pengujian di atas akan diukur *performance-nya* menggunakan aplikasi *sysbench* dan dilakukan perbandingan antara percobaan *MySQL Cluster* secara *loadbalancing* dan *non load balancing* dengan *mode* transaksi *writeonly* dan *readonly* dalam parameter *transaction per second.*

HASIL DAN PEMBAHASAN

Tahap implementasi ini adalah tahap dimana menerapkan hasil dari analisis sistem dan desain penelitian yang telah dibuat sebelumnya. Berbagai macam kebutuhan seperti perangkat lunak dan perangkat keras dibutuhakn pada tahap ini. Berikut daftar semua kebutuhan tersebut.

1. Kebutuhan perangkat keras

   Perangkat keras yang dibutuhkan adalah laptop. Pada pengujian ini laptop yang akan digunakan mempunyai spesifikasi antara lain :

   a. Processor
   Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.60 GHz
   b. RAM
   8,00 GB
   c. Tipe sistem
   Sistem operasi 64-bit

Hasil implementasi dan pembahasan merupakan proses pembangunan beberapa tahap dari awal yang akan dilakukan. Berikut adalah beberapa tahap yang dilakukan untuk implementasi kinerja *mysql cluster* dengan metode *loadbalancing* :

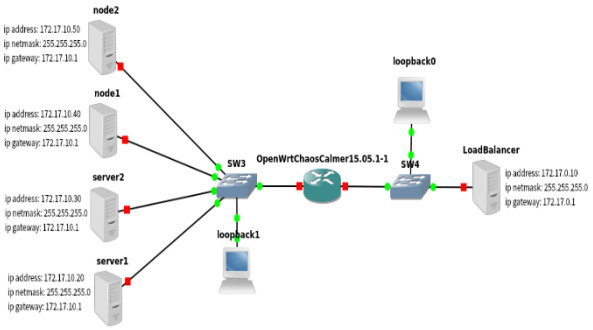
Installasi software

a) *Installasi GNS3* dan *virtualbox* untuk merancang sistem dan sebagai *platform* untuk virtualisasi.
b) *Install OS debian9* di *virtualbox* sebagai *server.*
c) *Install* aplikasi pendukung pengerjaan tugas akhir seperti, *mysql cluster*, *ha-proxy*, *sysbench*, *workbench.*

Dalam bab hasil dan pembahasan ini akan dibutuhkan beberapa proses yang akan menjadi langkah awal dalam melakukan konfigurasi. Berikut adalah beberapa tahapan konfigurasi yang dilakukan dalam implementasi kinerja *mysql cluster* dengan metode *loadbalancing*:

1. Konfigurasi *loopback 0 pada switch 4, loopback 1*
2. Konfigurasi *server 1*
3. Konfigurasi *server 2*
4. Konfigurasi *data node 1*
5. Konfigurasi *data node 2*
6. Konfigurasi *loadbalancing*

7. *Install* dan koneksi pada *workbench*
8. *Install sysbench*
9. Tes koneksi *server 1* dan *server 2* dapat terhubung dengan *data node 1* dan *data node* 2 serta *server loadbalancing(test mysql-cluster* dan *ha-proxy* berjalan*)*
10. Pengujian sinkronisasi sekaligus *high availbility* serta uji coba *failure* antar kedua *data node* yang terhubung pada *server 1* dan *server2.*
11. Pengujian *loadbalancing* pembagian beban antara *server1* dan *server 2*
    a) Uji coba memasukkan beberapa *user* melalui *command prompt*
    b) Melihat pembagian beban yang terjadi pada *url statistik* dari *haproxy* sendiri serta dari beban trafik *CPU* dan *memory* yang ditampilkan pada *htop*
    c) Melihat pembagian beban yang terjadi pada *command prompt*
12. Pengujian transaksi database *select*, *insert*, *delete*, *update* saat *default* saat *loadbalancing* dan *default* dan menganalisis waktu yang dilakukan saat transaksi.
13. Pengujian kinerja *Mysql cluster* secara *default* dan kinerja *mysql cluster* dengan *loadbalancing* dengan *sysbench.*



Gambar 7. Topologi implementasi mysql cluster dengan metode loadbalancing

Berikut penjelasan gambar 7:

a) pada *load balancer* terdapat *ip address: 172.17.0.10*
b) pada *server 1* sebagai *management node* dan *SQL API* terdapat *ip address: 172.17.10.20*
c) pada *server 2* sebagai *management node* dan *SQL API* terdapat *ip address: 172.17.10.30*
d) pada *nada node 1* sebagai *server database* terdapat *ip address: 172.17.10.40*
e) pada *nada node 2* sebagai *server database* terdapat *ip address: 172.17.10.5*

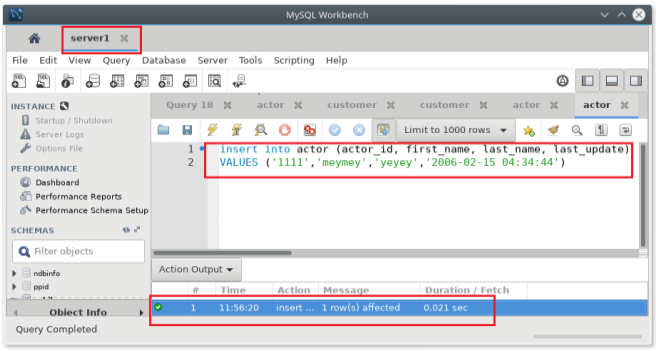
Pengujian dan Pembahasan

1. Pengujian sinkronisasi, high availbility dan failure

Pengujian pembuktian sinkronisasi sekaligus uji *high availbility* dan *failure* pada *mysql cluster* yang terjadi pada *server1* dan *server2*, aplikasi *database* yang digunakan adalah persewaan *vcd sakila.*

Pertama yaitu akan melakukan perubahan di *server1* yaitu melakukan perubahan salah satu *table* dari *database sakila*. Yaitu kita akan mencoba menambah data di *table actor* dengan *actor_id*= 1111, *first_name*= meymey, *last_name*= yeyey, dan *last_update*= 2006—02-15 04:34:44.

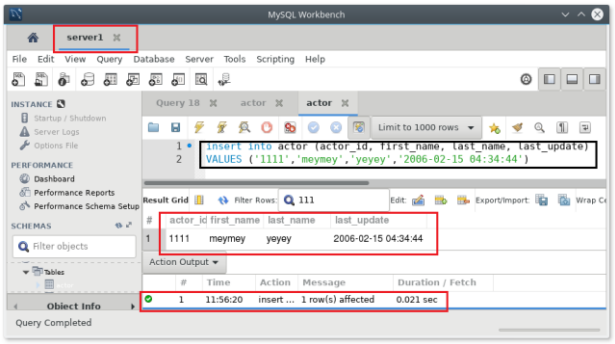
Gambar 10 adalah saat melakukan penambahan data pada *server1* dan berhasil.



Gambar 8. Tampilan insert data di server1

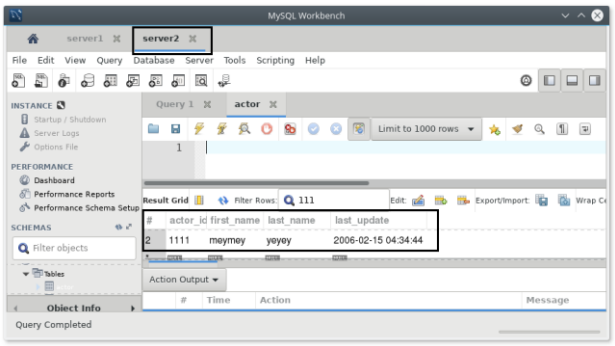
Setelah melakukan penambahan data pada *server1*, maka kita akan *menge-check* data yang kita masukkan sudah masuk atau belum.

Gambar 9 adalah bukti bahwa data sudah masuk, yaitu pada tanda kotak merah.



Gambar 9. Tampilan bukti data sudah masuk

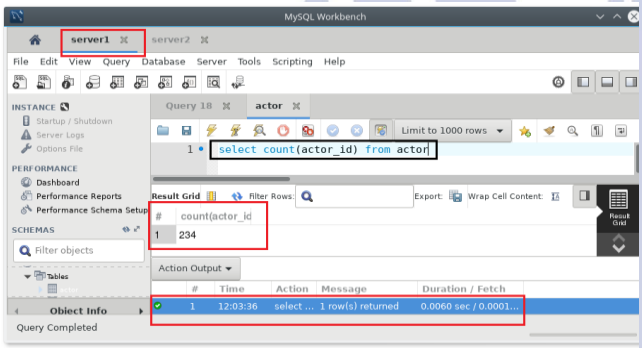
Selanjutnya setelah penambahan data pada *server1* berhasil sudah masuk, lalu kita akan menge-check apakah sudah masuk pada *server2*.



Gambar 10. Tampilan bukti data tersinkron pada server2

Pada gambar 10 terlihat garis kotak hitam bahwa pada *server2* sudah ada data yang sama seperti pada data di *server1*.

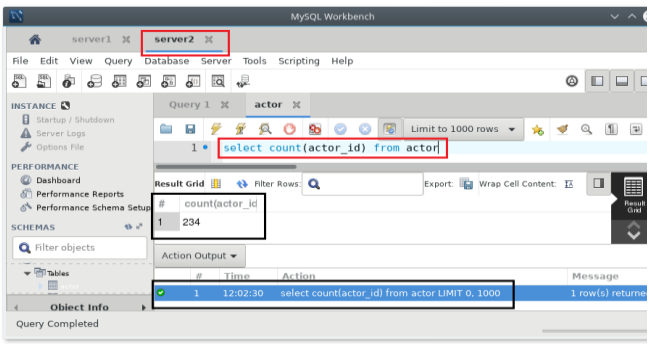
Lalu selanjutnya akan membuktikan lagi yaitu pembuktian *high availbility* apakah data yang terdapat *server1* dan *server2* benar-benar sama atau tidak.



Gambar 11. Tampilan bukti jumlah data pada server1

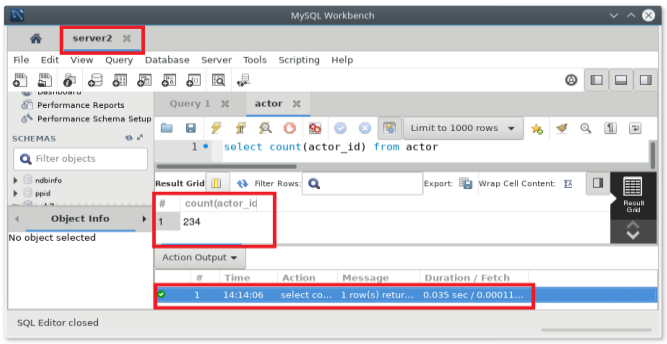
Terlihat pada *server1* bahwa data berjumlah "234" *row*. Kita akan melihat pada *server2* apakah jumlah datanya sama atau tidak.

Dapat dilihat pada gambar 12 bahwa pada *server2* jumlah datanya adalah "234' *row*, itu berarti sama dengan jumlah data yang terdapat pada *server1*. Itu berarti proses pengujian benar-benar bersifat *high avialbility* terbukti dengan adanya sinkronnya jumlah data yang sama dan berhasil antar kedua *server*.



Gambar 12. Tampilan bukti jumlah data pada server2

Kemudian akan mengetest juga apakah *mysql cluster* benar-benar berjalan dan bersifat *failure* atau tidak dengan cara kita matikan salah satu *node*, lalu kita akan menge-check apakah datanya masih utuh dan sama.



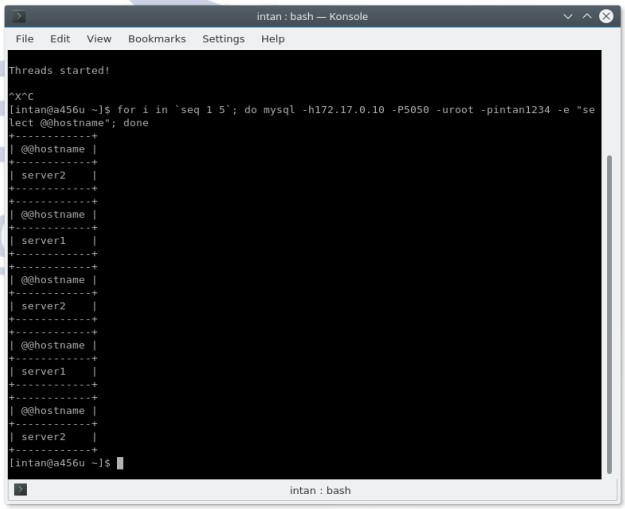
Gambar 13. Tampilan bukti jumlah data pada server2 saat node1 dimatikan

Pada gambar 13 bahwa pada *server2* jumlah data masih tetap sama dan tidak berubah walaupun *node1* telah dimatikan. Ini berarti *mysql cluster* sudah berhasil menangani apabila terdapat salah satu *node* yang mati proses *mysql cluster* masih dapat berjalan.

2. Pengujian beban pada *server1* dan *server2*

a) Memasukkan beberapa user dari command prompt

Dengan cara memasukkan perintah: "`for i in` `` `seq 1 1; do mysql -h172.17.0.10 -P5050 -uroot -pintan1234 -e "select @@hostname"; done` ``



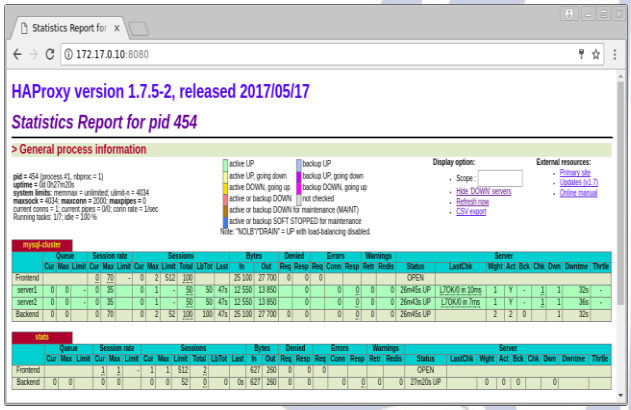
Gambar 14. Test pembagianLoadbalancer memasukkan user

Penjelasan pada gambar 14 adalah, terdapat uji coba pembagian *user* yang masuk ke *server1* dan *server2*, maksud

dari perintah tersebut adalah, *for i* yaitu untuk *iterasi* pada *user* 1 sampai dengan 5 yang akan dilakukan oleh *mysql* pada *hostname loadbalancer* dengan *ip hostname*: 172.17.0.10 yang memiliki *port*:5050 dan *–uroot* untuk masuk ke *root* dengan *password*: intan1234, setelah itu diketahui pembagian beban *user* yang masuk ke *server1* maupun *server2* dengan pembagian yang masuk sama yaitu, *host @@hostname* masuk ke *server2*, selanjutnya masuk ke *server1*, masuk *keserver2*, masuk *keserver1*, dan yang terakhir masuk *server2*. Dengan begitu *user* yang masuk terbagi rata antara yang akan masuk melaui *server1* ataupun melalui *server2*.

b) Melihat pembagian beban pada statik *haproxy*

Sebelumnya kita masuk melalui url statika *haproxy* dengan cara masuk ke *google*, lalu masuk dengan alamat *url*: 172.17.0.10:8080.



Gambar 15. Hasil statistik kinerja loadbalancing haproxy

3. Pengujian transaksi *database insert, delete update* dan *select* saat *loadbalancing* dan *default.*

1) Perbandingan *query* pemanggilan (*select*) data antara *mysql cluster default* dan menggunakan *loadbalancing* terlihat pada tabel yaitu pada saat kondisi *default* lebih memakan waktu yang banyak dibandingan pada kondisi *loadbalancing* yakni pada percobaan 1 sampai 5.

2) Perbandingan *query* memasukkan (*insert*) data antara *mysql cluster default* dan menggunakan *loadbalancing* terlihat pada tabel yaitu pada saat kondisi *default* lebih memakan waktu yang banyak dibandingan pada kondisi *loadbalancing* yakni pada percobaan 1 sampai 5.

3) Perbandingan *query* menghapus (*delete*) data antara *mysql cluster default* dan menggunakan *loadbalancing* terlihat pada tabel yaitu pada saat kondisi *default* lebih memakan waktu yang banyak dibandingan pada kondisi *loadbalancing* yakni pada percobaan 1 sampai 5.

4) Perbandingan *query* mengganti (*update*) data antara *mysql cluster default* dan menggunakan *loadbalancing* terlihat pada tabel yaitu pada saat kondisi *default* lebih memakan waktu yang banyak dibandingan pada kondisi *loadbalancing* yakni pada percobaan 1 sampai 5.

Jadi dari data yang didapat dan sudah dikelola, data menunjukkan kalau durasi waktu yang dihasilkan saat melakukan percobaan transaksi *select*, *insert*, *delete*, dan *update* pada implementai *mysql cluster default* dan *loadbalancing* lebih sedikit durasi waktu yang diperoleh pada waktu transaksi *mysql cluster* dengan metode *loadbalancing*, itu artinya kinerja pada saat *mysql cluster loadbalancing* lebih cepat dari pada saat *mysql cluster non balancing* atau *default*.

4. Pengujian kinerja *Mysql cluster* secara *default* dan kinerja *Mysql Cluster* dengan l*oadbalancing* dengan *sysbench.*

Pengujian ini dengan dengan cara perulangan, jadi maksudnya pada saat *sysbench* melakukan transaksi *readonly* maupun *writeonly*, pada saat transaksi akan diulang dengan perulangan “2 4 8 16 32 64 128 256 512”.

Setelah semua parameter disiapkan, kemudian akan dilakukan pengukuran kinerja. Kemudian dibawah ini adalah hasil pengujian perulangan *thread* melalui *sysbench* pada saat *default* dan *write only* dan *read only*:

Dari uji coba analisa kinerja dengan menggunakan *sysbench*, berikut kesimpulan hasilnya:

1. Write Only

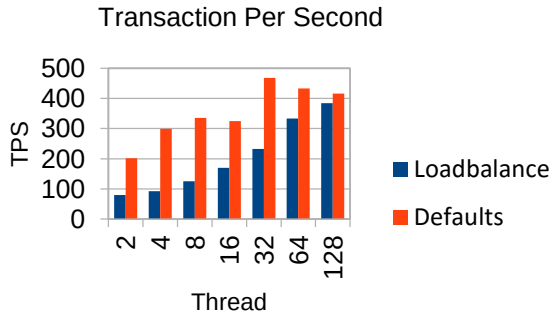
Tabel 1. Hasil analisis kinerja mysql cluster write only

Thread	TPS	
	Loadbalance	Defaults
2	79,60	201,30
4	92,63	298,93
8	125,17	335,49
16	169,45	324,49
32	232,02	468,00
64	333,29	432,92
128	383,70	415,78

Pada uji coba analisis kinerja *mysql cluster default* dan *loadbalancing* dengan *mode write only* ini akan ditampilkan data *Transaction per Second* dari hasil transaksi *write only*. Pada Tabel 1 nilai *Transaction per Second* tersebut diperoleh hasil jumlah kinerja *Transaction per Second* mulai dari perulangan *thread* yang semakin banyak, seperti pada jumlah *Transaction per Second* pada saat *loadbalancing* yaitu mulai dari *thread* pada saat *thread* 2-128 mengalami

kenaikan yang stabil. Proses ini terjadi karena transaksi pada kondisi *loadbalancing* terjadi berdasarkan jumlah *node* yang dijadikan *cluster*. Dan pada jumlah *Transaction per Second* pada saat *default* mengalami kinerja yang tidak stabil dikarenakan dari proses kinerja mengalami penurunan yaitu mulai dari *thread* 16 dan 64-128 yaitu dari jumlah *Transaction per Second* mengalami penurunan pada *thread* 16 yaitu jumlah *Transaction per Second*- nya 324,49, dan mengalami penurunan pada *thread* 64-128 yaitu jumlah TPS nya 432,92-415,78.

Seperti terdapat perbandingan pada tabel dibawah pada saat detik-detik perulangan *thread* semakin tinggi.



Gambar 16. TPS Write only

Selanjutnya menganalisa hasil kinerja *mysql cluster* pada saat *read only* dengan *loadbalancing* dan *default*. Pada uji coba analisis kinerja *mysql cluster default* dan *loadbalancing* dengan *mode read only* ini akan ditampilkan data *Transaction per Second* dari hasil transaksi *read only*. Berikut dibawah data yang diperoleh dari melakukan transaksi kinerja *mysql cluster* dengan metode *loadbalancing* dan *default* dalam *mode read only*.

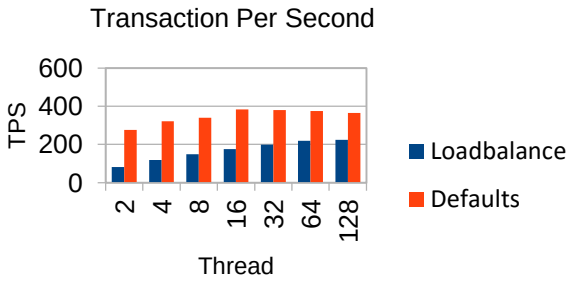
2. Read Only

Tabel 2. Hasil analisis kinerja mysql cluster read only

Thread	TPS	
	Loadbalance	Defaults
2	81,75	275,89
4	118,74	322,43
8	148,81	340,38
16	176,09	383,08
32	198,84	379,86
64	218,44	375,70
128	224,31	365,18

Pada uji coba analisis kinerja *mysql cluster default* dan *loadbalancing* dengan *mode read only* ini akan ditampilkan data *Transaction per Second* dari hasil transaksi *read only*. Pada Tabel 2 nilai *Transaction per Second* tersebut diperoleh hasil jumlah kinerja *Transaction per Second* mulai dari perulangan *thread* yang semakin banyak, seperti pada jumlah *Transaction per Second* pada saat *loadbalancing* yaitu mulai dari *thread* pada saat *thread* 2-128 mengalami kenaikan yang stabil. Proses ini terjadi karena transaksi pada kondisi *loadbalancing* terjadi berdasarkan jumlah *node* yang dijadikan *cluster*. Dan jumlah *Transaction per Second* pada saat *default read only*, dari *thread* ke2-16 mengalami kenaikan yaitu pada *thread* ke 2 adalah 275,89 lalu naik di *thread* ke 4 menjadi 322,43, kemudian pada *thread* 8 naik menjadi 340,38, lalu naik lagi pada *thread* ke 16

Dan untuk mempermudah melihat dengan jelas hasil kinerja dari TPS dengan *mode read only* dapat dilihat pada gambar 17



Gambar 17. TPS *read only*

KESIMPULAN DAN SARAN

Simpulan

Dari pembahasan dan hasil implementasi yang telah dilakukan, berikut kesimpulan yang didapat:

1) Secara fungsional, implementasi *analisis kinerja mysql dengan metode loadbalancing* telah berjalan dengan baik. *Engine database* yang digunakan dalam server *mysql cluster* ini menggunakan *ndbcluster* dimana terdapat 2 *node* dengan cara kerja yang sama yaitu sebagai *master* kedua yang bersifat *syncronisasi*. *Load balancingnya* sendiri menggunakan *ha-proxy* untuk membagi beban *server*. Pembagian server ini dapat diketahui dengan pengguanaan *CPU* dan *Memory* maupun dari *statik haproxy* juga mengguanakan perintah *command prompt* untuk pengujian menambahkan *user* yang nantinya terbagi ke tiap-tiap *server*.

2) Analisa hasil kinerja *mysql cluster* dengan metode *loadbalancing* dan *non loadbalancing* adalah yaitu pengujian secara sinkronisasi sekaligus *high availbility* dan *failure*, data yang dimasukkan pada *server1* begitu sebaliknya sudah dapat tersinkronisasi, dari segi *high avialbility*, jumlah data pada *server1* dan *server* 2 sama sama banyaknya walupun sudah terjadi adanya transaksi pada saat uji *failure* salah satu *node* dimatikan juga jumlah data yang berada pada *server* tersebut tetap sama.

Dalam pengujian *loadbalancing* pada *server1* dan *server2* mengalami pembagian beban bukti pembagian beban pada *CPU* maupun *memory* yang terpakai sama rata dan pada saat *non loadbalancing*, beban *CPU* dan *memory* tidak terbagi dengan merata. Pengujian *query* dengan lebih dari 1 tabel, dengan perbandingan pengujian secara *default* dan *loadbalancing*, pada pengujian tersebut setiap transaksi menghasilkan durasi waktu dan dianalisa menyimpulkan durasi waktu yang dihasilkan pada *loadbalancing* dan *non loadbalancing.* pengujan *benchmark sysbnech* yaitu Analisa kinerja *mysql cluster write only* dengan metode *loadbalancing* dan *non loadbalancing* yang diukuer dengan parameter *transaction per second* yang bertumpu dengan *thread*.

3) Berdasarkan hasil penelitian, dapat disimpulkan bahwa kinerja pada *mysql cluster* yang meliputi pengujian *backup data* sudah berhasil, pengujian *high availbility* yaitu jumlah data yang berhasil dibuah, pada kedua *server* menunjukkan kesamaan yang akurat, selanjutnya pengujian *failure* yaitu salah satu keunggulan *mysql cluster* yang dapat menggantikan salah satu *server database* yang mati agar sistem masih dapat berjalan. Ini berarti kinerja *mysql cluster* sudah berjalan dengan baik dan untuk penambahan *metode loadbalancer* terbukti dengan adanya trafik dikedua *server* yang sudah terbagi rata. Selain itu dalam pengujian transaksi *insert, delete, update, select* dengan lebih dari satu tabel menyimpulkan hasil durasi waktu untuk transaksi lebih sedikit menggunakan *loadbalancing* artinya kinerja lebih cepat menggunakan metode *loadbalancing*. Serta pada pengujian dengan menggunakan aplikasi *benchmark sysbench* dengan pengujian *mode writeonly* dan *readonly* secara *loadbalancer* dan *non balancer* terbukti yaitu dilihat dari analisis penurunan performansi yang terjadi pada *mysql cluster* dapat sedikit ditingkatkan dengan penambahan *loadbalancing*. Penambahan tersebut dapat berjalan dengan baik pada saat *Transaction per Second* pada transaksi *writeonly* dan *readonly loadbalancing.*

SARAN

Dari pembahasan dan hasil implementasi yang telah dilakukan, untuk mengembangkan tugas akhir ini terdapat beberapa hal yang perlu diteliti lebih lanjut. Berikut beberapa halnya:

1. Perancangan *mysql cluster* dengan lingkup yang lebih luas.
2. Arsitektur *mysql cluster* yang lebih baik serta perlunya membahas tentang faktor keamanan
3. Perlu penambahan jumlah server di setiap nodenya agar lebih jelas pengamatan *mysql cluster* itu sendiri.

DAFTAR PUSTAKA

Alex Davies, H. F. (2006). *MySQL Clustering.* MySQL Press.

Analisi Kinerja MySQL Cluster . (2015). Diambil kembali dari Analisi Kinerja MySQL Cluster Website: http://www.e-jurnal.com/2015/11/analisis-kinerja-mysql-cluster.html

How to Create Multi-Node MySQL Cluster On Ubuntu.16.04. (t.thn.). Diambil kembali dari How to Create Multi-Node MySQL Cluster On Ubuntu.16.04 Website: http://www.digitalocean.com/community/tutorials/how-to-create-a--multi-node-mysql-cluster-on-ubuntu-16-04

server-clustering

NDB Cluster Core Concepts. (t.th). Diambil kembali dari web NDB Cluster Core Concepts Website: https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster-basics

NDB Cluster overview. (t.th). Diambil kembali dari web NDB Cluster overview Website: https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster-overview

Mysql NDB Cluster Features & Benefits . (t.th). Diambil kembali dari web Mysql NDB Cluster Features & Benefits Website: https://www.mysql.com/products/cluster/features