Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 4, No. 1, Maret 2017, hlm. 69-74 e-ISSN: 2528-6579

ANALISIS AVAILABILITAS DAN RELIABILITAS *MULTI-MASTER DATABASE SERVER* DENGAN *STATE SNAPSHOT TRANSFERS* (SST) JENIS *RSYNC* PADA *MARIADB GALERA CLUSTER*

Mahendra Data[1], Gilang Ramadhan[2], Kasyful Amron[3]

[1,2,3] Fakultas Ilmu Komputer Universitas Brawijaya
Email: [1]mahendra.data@ub.ac.id, [2] gilang@ub.ac.id, [3]kasyful@ub.ac.id

(Naskah masuk: 27 Januari 2017, diterima untuk diterbitkan: 11 Maret 2017)

Abstrak

Sistem *database* merupakan bagian yang tak terpisahkan dari aplikasi berskala *enterprise*. Data didalamnya merupakan aset yang sangat penting, sehingga data tersebut tidak boleh rusak terlebih lagi hilang. Itulah sebabnya mengapa reliabilitas dan availabilitas sistem *database* menjadi hal yang sangat penting. Berbagai cara telah dikembangkan untuk meningkatkan reliabilitas dan availabilitas sebuah sistem *database*, salah satunya adalah teknik replikasi. MariaDB Galera Cluster adalah salah satu DBMS *open source* populer yang memiliki mekanisme replikasi. MariaDB Galera Cluster memiliki beberapa metode *State Snapshot Transfer* (SST) pada saat proses replikasi, yaitu rsync, mysqldump, xtrabackup, xtrabackup-v2. Kurangnya pemahaman administrator sistem terhadap perilaku tiap metode SST dapat mengakibatkan *error* pada sistem *database*. Untuk mencegahnya diperlukan analisis yang mendalam tentang dampak dan perilaku metode SST yang digunakan. Penelitian ini fokus pada analisis kinerja dari metode SST jenis rsync. Rsync dipilih karena metode ini merupakan metode SST *default* yang digunakan oleh MariaDB Galera Cluster. Dari hasil percobaan disimpulkan bahwa jumlah *node* dalam klaster menjadi hal yang perlu menjadi perhatian. Klaster yang hanya memiliki dua *node* akan sangat rentan terkena gangguan jika salah satu *node* terputus koneksinya atau mengalami *down* atau *crash*, sehingga jumlah *node* minimal dalam satu klaster yang disarankan adalah tiga *node* agar availabilitas dan reliabilitas MariaDB Galera Cluster dapat terjaga dengan baik.

Kata kunci: *database*, klaster, *availabilitas, reliabilitas, MariaDB Galera Cluster*

Abstract

The database system is an integral part of enterprise-scale applications. The data in it is a very important asset, so it may not be damaged or lost. That's why the reliability and availability of the database system become very important. Various ways have been developed to improve the reliability and availability of a database system, one of them is a replication technique. MariaDB Galera Cluster is one of the popular open-source DBMS that has a replication mechanism. MariaDB Galera Cluster has several methods of doing Snapshot State Transfer (SST) during the replication process, namely rsync, mysqldump, xtrabackup, xtrabackup-v2. Lack of understanding by the system administrator of the behavior of every SST method can lead to errors in the database system. To prevent it, depth analysis of the impact and the behavior of the SST methods is required. This study focused on analyzing the performance of rsync SST method. Rsync is chosen because it is the default SST method used by MariaDB Galera Cluster. The experimental results show that the number of nodes in a cluster should be concerned. Clusters which only have two nodes would be highly vulnerable to disruption if one node disconnected or experience down or crash. We recommend the minimum number of nodes in a cluster is three so that the availability and reliability of MariaDB Galera Cluster can be properly maintained.

***Keywords**: database, cluster, availability, reliability, MariaDB Galera Cluster*

1 PENDAHULUAN

Sistem *database* merupakan bagian yang tak terpisahkan dari aplikasi berskala *enterprise* (Domaschka et al. 2014). Hampir seluruh aplikasi pada skala *enterprise* memerlukan system *database* untuk menyimpan berbagai jenis data yang dimilikinya. Semua proses bisnis yang berjalan pada sebuah aplikasi *enterprise* akan membutuhkan kinerja sistem *database* yang optimal. Ketika *database server* mengalami kegagalan, maka semua layanan akan terhenti dan semua proses bisnis tidak akan dapat berjalan. Pentingnya peran *database* ini yang membuat sebuah *database* diharuskan memiliki *up time* yang tinggi. Data tersebut merupakan aset yang sangat penting bagi perusahaan, sehingga data tersebut tidak boleh rusak terlebih lagi hilang. Itulah sebabnya mengapa reliabilitas dan availabilitas sebuah sistem *database* menjadi hal yang sangat penting (Aditya & Juhana 2015).

Berbagai cara telah dikembangkan untuk meningkatkan reliabilitas dan availabilitas dari

sebuah sistem *database*, salah satunya adalah dengan teknik replikasi. Replikasi adalah teknik penggandaan data pada beberapa lokasi fisik yang berbeda untuk satu data logis yang sama (Domaschka dkk., 2014). Tujuannya agar ketika terjadi kerusakan atau kegagalan pada satu lokasi fisik maka tidak menyebabkan kegagalan keseluruhan sistem. Selain itu replikasi juga digunakan untuk meningkatkan availabilitas sistem *database* dengan cara membagi beban pekerjaan tiap *database server*. (Aditya & Juhana 2015). Pada prakteknya implementasi teknik replikasi pada sistem *database* akan berbeda, tergantung pada *Database Management System* (DBMS) yang digunakan.

Pengimplementasian replikasi pada MariaDB dapat dilakukan dengan menggunakan MariaDB Galera Cluster. MariaDB Galera Cluster adalah perangkat lunak DBMS MariaDB yang di dalamnya telah tertanam *patch* MySQL-wsrep dari Codership (Yurchenko 2009). MariaDB Galera Cluster mendukung teknik *multi-master database replication*. *Multi-master database replication* memungkinkan seluruh *node* dalam satu klaster menjadi *master database*, sehingga perubahan yang terjadi pada satu *node* akan direplikasi ke seluruh *node* yang tergabung dalam klaster tersebut. Teknik ini dapat meningkatkan reliabilitas dan availabilitas dari sistem *database*, karena bila terjadi kerusakan atau kesalahan pada salah satu *database server* maka trafik akses *database* dapat dialihkan ke *database server* yang lain dalam klaster tersebut. Alasan inilah yang menyebabkan teknik *multi-master database replication* ini banyak digunakan pada sistem *database* saat ini (Aditya & Juhana 2015).

Pada *multi-master database replication* dalam MariaDB Galera Cluster, ketika satu *node* bergabung ke dalam satu klaster (*joiner node*), *node* tersebut akan menerima pembaruan seluruh data dari salah satu *node* dalam klaster tersebut (*donor node*). Proses ini disebut dengan *State Snapshot Transfer* (SST). MariaDB Galera Cluster memiliki beberapa metode dalam melakukan SST, yaitu rsync, mysqldump, xtrabackup, xtrabackup-v2 (MariaDB 2017).

Tiap metode SST memiliki mekanisme yang bebeda dalam melakukan *state transfer*, sehingga berdampak pada perilaku *database server* dalam menangani kesalahan. Kurangnya pemahaman administrator sistem terhadap perilaku masing-masing metode SST dapat mengakibatkan *error* pada sistem *database*. Untuk mencegahnya diperlukan analisis yang mendalam tentang dampak dan perilaku metode SST yang digunakan. Penelitian ini fokus pada analisis kinerja dari metode SST jenis rsync. Rsync dipilih karena metode ini merupakan metode SST *default* yang digunakan oleh MariaDB Galera Cluster (MariaDB 2017). Hasil dari penelitian ini diharapkan dapat meningkatkan pemahaman dari administrator sistem tentang *multi-master database replication* yang menggunakan rsync sebagai metode SST.

Hasil penelitian ini disusun dengan struktur sebagai berikut. Bagian pertama adalah dasar teori mengenai reliabilitas, availabilitas, klaster *database*, jenis replikasi dan DBMS MariaDB secara umum. Bagian ini digunakan untuk menyamakan presepsi mengenai teori dan istilah-istilah yang digunakan. Bagian kedua adalah perancangan metode serta *testbed* yang digunakan dalam penelitian ini. Bagian ketiga adalah penjabaran hasil percobaan serta analisis dari hasil percobaan yang telah dilakukan. Bagian terakhir dari penelitian ini adalah penyampaikan kesimpulan dari keseluruhan percobaan yang telah dilakukan.

2 DASAR TEORI

2.1 Availabilitas dan Reliabilitas

Availabilitas adalah tingkat ketersediaan suatu sistem untuk diakses dan dipergunakan ketika diperlukan. Sedangkan reliabilitas adalah ukuran kemampuan suatu sistem dalam memberikan hasil yang benar ketika dipergunakan pada berbagai keadaan (Domaschka dkk., 2014). Dengan kata lain, availabilitas adalah jaminan ketersediaan layanan suatu sistem, sedangkan reliabilitas adalah jaminan kebenaran hasil pemrosesan dari suatu sistem. Secara teori, availabilitas dan reliabilitas adalah dua hal yang terbeda dan tidak terikat satu dengan yang lain. Namun pada prakteknya, availabilitas namun tanpa reliabilitas atau sebaliknya mengakibatkan sistem tersebut tidak banyak berguna (Domaschka et al. 2014). Sebagai contoh, misalnya terdapat satu sistem *database* dengan availabilitas tinggi namun reliabilitas rendah karena data didalamnya tidak konsisten atau sebaliknya, terdapat satu sistem *database* dengan reliabilitas tinggi namun availabilitas rendah karena koneksi *database*-nya sering terputus maka sistem *database* yang seperti itu menjadi tidak layak digunakan.

2.2 Klaster *Database*

Klaster *database* adalah kumpulan dari beberapa *database server* yang secara logika dapat dipandang sebagai satu kesatuan sistem *database*. Satu *server database* dalam satu klaster, atau biasa disebut sebagai *node*, memiliki perangkat keras (CPU, *memory*, *disk*, dll.) dan perangkat lunak (sistem operasi, *service*, dll.) tersediri yang bekerja secara independen. Namun secara logika, antar *node* dalam klaster *database* tersebut saling terhubung melalui sebuah perangkat lunak yang mengelola seluruh *node* di dalam klaster tersebut (Pacitti et al. 2005).

Klaster *database* merupakan salah satu solusi dalam meningkatkan availabilitas dan reliabilitas dari sebuah sistem *database*, karena dapat mencegah terjadinya *single point of failure* dari sistem

database. Sebagian besar DBMS saat ini telah mendukung klaster *database*. Salah satu yang paling populer diantaranya adalah MariaDB Galera Cluster (MariaDB 2014b).

2.3 Replikasi *Synchronous* dan *Asynchronous*

Perbedaan utama dari replikasi *synchronous* dan *asynchronous* adalah replikasi *synchronous* menjamin perubahan data pada satu *node* akan memicu perubahan data pada *node* lainnya secara *synchronous* atau dengan kata lain terjadi secara *real time*, sedangkan pada replikasi *asynchronous* tidak menjadmin sinkronisasi data terjadi secara *real time* karena besarnya *delay* sinkronisasi data pada replikasi *asynchronous* bervariatif. Dampak lainnya adalah bila *master node* (*node* utama) mengalami *crash* maka data yang belum tersinkonisasi ke *slave node* (*node* penerima) akan hilang (MariaDB 2014b).

Secara teori, replikasi *synchronous* memiliki beberapa kelebihan dibandingkan dengan replikasi *asynchronous*, kelebihan tersebut antara lain:

- Tingkat availabilitas yang lebih tinggi;
- Transaksi pada *database* terjadi secara pararel di seluruh *node*;
- Menjamin kualitas seluruh *node* dalam klaster, misalnya perintah SELECT yang dijalankan sesaat setelah perintah INSERT pada *node* A akan menghasilkan nilai yang salah walaupun di jalankan di *node-node* yang lain.

Namun secara praktek, replikasi *synchronous* mengimplementasikan mekanisme "2-phase commit" atau distributed locking yang terbukti memiliki performa yang sangat rendah. Itulah sebabnya mengapa penggunaan replikasi *asynchronous* masih dominan. DBMS *open source* populer seperti MySQL dan PostgreSQL secara *default* hanya menyediakan fasilitas replikasi *asynchronous* (MariaDB 2014a).

2.4 MariaDB

MariaDB adalah DBMS yang bersifat *open soruce* dan dikembangkan oleh pengembang yang sama dari MySQL. MySQL sendiri merupakan DBMS yang sudah sangat populer digunakan dan saat ini telah diakuisisi oleh perusahaan Oracle. Perkembangan MariaDB terbilang sangat cepat bila dibandingkan dengan DBMS lain yang sama-sama bersifat *open source*. Saat ini MariaDB telah digunakan lebih dari 12 juta pengguna di dunia, termasuk perusahaan-perusahaan besar seperti booking.com, HP, Virgin Mobile and Wikipedia (MariaDB 2014b).

MariaDB Galera Cluster adalah perangkat lunak DBMS MariaDB yang di dalamnya telah tertanam *patch* MySQL-wsrep dari Codership (Yurchenko 2009). *Patch* tersebut digunakan untuk membuat sistem replikasi eksternal untuk MySQL (dan turunan yang kompatibel dengannya, termasuk MariaDB) yang mengikuti standar API wsrep. Sistem replikasi ini nantinya dapat bersifat *asynchronous* atau *synchronous* dan dapat digunakan untuk *single* atau *multi-master database replication* (Yurchenko 2009). Pada awal perkembangannya MariaDB Galera Server merupakan *package* terpisah DBMS MariaDB, Namun pada versi 10.1, MariaDB Galera Server sudah terintegrasi pada DBMS MariaDB, sehingga pengguna tidak perlu lagi melakukan instalasi *package* MariaDB Galera Server secara manual (MariaDB 2014b)

Terdapat beberapa limitasi dalam implementasi replikasi menggunakan MariaDB Galera Server seperti yang telah dijelaskan pada laman resmi MariaDB. Limitasi tersebut dibagi ke dalam dua bagian, yaitu keterbatasan dari *patch* MySQL-wsrep dari Codership dan batasan yang didapatkan dari pengalaman pengguna (MariaDB 2015). Berikut ini adalah limitasi dari *patch* MySQL-wsrep:

- Hanya dapat bekerja pada InnoDB *storage engine*;
- Tidak mendukung *explicit locking* seperti LOCK TABLES, FLUSH TABLES {daftar tabel} WITH READ LOCK, (GET_LOCK(), RELEASE_LOCK(),…);
- Operasi *global locking* seperti FLUSH TABLES WITH READ LOCK masih dapat digunakan;
- Seluruh tabel harus memiliki *primary key*, atau dapat juga menggunakan *multi-column primary keys*. Operasi DELETE tidak dapat dilakukan pada tabel yang tidak memiliki *primary key*;
- *Query Log* tidak dapat diarahkan ke dalam tabel, melainkan harus diarahkan ke sebuah *file*;
- Tidak mendukung XA *transactions*;
- Ukuran data dalam satu transaksi tidak boleh melebihi limitasi parameter *wsrep_max_ws_rows* dan *wsrep_max_ws_size* yang telah ditentukan dalam konfigurasi MariaDB Galera Cluster. *Default*-nya *wsrep_max_ws_rows* bernilai 128K dan *wsrep_max_ws_size* bernilai 1Gb.
- Nilai *auto-increment* tidak akan berurutan karena galera memiliki mekanisme *auto-increment* tersendiri untuk mencegah konflik antar *server*;
- Bila koneksi antar *node* dalam klaster terputus, misalnya disebabkan karena gangguan koneksi, dan sebuah *node* dalam kondisi belum tersinkronisasi, maka perintah yang dijalankan pada *node* tersebut akan *error* dan menampilkan pesan error '*WSREP has not yet prepared node for application use*'. Hal ini dilakukan oleh MariaDB Galera Cluster untuk mencegah terjadinya inkonsistensi data;
- Bila metode SST yang digunakan adalah rsync dan terdapat salah satu *node* yang mengalami *crash* sebelum SST berakhir maka proses rsync akan *hang* dan *port* yang digunakan akan tertahan selamanya. Pesan *error* '*port in use*' akan ditampilkan pada *error log* pada server;

- Pada klaster dengan jumlah transaksi yang tinggi, performa sistem *database* tidak dapat lebih tinggi dari performa *node* yang paling rendah dalam klaster tersebut. Hal ini disebabkan karena penggunaan *wsrep provider* pada MariaDB Galera Cluster akan meningkatkan jumlah transaksi dalam jumlah yang cukup besar;
- Perintah FLUSH PRIVILEGES tidak akan direplikasi sehingga pengguna harus menjalankan perintah tersebut secara manual untuk tiap *node* dalam klaster;

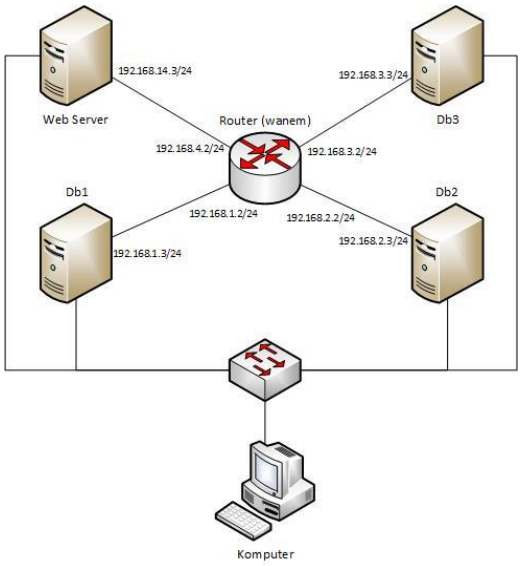
3 PERANCANGAN

Proses perancangan pada penelitian ini terbagi menjadi dua bagian. Bagian pertama adalah proses perancangan *testbed* pengujian. Bagian kedua adalah perancangan skenario pengujian.

3.1 Rancangan *Testbed*

Testbed percobaan dibangun menggunakan satu komputer fisik dan teknologi virtualisasi VirtualBox. Dalam komputer fisik tersebut akan dirancang beberapa komputer virtual dengan topologi sesuai dengan Gambar 1. Ada beberapa perangkat lunak yang digunakan dalam penelitian ini. Perangkat lunak yang tersebut adalah:

- MariaDB versi 10.1.18. Pada versi ini, MariaDB Galera cluster sudah terintegrasi di dalamnya.
- Wanem versi 3.0 yang akan digunakan untuk emulasi kondisi jaringan sesuai dengan skenario pengujian.
- VirtualBox versi 5.1.6 yang akan digunakan untuk membangun testbed penelitian.



Gambar 1 Testbed Pengujian

Terdapat lima komputer virtual dalam *testbed* tersebut. Tiga komputer virtual berperan sebagai klaster *database*. Di dalamnya terdapat perangkat lunak MariaDB Galera Cluster yang telah dikonfigurasi sesusai dengan skenario pengujian. Satu komputer berperan sebagai *Web Server* yang bertugas membangkitkan data yang nantinya akan dimasukkan ke dalam *node* tertentu dalam klaster *database*. Satu komputer virtual terakhir berperan sebagai *router* yang menghubungkan ketiga *database server* sekaligus melakukan emulasi trafik jaringan sesuai dengan skenario pengujian menggunakan perangkat lunak Wanem.

Keseluruhan komputer virtual tersebut terhubung ke komputer fisik (*host*) melalui sebuah jaringan khusus yang disebut dengan *host only network*. Tujuannya agar proses pemantauan tiap komputer virtual tidak terganggu oleh dampak emulasi trafik jaringan yang dilakukan oleh *router*.

3.2 Rancangan Skenario Pengujian

Bagian ini mejelaskan rancangan skenario pengujian. Dari skenario ini diharapkan dapat menghasilkan nilai pengujian yang mampu menggambarkan availabilitas dan reliabilitas dari replikasi *master-to-master database server* menggunakan metode SST *rsync* pada MariaDB Galera Cluster. Daftar skenario pengujian dapat dilihat pada Tabel 1.

Tabel 1. Skenario Pengujian

Skenario	Jenis Kegagalan	Jumlah *Node*
1	*database service down*	2
2	*database service down*	3
3	*connection down*	2
4	*connection down*	3
5	*server down*	2
6	*server down*	3
7	10% *packet loss*	2
8	10% *packet loss*	3

Secara umum terdapat empat jenis kegagalan yang diuji coba, yaitu:

- *database service down*, yaitu kondisi dimana *database service* pada salah satu *node* sengaja dihentikan secara normal. Skenario ini mensimulasikan kondisi ketika salah satu *node* mengalami *maintenance* sehingga *database service* perlu dihentikan;
- *connection down*, yaitu kondisi dimana koneksi jaringan pada salah satu *node* sengaja diputus. Skenario ini mensimulasikan kondisi ketika koneksi jaringan pada salah satu *database server* terputus;
- *server down*, yaitu kondisi dimana salah satu *node* sengaja dimatikan paksa sehingga menyebabkan *database service* akan terhenti secara abnormal. Skenario ini mensimulasikan kondisi ketika perangkat keras *server* mengalami gangguan yang menyebabkan komputer *server* berhenti bekerja;
- 10% *packet loss*, yaitu kondisi dimana *packet loss* pada koneksi jaringan salah satu *node* sengaja dibangkitkan sebesar 10% menggunakan *network* emulator. Skenario ini mensimulasikan

kondisi ketika tejadi gangguan pada jaringan berupa *packet loss*.

Keempat jenis kegagalan tersebut diuji pada klaster dengan jumlah *node* yang berbeda, yaitu dua *node* dan tiga *node* dalam satu klaster. Tujuan dari skenario dengan perbedaan jumlah *node* ini adalah untuk mengetahui dampak dari penambahan jumlah *node* terhadap availabilitas dan reliabilitas dari klaster *database*.

Ketika Web Server membangkitkan data yang kemudian dimasukkan ke dalam salah satu *node* dalam klaster *database*, delapan skenario kegagalan tersebut akan dijalankan pada satu *node* lainnya. Kemudian dipantau dampak dari kegagalan tiap skenario tersebut. Langkah berikutnya adalah memulihkan kondisi kegagalan dan memantau kondisi klaster *database* setelah kondisi kegagalan tersebut behasil dipulihkan. Langkah ini mensimulasikan kondisi ketika kegagalan tersebut berhasil diperbaiki. Dengan skenario pengujian seperti ini diharapkan dapat diketahui availabilitas dan reliabilitas dari klaster *database* ketika dan setelah kondisi kegagalan terhadap terjadi.

4 HASIL DAN ANALISIS PERCOBAAN

Hasil dari percobaan sesuai dengan skenario pada Tabel 1 dapat dilihat pada Tabel 2. Data tersebut didapatkan dari beberapa kali percobaan pada tiap skenario. Hasil yang didapat tiap kali percobaan tidak berubah, sehingga dapat disimpulkan bahwa dampak dari tiap skenario kegagalan bersifat konsisten.

Skenario nomor 1 dan 2, yaitu skenario kegagalan berupa *database service down* dengan jumlah *node* 2 dan 3, tidak menyebabkan gangguan pada klaster *database*. *Node* yang aktif masih dapat melayani akses dari pengguna.

Pada skenario nomor 3, yaitu skenario kegagalan berupa *network down* pada salah satu *node* dalam klaster dengan jumlah *node* 2, *node* lain yang masih aktif tidak dapat beroperasi. Ketika mencoba memasukkan data menggunakan *script* yang dijalankan dari *Web Server* ke dalam *node* tersebut muncul pesan *error* '*Deadlock found when trying to get lock*'. Hal ini disebabkan karena koneksi jaringan pada skenario tersebut terputus ketika *State Snapshot Transfer* (SST) sedang berlangsung pada kedua *node*, sehingga MariaDB mencegah adanya penambahan data baru untuk menghindari inkonsistensi data. Namun kondisi ini tidak terjadi pada klaster dengan tiga *node* (skenario nomor 4). Pada klaster dengan tiga *node*, proses SST masih dapat berlangsung pada dua *node* yang masih aktif, sehingga gangguan pada satu *node* tidak menyebabkan gangguan sinkronisasi data pada *node* lainnya dalam klaster tersebut. Hal ini tidak terjadi pada skenario nomor 4 yang menggunakan 3 buah *node*.

Pada skenario nomor 5, yaitu skenario kegagalan berupa *server down* pada salah satu *node* dalam klaster dengan jumlah *node* 2, node lainnya tidak mengalami gangguan dan dapat beroperasi normal. Namun ketika *node* yang dihentikan paksa tersebut dinyalakan kembali muncul pesan *error* 1407 yang bertuliskan '*WSREP has not yet prepared node for application use*' pada kedua *node*. *Error* ini berlangsung terus menerus sehingga seluruh node dalam klaster berhenti bekerja dan tidak dapat melakukan sinkronisasi data. Hal ini disebabkan karena adanya data yang *corrupt* pada *node* yang dimatikan secara paksa.

Tabel 2. Hasil Percobaan

Skenario	Dampak	Kondisi Setelah Kegagalan Dipulihkan
1	NORMAL; *Node* yang lain tetap dapat beroperasi.	NORMAL; Replikasi kembali berjalan; Data sinkron.
2	NORMAL; *Node* yang lain tetap dapat beroperasi.	NORMAL; Replikasi kembali berjalan; Data sinkron.
3	*ERROR*; *Node* yang lain tidak dapat beroperasi.	NORMAL; Replikasi kembali berjalan.
4	NORMAL; *Node* yang lain tetap dapat beroperasi.	NORMAL; Replikasi kembali berjalan.
5	NORMAL; *Node* yang lain tetap dapat beroperasi.	*ERROR*; Replikasi terhenti dan muncul error 1407.
6	NORMAL; *Node* yang lain tetap dapat beroperasi.	NORMAL; Replikasi sempat terhenti dan muncul error 1407 namun tidak berapa lama replikasi berjalan normal kembali.
7	NORMAL; *Node* yang lain tetap dapat beroperasi.	NORMAL; Replikasi kembali berjalan.
8	NORMAL; *Node* yang lain tetap dapat beroperasi.	NORMAL; Replikasi kembali berjalan.

Namun sekali lagi, kondisi ini tidak terjadi pada klaster dengan tiga *node* (skenario nomor 6). Pada klaster dengan jumlah *node* 3, pesan *error* 1407 tersebut hanya muncul sesaat ketika *node* yang dihentikan paksa tersebut dinyalakan kembali, namun setelah beberapa lama pesan *error* tersebut hilang dan sinkronisasi data dalam klaster tersebut dapat berjalan normal kembali.

Pada skenario nomor 7 dan 8, yaitu skenario kegagalan berupa *packet loss* sebesar 10%, klaster *database* dapat berjalan dengan baik. Gangguan berupa *packet loss* sebesar 10% tersebut masih dapat

diatasi oleh klaster *database* baik klaster yang menggunakan dua *node* maupun klaster yang menggunakan tiga *node*.

Terdapat beberapa catatan yang juga menjadi perhatian dari hasil percobaan. Pertama adalah kondisi *service* MariaDB setelah *server* mengalami *crash* atau dihentikan paksa. Ketika *server* dinyalakan kembali, *service* MariaDB tidak dapat dinyalakan. Langkah yang harus dilakukan adalah dengan melakukan *recovery* database dengan perintah 'mysqld --wsrep-recover'. Perintah ini akan melakukan perbaikan data yang gagal tersinkonisasi.

5 KESIMPULAN

Tabel 3. Kesimpulan hasil percobaan

Jenis Kegagalan	Jumlah Node	
	2	3
database service down	Normal	Normal
connection down	*Error*	Normal
server down	*Error*	Normal
10% *packet loss*	Normal	Normal

Percobaan yang dilakukan terhadap MariaDB Galera Cluster pada penelitian ini semakin menguatkan pernyataan limitasi yang telah tertulis pada dokumentasi MariaDB Galera Cluster. Selain itu terdapat beberapa hal yang perlu menjadi perhatian oleh administrator sistem ketika mengimplementasikan MariaDB Galera Cluster.

Sepeti yang telah dituliskan pada Tabel 3, hal tertama pertama yang harus menjadi perhatian adalah jumlah *node* dalam klaster. Hasil percobaan membuktikan bahwa klaster yang hanya memiliki dua *node* akan sangat rentan terkena gangguan jika salah satu *node* terputus koneksinya atau mengalami *down* atau *crash*. Bahkan gangguan tersebut dapat menyebabkan keseluruhan klaster berhenti bekerja. Hal ini sangat fatal karena *database* adalah bagian krusial dari sebuah sistem atau aplikasi, dimana kegagalan database akan menyebabkan keseluruhan sistem atau aplikasi dapat terganggu. Untuk mencegah hal tersebut, jumlah node minimal yang disarankan ketika membangun klaster *database* adalah tiga *node*. Dari hasil percobaan terbukti bahwa klaster dengan 3 buah *node* dapat menanggai seluruh skenario kegagalan yang diujikan dan klaster *database* tetap dapat bekerja dengan baik. Hal kedua adalah ketika *server* mengalami *crash*, maka *service* MariaDB tidak dapat berjalan ketika server meyala kembali. Perlu dilakukan *recovery* terlebih dahulu secara manual dengan perinah 'mysqld --wsrep-recover'.

Secara umum dapat disimpulkan bahwa dengan menggunakan tiga *node* dalam satu klaster *database*, availabilitas dan reliabilitas MariaDB Galera Cluster dapat terjaga dengan baik, dengan catatan bahwa tidak ada lebih dari dua *node* yang mengalami kegagalan pada saat yang bersamaan. Bila memang data yang disimpakan sangat krusial, maka disarankan untuk menambah jumlah node agar dapat meningkatkan availabilitas dan reliabilitas dari sistem *database*.

6 DAFTAR PUSTAKA

ADITYA, B. & JUHANA, T., 2015. A high availability (HA) MariaDB Galera Cluster across data center with optimized WRR scheduling algorithm of LVS - TUN. In *2015 9th International Conference on Telecommunication Systems Services and Applications (TSSA)*. IEEE, hal. 1–5.

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.

MARIADB, 2014a. About Galera Replication. Tersedia di: https://mariadb.com/kb/en/mariadb/about-galera-replication/ [Diakses Januari 22, 2017].

MARIADB, 2014b. About MariaDB. Tersedia di: https://mariadb.org/about/ [Diakses Januari 12, 2017].

MARIADB, 2015. MariaDB Galera Cluster - Known Limitations. Tersedia di: https://mariadb.com/kb/en/mariadb/mariadb-galera-cluster-known-limitations/ [Diakses Januari 19, 2017].

MARIADB, 2017. MariaDB Galera Cluster Configuration Variables. Tersedia di: https://mariadb.com/kb/en/mariadb/galera-cluster-system-variables/.

PACITTI, E. ET AL., 2005. Preventive Replication in a Database Cluster. *Distributed and Parallel Databases*, 18(3), hal.223–251.

YURCHENKO, A., 2009. MySQL patches by Codership. Tersedia di: https://launchpad.net/codership-mysql [Diakses Januari 19, 2017].