Perancangan dan Pembangunan Sistem *Failover* Pada MySQL Menggunakan *Heartbeat* dan MySQL Native Replication untuk Menunjang Ketersediaan Data *Online*

Prajna Deshanta Ibnugraha

Jurusan Teknologi Informasi, Politeknik Telkom Bandung
prj@politekniktelkom.ac.id

Abstrak

Sistem basis data merupakan unsur penting untuk membangun aplikasi *online*. Aplikasi *online* tidak bekerja dengan baik jika sistem basis data terjadi kegagalan. Solusi untuk masalah ini adalah dengan menggunakan sistem *failover*. Sistem *failover* digunakan untuk mendukung ketersediaan data dengan sistem basis data cadangan jika sistem basis data utama sedang bermasalah. Sistem *failover realtime* di MySQL dapat dibangun dari kombinasi antara hearthbeat dan MySQL native. Pengujian *failover* dan sinkronisasi pada sistem yang dibangun dengan cara mematikan *server* basis data utama. *Response time query* digunakan sebagai parameter pengujian performansi. Dari hasil pengujian didapat bahwa sistem *single server* dengan sistem *server* berbasis *failover* memiliki performansi yang tidak terlalu jauh. Sehingga dapat disimpulkan bahwa sistem *failover* model ini layak digunakan sebagai solusi kegagalan *server*.

Kata Kunci: failover, basis data, replikasi

Abstract

***Abstract* – Database system is important element to build online application. Online application does not work properly if the database system gets failure. The solution for the problem is using failover system. Failover system is used for supporting availability of data with backup database system if the main database system is down. Realtime failover system in MySQL can be built from combination between hearthbeat and MySQL Native Replication. Failover and synchronization functions can be tested by turning off the main of database server. Response time query is used for performance testing parameter. From the testing, single server system and server based failover system has a small performance differences. It can be concluded that this failover system model is feasible to be used as a solution for server failure.**

Keywords: failover, database, replication

1. Pendahuluan

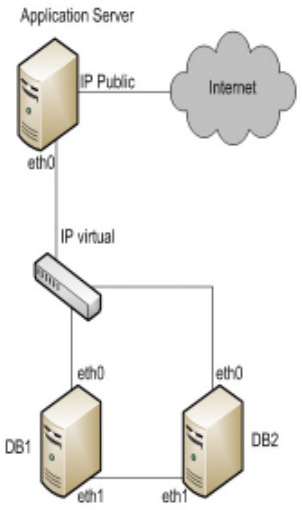
Penggunaan basis data MySQL sebagai tempat pemyimpanan data digital yang besar dan terstruktur memudahkan pengguna dalam pencarian data[1]. Oleh karena itu penggunaan basis data menjadi hal yang sangat penting dalam pembangunan aplikasi *online*.

Sistem aplikasi *online* memudahkan *customer* untuk mengakses layanan tanpa terbatas waktu[2]. Namun harus ditunjang dengan ketersediaan data yang terbatas waktu dan siap setiap saat.

Masalah muncul ketika terjadi kegagalan *server* sistem basis data. Oleh karena itu perlu mekanisme penanganan kegagalan (*failover*)[3]. Heartbeat merupakan aplikasi yang dapat dimanfaatkan untuk melakukan *failover*. Heartbeat dapat dikombinasikan dengan replikasi *native* pada MySQL sehingga menghasilkan 2 sistem basis data dengan identik dan dapat saling tersinkronisasi.

2. Arsitektur Sistem

A. Topologi Sistem



Gambar 1. Topologi sistem

Kebutuhan perangkat keras:

1) *Server* aplikasi *online*
2) *Server* DB1 dengan 2 NIC
3) *Server* DB2 dengan 2 NIC
4) *Switch*

Kebutuhan perangkat lunak:

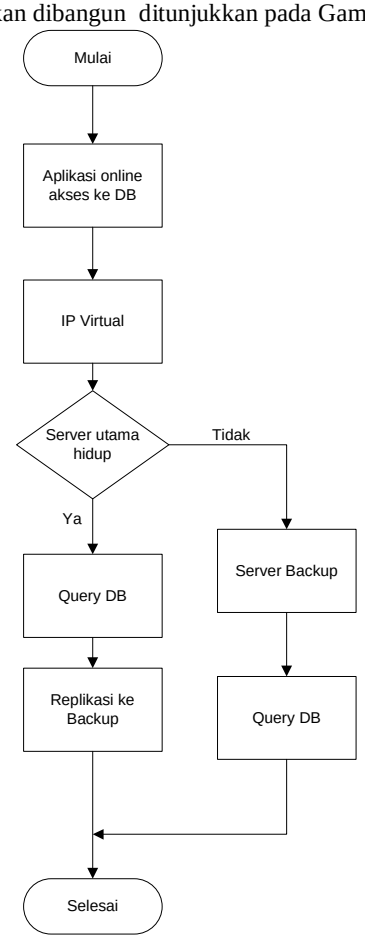
1) MySQL
2) Hearthbeat

TABEL 1
PENGALAMATAN SISTEM

Perangkat	IP Address
Eth0 DB1	192. 168.1.1/29
Eth0 DB2	192. 168.1.2/29
IP *virtual*	192. 168.1.3/29
Eth0 *Application Server*	192.168.1.4/29
Eth1 DB1	192.168.1.9/30
Eth1 DB2	192.168.1.10/30

B. Cara Kerja

Cara kerja sistem *failover* dan replikasi yang akan dibangun ditunjukkan pada Gambar 2.



Gambar 2. Alur kerja sistem

Aplikasi *online* mengakses ke basis data MySQL menggunakan alamat *IP Virtual* yang dibentuk oleh *hearthbeat*. Jika *server* utama yaitu DB1 tidak bermasalah, maka *query* basis data akan berjalan normal dan *server* utamajuga akan mereplikasi setiap perubahan datanya ke *server* cadangan yaitu DB2. Namun jika *server* utama terjadi masalah maka IP *virtual* akan terbentuk di *server* cadangan (DB2).

3. Pembangunan Sistem

A. Heartbeat

Heartbeat difungsikan untuk *failover* dengan membentuk *IP virtual*. *File* konfigurasi heartbeat adalah :

a) ha.cf

```
logfile /var/log/ha-log
logfacility local0
keepalive 2
deadtime 5
initdead 120
bcast eth0
udpport 694
auto_failback on
node DB1
node DB2
```

Keterangan :

DB1 = *hostname server* utama basis data

DB2 = *hostname server* kedua basis data

b) haresources

```
DB1      IPaddr::192.168.1.3/29/eth0
apache mysql
```

Keterangan :

192.168.1.3 = *IP Address virtual* yang digunakan sebagai *IP Address database* untuk aplikasi

eth0 = NIC yang digunakan untuk komunikasi *failover*

c) authkeys

```
auth 2
2 sha1 test-ha
```

Keterangan :

Auth 2 = kunci yang digunakan untuk mengizinkan paket keluar

2 sha1 = jenis autentikasi yang digunakan

test-ha = simple shared key yang digunakan oleh kedua server

B. MySQL Native Replication

MySQL native replication digunakan untuk melakukan sinkronisasi data antara DB1 dengan DB2[4]. Konfigurasi dilakukan di mesin DB1 dan DB2.

a) Konfigurasi my.cnf DB1

```
log-bin=mysql-bin

# identitas penamaan untuk  server
pertama
server-id    = 1

#Untuk mencegah  infinite loops
antar  server
replicate-same-server-id = 0
#jumlah total server database
replikasi
auto-increment-increment = 2

#nilai titik awal dari
auto_increment server pertama
auto-increment-offset = 1

#alamat IP  address  dari server
kedua
master-host = 192.168.1.10

#nama user login database server
pertama
master-user = root

#password database server pertama
master-password = rahasia

#waktu yang dibutuhkan  database
server  pertama
untuk mencoba connect ke database
server kedua jika koneksi terputus
master-connect-retry = 60

#nama database pertama yang
direplikasi ke server kedua
binlog_do_db = contohdb

#nama database kedua yang
direplikasi ke server kedua
binlog_do_db = cthdb

#nama database pertama yang
direplikasi dari server kedua
replicate-do-db = contohdb

#nama database kedua yang
direplikasi dari server kedua
replicate-do-db = cthdb
```

b) Konfigurasi my.cnf DB2

```
log-bin=mysql-bin

## identitas penamaan untuk  server
kedua
server-id    = 2

#Untuk mencegah  infinite loops
antar  server
replicate-same-server-id = 0

#jumlah total server database
replikasi
auto-increment-increment = 2

#nilai titik awal dari
auto_increment dari server kedua
auto-increment-offset = 2

#alamat IP  address  dari server
pertama
master-host = 192.168.1.9

#nama user login database server
kedua
master-user = root

#password database server kedua
master-password = rahasia

#waktu yang dibutuhkan  database
server  pertama
untuk mencoba connect ke database
server kedua jika koneksi terputus
master-connect-retry = 60

#nama database pertama yang
direplikasi ke server pertama
binlog_do_db = contohdb
#nama database kedua yang
direplikasi ke server pertama
binlog_do_db = cthdb

#nama database pertama yang
direplikasi dari server pertama
replicate-do-db = contohdb

#nama database kedua yang
direplikasi dari server pertama
replicate-do-db = cthdb
```

c) Konfigurasi MySQL DB1

```
mysql>FLUSH TABLES WITH READ LOCK;

mysql>SHOW MASTER STATUS;
```

Hasil dari perintah ***SHOW MASTER STATUS*** akan menampilkan kolom File dengan isi **mysql-bin.000019** dan kolom ***Position*** dengan isi 106. Isi kolom ***File*** tersebut akan dimasukkan dalam parameter ***MASTER_LOG_FILE*** di *server* DB2. Sedangkan isi kolom ***Position*** akan dimasukkan ke dalam parameter ***MASTER_LOG_POS*** di *server* DB2. Kemudian lanjutkan dengan perintah berikut :

```
mysql>UNLOCK TABLES;

mysql>LOAD DATA FROM MASTER;

mysql>SLAVE STOP;
```

Untuk perintah selanjutnya parameternya diisi dari hasil perintah ***SHOW MASTER STATUS*** di *server* DB2:

```
mysql>CHANGE MASTER TO
MASTER_HOST='192.168.1.10',MASTER_USER='
root',MASTER_PASSWORD='rahasia',MASTER_L
OG_FILE='mysql-
bin.000003',MASTER_LOG_POS=106;

mysql>SLAVE START;
```

d) Konfigurasi MySQL DB2

```
mysql>FLUSH TABLES WITH READ LOCK;

mysql>SHOW MASTER STATUS;

mysql>UNLOCK TABLES;
```

Hasil dari perintah ***SHOW MASTER STATUS*** akan menampilkan kolom ***File*** dengan isi **'mysql-bin.000003** dan kolom ***Position*** dengan isi 106. Isi kolom ***File*** tersebut akan dimasukkan dalam parameter ***MASTER_LOG_FILE*** di *server* DB1. Sedangkan isi kolom ***Position*** akan dimasukkan ke dalam parameter ***MASTER_LOG_POS*** di *server* DB1. Kemudian lanjutkan dengan perintah berikut :

```
mysql>LOAD DATA FROM MASTER;

mysql>SLAVE STOP;
```

Untuk perintah selanjutnya parameternya diisi dari hasil perintah ***SHOW MASTER STATUS*** di *server* DB1:

```
mysql>CHANGE MASTER TO
MASTER_HOST='192.168.1.9',MASTER_USER='r
oot',MASTER_PASSWORD='rahasia',MASTER_LO
G_FILE='mysql-
bin.000019',MASTER_LOG_POS=106;

mysql>SLAVE START;
```

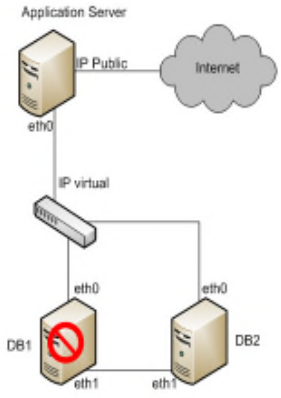
4. Pengujian Sistem

A. Skenario Pengujian Sistem.

Skenario pengujian sistem dilakukan pada dua kondisi berikut:

1) ***Failover***.

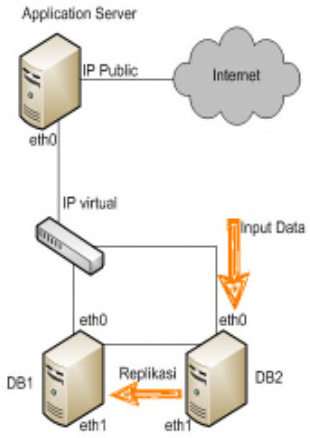
Salah satu skenario pengujian yang dapat dilakukan adalah dengan mematikan mesin DB1. Sehingga layanan basis data akan dilakukan oleh mesin DB2. Ketika mesin DB1 hidup, maka layanan basis data akan kembali ke mesin DB1.



Gambar 3. Topologi Skenario *Failover*

2) **Sinkronisasi data**.

Salah satu skenario pengujian yang dapat dilakukan adalah dengan melakukan penambahan data pada DB1. Ketika data ditambahkan di DB1 maka data akan di DB2 akan bertambah secara otomatis dan identik dengan data yang ditambahkan di DB1. Karena replikasi yang dirancang adalah *master to master replication* maka pengujian sebaliknya perlu dilakukan dengan menambahkan data melalui DB2 sehingga data di DB1 juga akan bertambah dan bersifat identik dengan data DB2 yang ditambahkan.



Gambar 4. Topologi skenario replikasi dan sinkronisasi data

Hasil pengujian sistem untuk skenario diatas adalah sebagai berikut [5]:

a. Skenario *Failover*

Ketika mesin server DB1 dimatikan, penggunaan aplikasi *online* berbasis database MySQL tetap bisa berjalan karena kebutuhan data berhasil dilayani oleh server DB2.

b. Skenario Sinkronisasi Data

Ketika mesin *server* DB1 masih dalam kondisi mati dan kebutuhan data dilayani oleh *server* DB2. Input data dari aplikasi tetap berhasil dijalankan dan data tersimpan di *server* DB2. Selanjutnya ketika mesin *server* DB1 dan layanan MySQL kembali hidup, *server* DB2 akan melakukan replikasi ke *server* DB1 secara otomatis. Setelah dilakukan pengecekan menggunakan perintah ***SELECT*** pada isi basis data, ternyata data antara *server* DB1 dan DB2 telah berhasil sinkron.

B. Pengujian Kinerja

Pengujian kinerja dilakukan setelah sistem yang dibangun berjalan dengan baik. Parameter yang diuji adalah *response tine query*. Berikut hasil

pengujian *response time query* yang telah dilakukan dalam satuan detik:

TABEL 2
HASIL PENGUJIAN *RESPONSE TIME QUERY* TANPA REPLIKASI DAN FAILOVER

No	Jenis Eksekusi	Jumlah Eksekusi	Rata-rata RTTRF
1	Select	15 kali	0,0917
2	Update	10 kali	0,0836
3	Delete	2 kali	0,0858
4	Insert	3 kali	0.0891

TABEL 3
HASIL PENGUJIAN *RESPONSE TIME QUERY* KONDISI REPLIKASI DAN FAILOVER

No	Jenis Eksekusi	Jumlah Eksekusi	Rata-rata RTKRF
1	*Select*	15 kali	0,1063
2	*Update*	10 kali	0,0977
3	*Delete*	2 kali	0,0914
4	*Insert*	3 kali	0,1067

Keterangan :

RTTRF : *Response Time* Tanpa Replikasi dan *Failover*

RTKRF : *Response Time* Kondisi Replikasi dan *Failover*

Pengujian dilakukan sebanyak 30 kali dengan 4 jenis eksekusi yaitu *select, update, delete,* dan *insert*. Untuk rata-rata *response time* masing-masing kondisi adalah sebagai berikut:

TABEL 4
RATA-RATA *RESPONSE TIME* TIAP KONDISI PENGUJIAN

Kondisi	Rata-Rata (detik)
RTTRF	0,0876
RTKRF	0,1005

Selisih *response time query* (ΔRTQ) antara RTTRF dengan RTKRF adalah

$$\Delta RTQ = | AVR(RTTRF) - AVR(RTKRF) |$$
$$= | 0,0876 - 0,1005 |$$
$$= 0,0130 \text{ detik}$$

Jadi perbedaan *response time* dengan jenis *query* dengan besar data yang sama persis antara sistem basis data tanpa replikasi dan failover dengan sistem basis data menggunakan replikasi dan failover adalah 0,0130 detik. Jadi perbedaan *response time* kedua kondisi tersebut kecil.

5. Simpulan

Berdasarkan hasil pengujian dapat diambil beberapa kesimpulan yaitu *:*

a. Heartbeat dapat dikombinasikan dengan *Native Replication* MySQl sehingga dihasilkan sistem failover dengan data identik antara 2 *server*.
b. Berdasarkan hasil pengujian, response time query dari sistem basis data yang menggunakan *failover* dan replikasi dengan *single server* basis data mempunyai perbedaan yang sedikit.

Daftar Pustaka :

[1] W. Luke, T. Laura, "MySQL Tutorial : A Concise Introduction to The Fundamentals of Working with MySQL", MySQL Press, 2004.

[2] Kadir, Abdul, "Membuat Aplikasi Web dengan PHP + Database MySQL". Andi, 2009.

[3] Oracle Documentation, *Failover and System Reliability*, [online], (*http://docs.oracle.com/cd/E12890_01/ales/docs32/HowTo/Failover.html*, diakses tanggal 1 Juni 2013

[4] MySQL Documentation, *How to Set Up Replication*

[5] ,[online], (*http://dev.mysql.com/doc/refman/5.1/en/replication-howto.html*, diakses tanggal 1 Juni 2013)

[6] Sahara. Soffa, D.I. Prajna, M. Barja Sanjaya", Implementasi Database Failover System Menggunakan Master To Master Replication (Studi Kasus Apotek Canggu Farma)", Politeknik Telkom 2012