PEMANFAATAN TEKNOLOGI OPENSOURCE DALAM MERANCANG INFRASTRUKTUR *HA-CLUSTER SERVER* UNTUK MENGHINDARI *SINGLE POINT OF FAILURE*

Candra Kusuma[1], Suwanto Raharjo[2], Catur Iswahyudi[3]
Program Studi Informatika, Fakultas Teknologi Industri
Institut Sains & Teknologi AKPRIND Yogyakarta
Email: [1]candrakusuma969@gmail.com,[2]wa2n@akprind.ac.id,[3]catur@akprind.ac.id

ABSTRACT

The latest information system requirements encourage the provision of High Availability Clusters (HAC) to reduce the difficulty of overcoming the system recovery process associated with server downtime and produce a reliable system to access the database and simplify the process of synchronizing databases between servers which is carried out automatically by the cluster system.

This research method is carried out independently, with simulation techniques on the database research object so that it can always be accessed by users without being disturbed whether or not the server is active, so a system called high availability is used to avoid single point of failure and to test penetration or stressing the database using The Apache jMeter to get a good Availibility parameter value. In this study using VirtualBox as a simulation platform, the High Availability Cluster is built using two virtual servers with the Debian operating system using MariaDB to store databases. Then Pacemaker / Corosync relates to the network communication infrastructure between the two servers in forming a cluster system with the step set up nodelist communication or can be interpreted as a list of servers referring to the server used as a cluster to synchronize. The parameters measured in this study are Availability, Downtime, Failover Downtime, Throughput.

Based on availability testing, it gets good results with zero downtime availability. It is said to be zero downtime because in 5 trials the availability value is obtained with an average of 99% so that the simulation is carried out smoothly. With a stable level of failover downtime and throughput, so this cluster system can be a solution to improve systems with high levels of availability

Keywords: High Availability Cluster, MariaDB, Single Point of Failure, Failover.

INTISARI

Kebutuhan sistem informasi yang terbaru mendorong menyediakan *High Availability Cluster* (HAC) untuk mengurangi kesulitan penanggulangan proses pemulihan sistem yang terkait dengan *server down* dan menghasilkan sistem yang handal mengakses basis data dan mempermudah proses sinkronisasi basis data antar *server* yang dilakukan otomatis oleh sistem *cluster*.

Metode penelitian ini dilakukan secara mandiri, dengan teknik simulasi pada objek penelitian *database* agar selalu bisa diakses oleh *user* tanpa terganggu aktif tidaknya *server*, maka digunakanlah sistem yang bernama *high availability* untuk menghindari *single point of failure* serta melakukan *penetrasi test* atau *stressing* terhadap *database* menggunakan *The Apache jMeter* untuk mendapatkan nilai parameter *Availibility* yang baik . Pada penelitian ini menggunakan *VirtualBox* sebagai *platform* simulasinya, *High Availability Cluster* yang di bangun menggunakan dua buah *server virtual* dengan sistem operasi *Debian menggunakan* MariaDB untuk menyimpan basis data. Kemudian *Pacemaker/Corosync* berkaitan dengan infrastruktur komunikasi jaringan antar kedua *server* dalam membentuk sistem *cluster* dengan langkah *set up* komunikasi *nodelist* atau bisa diartikan sebagai daftar *server* mengacu pada *server* yang digunakan sebagai *cluster* untuk disinkronkan. Parameter yang diukur pada penelitian ini adalah *Availability, Downtime*, *Failover Downtime, Throughput.*

Berdasarkan pengujian *availability* mendapatkan hasil yang baik dengan tingkat *availability zero downtime*. Dikatakan *zero downtime* karena pada 5 kali percobaan nilai *availability* didapat dengan rata-rata 99% sehingga simulasi yang dilakukan berjalan

lancar.dengan tingkat kestabilan *Failover Downtime* dan *Throughput,* sehingga sistem *cluster* ini dapat menjadi solusi untuk meningkatkan sistem dengan tingkat *availaibility* yang tinggi.

Kata kunci: Ketersediaan Cluster Tinggi, MariaDB, Kegagalan Tunggal*,* Kegagalan*.*

PENDAHULUAN

Perkembangan teknologi yang begitu pesat diikuti dengan perkembangan jaman yang penuh dengan kegiatan akses data untuk memenuhi kebutuhan informasi pengguna*,* menyediakan sebuah infrastruktur yang mumpuni sudah sewajarnya kita sediakan untuk dapat memenuhi kebutuhan pengguna tersebut. Dalam hal ini bisa kita ambil contoh yaitu akses ke media sosial ataupun aplikasi dari sebuah layanan tertentu. Beberapa aplikasi mungkin sering kita temukan yang akses dan responnya lambat, bahkan ketika diakses terus menerus ataupun mendapatkan trafik akses *server* yang tinggi mengakibatkan *server* dari aplikasi tersebut mati, sehingga pengguna tidak dapat melanjutkan kembali aktivitas mereka terkait dengan kebutuhan dari penggunaan layanan tersebut.

High Availability adalah tingkat ketersediaan suatu sistem untuk diakses dan dipergunakan ketika diperlukan. Kata *availability* mengacu kepada kemampuan sekelompok pengguna agar dapat mengakses ke dalam sistem seperti menambah pekerjaan baru, mengubah pekerjaan atau mengumpulkan hasil dari pekerjaan yang sebelumnya. Jika pengguna tidak dapat mengakses sistem, dapat dikatakan sistem tersebut tidak tersedia (*unavailable*). Secara umum *variabel* yang digunakan untuk memutuskan sebuah sistem *available* atau *unavailable* adalah *downtime*.

Oleh karena itu, dibutuhkan perancangan infrastuktur pendukung untuk dapat mampu memberikan kenyamanan kepada pengguna dalam hal menyediakan sebuah layanan dari aplikasi serta menjamin akses pengguna agar dapat selalu berjalan tanpa harus berakhir dengan masalah *server down.*

TINJAUAN PUSTAKA

Penelitian pada *Clustering Database Server* pernah dilakukan oleh (Suryanto 2015), dengan menggunakan sistem (*clustering*) ini, maka *database* akan direplikasi pada server lain, sehingga akan memiliki *server* yang identic pada Implementasi *Clustering Database Server* Menggunakan PGCluster untuk Optimalisasi Kinerja Sistem Basis Data. Metode yang dipakai pada perancangan ini adalah *Cisco lifecycle services* yang merupakan metode yang dirancang untuk mendukung perkembangan jaringan komputer. Hasil dari semua hasil pengujian yang telah dilakukan, hasil analisa dari sistem *load balancing* untuk *Database server* dengan *PGCluster* salah satunya adalah user mampu dibagi ke semua *database server* yang menjadi anggota cluster.

Penelitian tentang perancangan sistem *Cluster Server* pernah dilakukan oleh (Adi. Dkk. 2016). Tujuannya adalah untuk meningkatkan kinerja dari perusahaan, instansi, atau organisasi. Metode pengembangan yang digunakan mengacu pada kerangka kerja metode *Network Development Life Cycle* (NDLC). Berdasarkan hasil pengujian dan analisis, dapat diambil beberapa kesimpulan. Rancang bangun sistem *cluster server* yang dapat memberikan jaminan ketersediaan layanan tinggi di atas lingkungan *virtual* berhasil dibangun dengan menggunakan Proxmox VE 3.4 sebagai *server* virtualisasi pada dua unit komputer, FreeNAS 9.2.1.9 x86 sebagai server NAS, dan dua unit switch Cisco Catalyst 2960.

Penelitian menggunakan *Cluster* juga pernah dilakukan oleh (Aspriyono 2016). Penelitian ini bertujuan untuk memahami desain sistem server yang telah ada dan selanjutnya melakukan analisa masalah-masalah yang ada sebagai dasar pengembangan server untuk mengatasi masalah yang ada pada *server* yang telah ada. Metodologi yang digunakan adalah untuk memahami sesuatu melalui penyelidikan atau usaha mencari bukti-bukti yang muncul sehubungan dengan masalah itu, yang dilakukan secara teliti sehingga diperoleh suatu pemecahan masalah. Hasil dari sistem dapat menjalankan fungsinya baik dalam keadaan normal dan melakukan *failover* ketika terjadi crash/down pada server utama (primary server) dan sebaliknya.

Berdasarkan penelitian-penelitian tinjauan pustaka di atas yang telah digunakan seperti Clustering Database Server, perancangan Cluster Server, dan Failover, maka pada penelitian menggunakan tekmologi opensource dalam merancang infrastruktur HA-Cluster Server untuk menghindari single point of failure.

Landasan teori

Distributed Replicated Block Device (DRBD)

DRBD (*Distributed Replicated Block Device*) adalah *storage block device* yang dirancang untuk membangun *High Availability* sistem. Kinerja DRBD (*Distributed Replicated Block Device*) dilakukan dengan cara *mirroring* terhadap semua melalui jaringan serta menggunakan jaringan sebagai media transmisi untuk melakukan sinkronisasi data antar *disk* secara *realtime*. (Pratama, 2017).

The Apache JMeter™

jMeter atau *The Apache JMeter™* adalah aplikasi *open source* berbasis Java yang dapat dipergunakan untuk *performance test* . Bagi seorang QA *Engineer* jMeter bisa digunakan untuk melakukan *load/stress* testing *Web Application*, FTP *Application* dan *Database server test.* Dalam penelitian ini menggunakan rumus yang tentukan ITU (*international telecommunication union*).

$$Availability = \frac{PotentialUserMinutes - UserOutageMinutes}{PotentialUserMinutes} \times 100\% \quad \text{.....(1)}$$

$$Availability = \frac{AST - DT}{AST} \times 100\% \quad \text{.....(2)}$$

High Availability

High Availability adalah tingkat ketersediaan suatu sistem untuk diakses dan dipergunakan ketika diperlukan. Kata *availability* mengacu kepada kemampuan sekelompok pengguna agar dapat mengakses ke dalam sistem seperti menambah pekerjaan baru, mengubah pekerjaan atau mengumpulkan hasil dari pekerjaan yang sebelumnya. Jika pengguna tidak dapat mengakses sistem, dapat dikatakan sistem tersebut tidak tersedia (*unavailable*). Secara umum *variabel* yang digunakan untuk memutuskan sebuah sistem *available* atau *unavailable* adalah *downtime*. (Domaschka. Dkk. 2014).

PEMBAHASAN

Database availability diukur berdasarkan metode *benchmark database stressing* dengan menggunakan aplikasi *Apache jMeter*. Parameter yang diujikan adalah dengan menggunakan penghitungan *PotentialUserMinutes* dan *UserOutageMinutes*. Simulasi *downtime* nantinya ditujukan untuk mengetahui dampak presentase yang akan digunakan sebagai acuan untuk mengetahui *productivity service database* yang tersedia seperti pada gambar .

Pengujian *database availability* akan dilakukan baik saat sistem berjalan normal maupun ketika simulasi kegagalan sistem terjadi baik ketika salah satu *cluster server* mati maupun *service database* dinonaktifkan. Proses analisa pengujian adalah beradasarkan kondisi tersebut. Adapun metode perhitungan yang digunakan adalah berdasarkan rumus(1).

a. *Potential User Minutes (PUM)*

Merupakan jumlah total user dikali lamanya waktu mereka bekerja (tetap mendapat akses database). Misalnya, jika memiliki 10 user yang bekerja selama 8 jam maka nilai *PotentialUserMinutes* adalah 10x8x60

.

b. *User Ooutage Minutes (UOM)*
Merupakan jumlah total user yang tidak dapat bekerja (akses database tidak berhasil), dikalikan dengan saat waktu *user* tidak dapat bekerja. Misalnya, jika sebuah insiden atau kegagalan sistem mencegah 5 orang bekerja selamat 10 menit maka nilai *UserOutageMinutes* adalah 5 x 10.

Hasil pengujian dilakukan sebanyak 5 kali percobaan dengan record data berjumlah 1000 *records* dengan nilai *UOM* diambil dari hasil deviasi dari aplikasi *Apache jMeter* sehingga menghasilkan data yang terlihat pada Tabel 1.

Tabel 1 Hasil pengujian *database availability*

No	Pengujian	User	PUM	UOM	Availability
1	Pertama	10	4800	9 (1 menit)	99,81%
2	Kedua	50	24000	5,2 (1,2 menit)	99,97%
3	Ketiga	100	48000	4,2 (1.4 menit)	99,99%
4	Keempat	300	144000	2,25 (1,5 menit)	99,99%
5	Kelima	500	240000	3,6 (1,8 menit)	99,99%

Berdasarkan data pengujian pada Tabel 1 menghasilkan data *availability* yang baik dengan tingkat *availability zero downtime*. Dikatakan *zero downtime* karena pada 5 kali percobaan nilai *availability* didapat dengan rata-rata 99% sehingga simulasi yang dilakukan berjalan lancar. Semakin tinggi nilai *availability* yang didapat maka semakin baik pula *server* dalam merespon akses dari *user*. Nilai-nilai di atas juga sangat berpengaruh terhadap perhitungan *PUM* dan *UOM* pada saat simulasi pengujian dilakukan, meskipun jumlah *user* yang ditetapkan pada setiap pengujian berbeda, pada akhirnya *UOM* akan menjadi penentu dari nilai akhir *availability* yang didapat.

1. Analisis *Downtime Availability*

Analisis *downtime availability* dilakukan dengan simulasi mematikan salah satu *cluster server* yang aktif maupun menonaktifkan *service database*. *Downtime* akan berpengaruh terhadap *delay* sinkronisasi data *database*. Metode perhitungan yang digunakan akan mengacu pada simulasi waktu *downtime* dan *Agreed Service Time (AST)*. Menggunakan rumus (2)

a. *Agreed Service Time (AST)*
Merupakan waktu atau periode dimana layanan seharusnya tersedia. Misalnya, layanan yang ditawarkan mulai pukul 08:00 hingga 19:00 Senin hingga Jumat, 52 minggu per tahun, akan memiliki *AST* 11x5x52 = 2860 jam per tahun.

b. *Downtime (DT)*
Merupakan waktu dimana sistem tidak dapat digunakan untuk menjalankan sistem.

Dalam pengujiannya, *AST* ditetapkan nilai yaitu 100 jam (bersifat opsional untuk simulasi), begitu juga untuk nilai *Downtime* akan mengacu terhadap waktu (satuan jam) dari proses pengujian dilakukan. Tabel 1 adalah hasil pengujian *downtime* yang didapat dengan melakukan 5 kali percobaan.

.

Tabel 2 Hasil pengujian *downtime*

No	Pengujian	AST (jam)	DT (jam)	Availability
1	Pertama	100	0,08 (5 menit)	99,92%
2	Kedua	100	0,33 (20 menit)	99,67%
3	Ketiga	100	0,50 (30 menit)	99,50%
4	Keempat	100	0.66 (40 menit)	99,34%
5	Kelima	100	1	99%

Hasil pengujian seperti Tabel 2 di atas menunjukkan bahwa *availability* sangat berpengaruh terhadap waktu *downtime.* Akan tetapi waktu *downtime* pada simulasi di atas akan berbeda jika mengacu pada ketentuan nilai nyata dari *AST* maupun *DT* yang sebenarnya. Umumnya sebuah organinsasi IT sebagai contoh telah memiliki nilai *AST* yang ditetapkan, sehingga perhitungan *downtime* akan mendapatkan hasil yang lebih nyata. Hasil *availability* pada tabel 2 di atas menghasilkan rata-rata 99% yang mana hasil ini pada dasarnya hanyalah hasil dari simulasi, karena nilai *AST* ditetapkan secara opsional. Namun sama halnya dengan pengujian *database availability* yang jika hasil *availability* yang didapat semakin tinggi maka *server* akan selalu dapat memberikan akses ke *user* tanpa gangguan.

2. Analisis *Failover Downtime*

Analisis yang dilakukan berdasarkan data yang didapat melalui aplikasi *Apache jMeter* dengan langkah yang mirip dengan analisis *database availability* namun lebih berfokus kepada *timing* dari *failover.* Pengujian masih dilakukan sebanyak 5 kali untuk simulasi kegagalan sistem terjadi baik ketika salah satu *cluster server* mati maupun *service database* dinonaktifkan. Hasil dari pengujiannya dapat dilihat pada Tabel

Tabel 3 Hasil pengujian *failover downtime*

No	Kondisi Server	Failover (menit)	Fallback (menit)
1	Server Down	0,86 (52 detik)	0,85 (50 detik)
2	Cluster Offline	0,16 (10 detik)	0,11 (7 detik)
3	Server UP	0,33 (20 detik)	0,33 (20 detik)
4	Cluster Online	0,26 (16 detik)	0,25 (15 detik)
5	Server Reboot	0,33 (20 detik)	0,15 (9 detik)

Data yang didapat pada Tabel 3 di atas menunjukkan hasil *delay failover* dan *fallback* yang tidak cukup jauh satu dengan yang lain. Khusus untuk kondisi *Server Up* akan selalu tetap sama karena sudah di konfigurasi sebelumnya dengan nilai konstan yaitu 20 detik dan nilai ini dapat ditentukan sesuai dengan kebutuhan.

Waktu *fallback* tercepat adalah ketika salah satu *cluster server offline*, jeda waktu cukup singkat untuk akses data didapatkan kembali oleh *user.* Kemudian waktu yang cukup lama ialah ketika *Server Down* dengan nilai *failover* dan *fallback* yang hampir sama, sehingga *user* akan lebih merasakan efek dari jeda yang didapatkan. Untuk nilai-nilai yang didapatkan pada Tabel 3 di atas sangat berpengaruh dengan spesifikasi dan kondisi *hardware* yang digunakan. Jika waktu delay yang didapat semakin kecil, tentunya respon *server* akan semakin baik. Tetapi khusus untuk kondisi *Cluster Up* maupun *Cluster Reboot* sangat disarankan untuk memberikan waktu *interval* untuk *cluster resource* melakukan proses sinkronisasi dengan tujuan untuk menghindari *data corrupt* didalam *database*.

3. Analisis *Throughput*

Pengujian analisis *throughput* adalah berdasarkan pengujian analisis *database availability* dengan menggunakan aplikasi *Apache jMeter* seperti pada contoh Gambar 4. Percobaan dilakukan masih sejumlah 5 kali dengan beban *database stressing* yang sama. Selain nilai *throughput*, nilai data yang didapatkan juga bisa menjadi acuan. Hasil yang didapatkan dari pengujian *throughput* dapat dilihat pada Tabel 4.

Tabel 4 Hasil analisis *throughput*

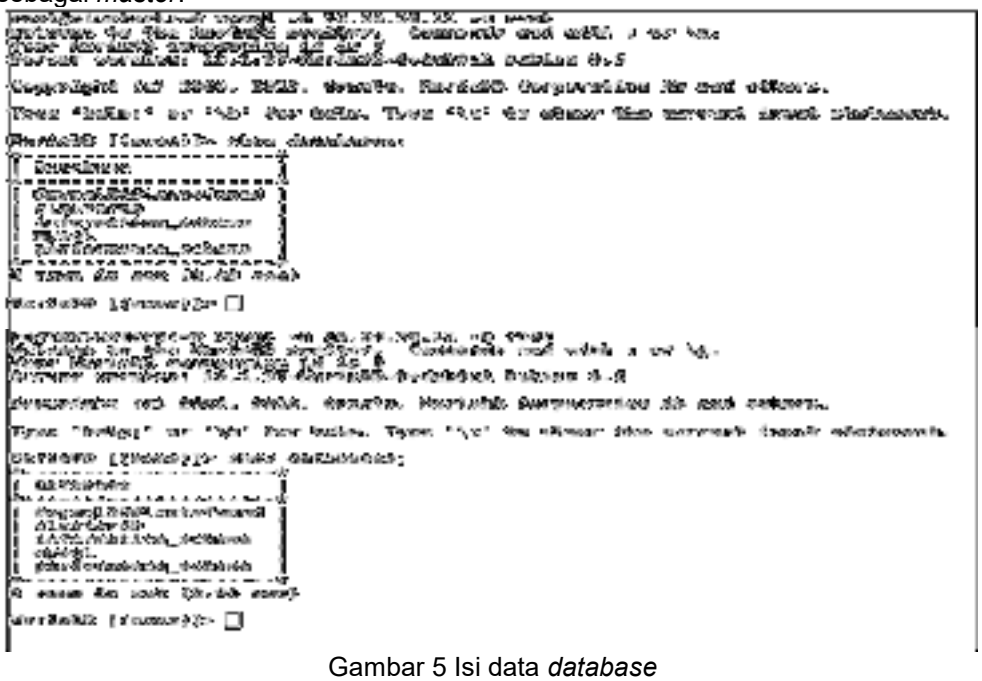
No	Pengujian	User	Throughput	Received KB/sec
1	Pertama	10	11.09336	129.87
2	Kedua	50	26.60483	311.46
3	Ketiga	100	100.88781	1181.1
4	Keempat	300	300.95101	3523.24
5	Kelima	500	500.41534	5858.38

Hasil pengujian analisis *throughput* pada Tabel 4 di atas disimulasi dengan jumlah koneksi untuk setiap langkah pengujian sejumlah maksimal 100 koneksi. Jumlah *user* menjadi pengaruh untuk hasil *throughput* yang didapatkan, sehingga semakin banyak *user* maka semakin besar juga data yang akan direspon oleh *server*. Pada tabel 4 di atas dapat terlihat bahwa *throughput* dari jumlah *user* terkecil hingga jumlah *user* tertinggi masih dapat merespon sesuai dengan jumlah *request* yang diterima *server*. Untuk nilai *throughput* yang terus meningkat menandakan bahwa performa *server* dalam memberikan respon sangat bagus.

4. Pengujian *Sample* Data *MariaDB*

Pengujian *sample* data *MariaDB* akan dilakukan dengan menggunakan *MariaDB shell* langsung di dalam *Server.* Untuk memastikan *database* selalu dapat diakses, maka pengujian dilakukan juga dengan mematikan salah satu *Server* yang aktif kemudian melakukan input data ke dalam *database*. Status awal adalah *Server* 1 aktif sebagai *master* sesuai dengan yang terlihat pada gambar 5 sebelumnya.

Data *database* yang tersedia antar kedua *Server* adalah sama karena belum ada modifikasi sebelumnya. Pengujian akan dilakukan dengan membuat *database* baru beserta tabel dan isi tabel dengan terlebih dahulu mematikan salah satu *Server* yaitu *Server 1* yang aktif sebagai *master*.



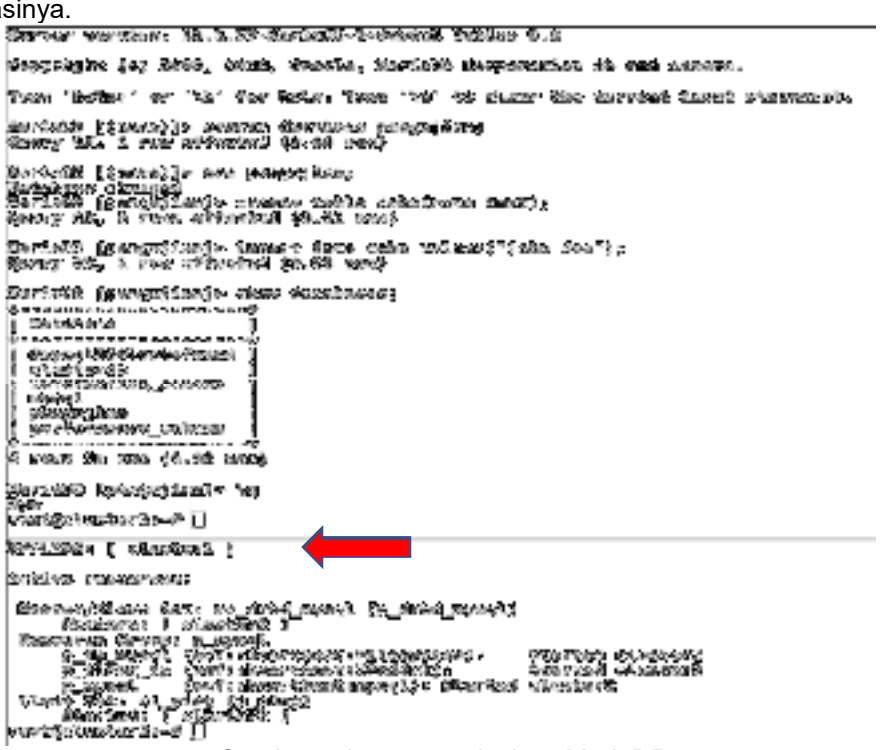
Gambar 5 Isi data *database*

```
root@cluster2:~# crm_mon -1
Stack: corosync
Current DC: cluster2 (version 1.1.16-94ff4df) - partition with quorum
Last updated: Mon Jul  1 20:05:58 2019
Last change: Sun Jun 30 23:13:06 2019 by root via crm_resource on cluster1

2 nodes configured
7 resources configured

Online: [ cluster2 ]
OFFLINE: [ cluster1 ]

Active resources:

 Master/Slave Set: ms_drbd_mysql [p_drbd_mysql]
     Masters: [ cluster2 ]
 Resource Group: g_mysql
     p_fs_mysql (ocf::heartbeat:Filesystem):    Started cluster2
     p_float_ip (ocf::heartbeat:IPaddr2):       Started cluster2
     p_mysql    (ocf::heartbeat:mysql): Started cluster2
 Clone Set: cl_ping [p_ping]
     Started: [ cluster2 ]
root@cluster2:~#
```

Gambar 6 Status *state cluster Server master*

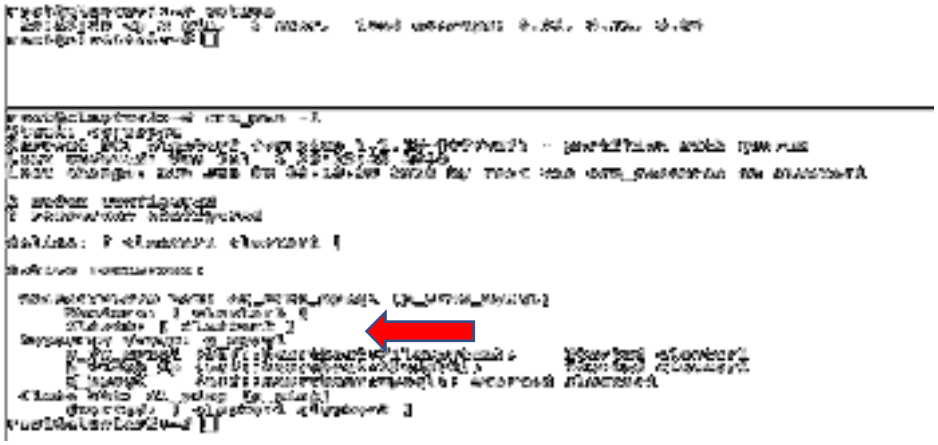
Pada Gambar 6 terlihat bahwa *Server 1* berada dalam kondisi mati (*Offline*) sehingga *Server 2* akan langsung otomatis menjadi sebagai *master* (tanda lingkaran merah).

Pada *state* ini *input sample* data tetap dapat dilakukan, nantinya *Server 1* akan secara otomatis mendapatkan data terbaru setelah dinyalakan kembali. Proses ini di *handle* oleh sinkronisasi *DRBD* yang dibantu oleh *Corosync/Pacemaker* dalam menentukan jalur komunikasinya.



Gambar 7 *Input sample* data *MariaDB*

Gambar 7 menunjukkan proses *input sample* data yang dilakukan berjalan normal tanpa *error*. Proses *input* dilakukan melalui *Server 2* yang sedang aktif. Pada *state* ini *Server* 1 masih dalam kondisi mati (tanda panah merah).

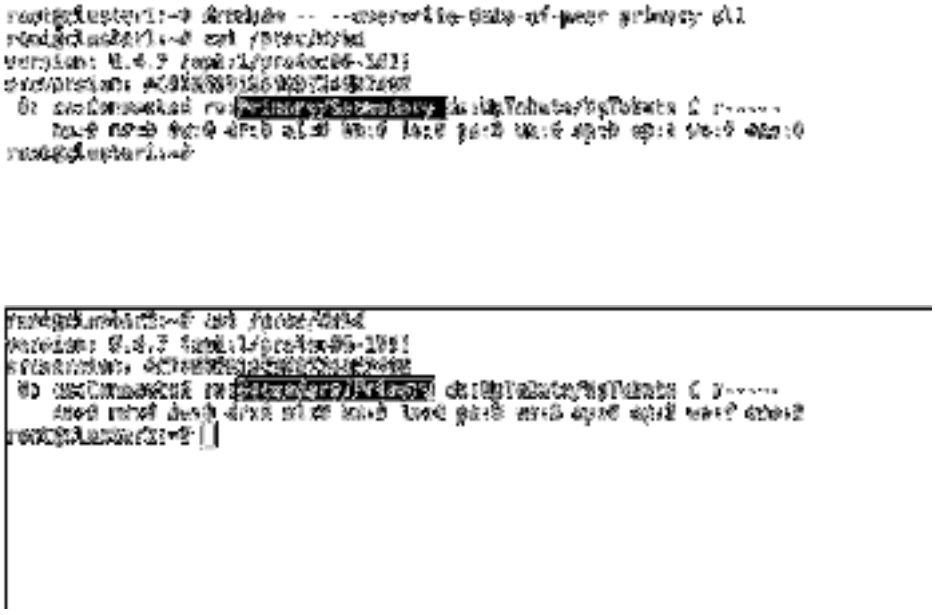


Gambar 8 *Status cluster state Server 1*

Pada gambar 8 dapat terlihat setelah *Server 1* aktif kembali *Server 1* akan otomatis masuk ke dalam grup *cluster* dan berubah status menjadi *online*, (tanda panah merah), proses ini memakan waktu dengan *delay* cukup singkat berdasarkan *set up ping cluster resource* sebelumnya dengan *interval* yang telah ditentukan.

5. Analisis Terhadap Hasil Pengujian

Berdasarkan pengujian dari sistem *HA Cluster Database* yang telah diimplementasikan, dapat dilihat bahwa implementasi sistem berhasil berfungsi dengan baik.



Gambar 9 Status *service DRBD*

Dalam sistem *HA Cluster* yang berjalan, kedua *Server* dapat merespon sinkronisasi data yang di *input* dari *shell database.* Hal tersebut dapat dilihat melalui masing-masing *shell database* seperti yang ditunjukkan pada gambar 9 Dari hasil yang ditunjukkan pada gambar tersebut, menandakan sinkronisasi data dari *database* berjalan dengan baik tanpa adanya kendala, baik saat salah satu *Server* mati maupun saat kedua *Server* dalam keadaan aktif.

Kemudian dari pengujian penggunaan terhadap sistem *HA Cluster* yang dibuat tersebut, didapatkan hasil yang menunjukkan bahwa ketersediaan data akan selalu dapat

diakses oleh *user* dengan adanya *failover IP* sebagai pintu masuk untuk akses *database*. Sistem semacam ini memberikan manfaat yang menjanjikan bagi perusahaan maupun instansi karena bagi *user* yang menggunakan sebuah layanan akan terhindar dari gangguan, karena data selalu dapat diakses. Terlebih lagi sistem ini menggunakan *opensource software* yang akan mengurangi pengeluaran untuk memenuhi kebutuhan infrastruktur yang akan dibangun.

KESIMPULAN

Dalam sistem *HA Cluster* yang berjalan, kedua *Server* dapat merespon sinkronisasi data yang di *input*, menandakan sinkronisasi data dari *database* berjalan dengan baik tanpa adanya kendala, baik saat salah satu *Server* mati maupun saat kedua *Server* dalam keadaan aktif. Hasil yang menunjukkan bahwa ketersediaan data akan selalu dapat diakses oleh *user* dengan adanya *failover IP* sebagai pintu masuk untuk akses *database*. Sistem semacam ini memberikan manfaat yang menjanjikan bagi perusahaan maupun instansi karena bagi *user* yang menggunakan sebuah layanan akan terhindar dari gangguan, karena data selalu dapat diakses Berdasarkan pengujian *availability* mendapatkan hasil yang baik dengan tingkat *availability zero downtime*. Dikatakan *zero downtime* karena pada 5 kali percobaan nilai *availability* didapat dengan rata-rata 99% sehingga simulasi yang dilakukan berjalan lancar. F*ailover* dan *fallback* sangat berpengaruh dengan spesifikasi dan kondisi *hardware* yang digunakan. Jika waktu delay yang didapat semakin kecil, tentunya respon *server* akan semakin baik. Tetapi khusus untuk kondisi *Cluster Up* maupun *Cluster Reboot* sangat disarankan untuk memberikan waktu *interval* untuk *cluster resource* melakukan proses sinkronisasi dengan tujuan untuk menghindari *data corrupt* didalam *database*.

DAFTAR PUSTAKA

Adi, R, Y. Nurhayati, D, O. Widianto, D, E. (2016). Perancangan Sistem Cluster *Server* Untuk Jaminan Ketersediaan Layanan Tinggi pada Lingkungan Virtual. *JNTETI*. Vol. 5, No. 2. Hal 69-77.

Arianto, E. Sholeh, M. Nurnawati, E. K. (2014). Implementasi Load Balancing Dua Line ISP Menggunakan Mikrotik RouterOS Studi Kasus Sistem Jaringan LAN di PT. Wahana Semesta Bangka (Babel Pos)", Jurnal Jarkom, vol.1, no.2. Hal 57-61.

Aspriyono, H. (2016). Pengembangan *Server* SIAKAD Universitas Dehasen Bengkulu Menggunakan High Availability Clustering dan MYSQL Database Replication. *Jurnal Ilmu Komputer.* Vol. 1, No. 2. Hal 104-113.

Bartholomew. D. (2015). *Getting* Started with MariaDB. Packt Publishing Ltd. Birmingham.

DOMASCHKA, J., HAUSER, C.B. & ERB, B. 2014. Reliability and Availability Properties of Distributed Database Systems. *In 2014 IEEE 18th International Enterprise Distributed Object Computing Conference*. IEEE, hal. 226–233.

Pratama, A.(2017). *MySQL Uncover Panduan Belajar MySQL dan MariaDB untuk Pemula*. Bandung. Duniailkom.

Suryanto. (2015). Implementasi Clustering Database *Server* Menggunakan PGCluster untuk Optimalisasi Kinerja Sistem Basis Data. *Jurnal Teknik Komputer AMIK BSI*. Vol. 1, No. 1.Hal 134-143.

Towidjojo, R. (2016). *Mikrotik Kung Fu : Kitab 1 .* Jakarta. Jasakom.

Vugt, V, S. (2014). *Pro Linux High Availability Clustering*. New York. Heinz Weinheimer