IJCCS, Vol.10, No.2, July 2016, pp. 149~160
ISSN: 1978-1520

Implementasi *Highly Available Website* Dengan *Distributed Replicated Block Device*

Mulyanto*[1], Ahmad Ashari[2]
[1]Jurusan Informatika, Fakultas Ilmu Komputer, UMRI, Riau
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
e-mail: ***[1]myanto@gmail.com**, [2]ashari@ugm.ac.id

Abstrak

Sebagai sebuah infrastruktur IT yang penting, website merupakan sistem yang membutuhkan kehandalan dan tingkat availability yang tinggi. Website memenuhi kriteria sebagai sistem yang harus highly available karena website dituntut untuk memberikan layanan terhadap client secara real time, menangani data yang banyak dan tidak boleh ada kehilangan data ketika terjadi transaksi. Sistem yang highly available harus memenuhi syarat yaitu layanan tidak hanya mampu berjalan secara terus menerus, tetapi juga menjamin konsistensi terhadap permintaan data.

Pada penelitian ini akan dilakukan rancang bangun website yang memiliki sifat highly availability. Pendekatan yang dilakukan dengan membangun network cluster dengan fungsi failover dan replicated block device. Failover dibangun untuk memberikan ketersediaan layanan sedangkan replicated block device dimanfaatkan untuk memberikan kepastian terhadap konsistensi data pada saat terjadinya kegagalan terhadap layanan. Dengan pendekatan failover cluster dan replicated block device didapatkan cluster yang mampu menangani terjadinya kegagalan layanan web server maupun database server pada website.

Hasil dari penelitian ini, layanan website yang dibangun dapat berjalan dengan baik apabila terjadi kegagalan pada anggota node dari cluster. Sistem mampu memberikan tingkat availability pada layanan database server sebesar 99,999 (five nines) dan 99,98 (three nines) pada layanan web server.

Kata kunci*—High Availability Website, Failover, Distributed Replicated Block Device*

Abstract

As an important IT infrastructure, website is a system which requires high reliability and availability levels. Website meets the criteria as a highly available system because website must provide services to clients in real time, handle a large amount of data, and not lose data during transaction. A highly available system must meet the condition of being able to run continuously as well as guaranteeing consistency on data requests.

This study designed a website with high availability. The approach was building network cluster with failover and replicated block device functions. Failover was built to provide service availability, while replicated block device provides data consistency during failure of service. With failover cluster and replicated block device approaches, a cluster which is able to handle service failures of web server and database server on the website.

The result of this study was the services of the website could run well if there was any failure in node members of the cluster. The system was able to provide 99,999 (five nines) availability on database server services and 99,98 (three nines) on web server services.

Keywords—*High Availability Website, Failover, Distributed Replicated Block Device*

Received October 26th,2016; Revised January 13th, 2017; Accepted January 13th, 2017

1. PENDAHULUAN

1.1 Latar Belakang Masalah

Sebuah sistem dan infrastruktur IT yang harus selalu *online* seperti *website*, email, dan media *e-learning* merupakan sistem yang membutuhkan kehandalan. Ketika ukuran data tumbuh menjadi besar karena semakin banyak orang dan sistem yang berinteraksi, mengakibatkan pengelolaan data menjadi tidak efektif dan memiliki imbas terhadap kemampuan sistem untuk menyediakan layanan yang handal [1].

Syarat agar layanan *highly available* yaitu memiliki kecepatan kinerja, *scalable*, dan data yang terdistribusi [2]. Dalam menerapkan sistem yang *high availability*, banyak metode yang dapat dilakukan antara lain dengan melakukan pemisahan penyimpanan data. Tempat penyimpanan data dapat dibagi menjadi beberapa *node*. Hal ini berkaitan dengan mekanisme distribusi data yang digunakan oleh sistem dalam membagi-bagi data yang disimpannya ke dalam beberapa *node*.

Pada penelitian ini akan dilakukan rancang bangun *website* yang memiliki sifat *highly available. Website* memenuhi kriteria sebagai sistem yang *high availability* karena *website* harus memberikan layanan terhadap beberapa *client* secara *realtime*, menangani data yang banyak dan tidak boleh ada kehilangan data ketika terjadi transaksi [3]. *High availability* yang akan dibangun pada penelitian ini meliputi *server-server* yang berkorelasi dengan layanan *website* yaitu *web server* dan *database* server.

Website yang *high availability* dibangun dengan memanfaatkan *protocol* Heartbeat dan *failover clustering*. Heartbeat yang digunakan adalah Linux yaitu *Corosync Cluster Engine* dan replikasi data dengan *Distributed Replicated Block Device* (DRBD). Heartbeat dimanfaatkan untuk melakukan manajemen *node* sehingga mampu saling berkomunikasi dan melakukan sinkronisasi *service* yang berjalan. Sedangkan *Distributed Replicated Block Device* bekerja secara redundan dimana terdapat *block device* pada *secondary server* yang akan melakukan replikasi terhadap *block data* yang ada pada *primary server*. Penelitian perlu dilakukan untuk mengukur kepastian terhadap resiko kehilangan data dengan menggunakan konsep layanan *recovery* secara *mirroring* dan *synchronous replication* [4].

1.2 Rumusan Masalah

Dari latar belakang dan permasalahan yang telah disampaikan, permasalahan yang menjadi fokus penelitian ini adalah:

- Bagaimana merancang layanan *website* yang memiliki tingkat *availability* tinggi.
- Bagaimana mengimplementasikan *website* yang memiliki tingkat *availability* yang tinggi menggunakan *failover clustering* dengan pendekatan *Distributed Replicated Block Device.*
- Bagaimana perbandingan pada sistem *high availability* melalui pengujian *availability* sistem, waktu proses terhadap permintaan data (*response time*) dan kemampuan sistem terhadap beban kerja.

1.3 Batasan Masalah

Agar permasalahan tidak menjadi bias dan meluas, maka masalah yang diteliti akan dibatasi sebagai berikut:

- Sistem akan dibangun atas 2 (dua) *node* yang saling terhubung pada sebuah *network cluster*.
- *Web server* yang dipilih adalah Apache *web server* dengan *database relational* MySQL yang berjalan pada sistem operasi Linux 64-bit.
- Pengujian fungsional dilakukan dengan mengukur tingkat ketersediaan sistem *(availability)* dan pengujian beban dilakukan dengan bantuan *tool* Siege dimana jumlah simulasi pengguna *konkuren* dan perulangan pengujian ditentukan oleh penguji.
- Simulasi pengujian dilakukan hanya untuk menguji *availability website* yaitu *web server* dan *database server* pada *local area network*.

1.4 Tujuan

Penelitian ini bertujuan untuk menyediakan sistem dengan kemampuan *high availability* pada *website* sehingga diharapkan dapat meningkatkan *availability* dalam melayani pertumbuhan dan permintaan data. Penelitian ini juga menguji tingkat *availability* dari sistem dengan menggunakan *multi-node Distributed Replicated Block Device*.

1.5 Manfaat

Adapun manfaat yang ingin diperoleh dari penelitian ini adalah memberikan kontribusi terhadap bagaimana implementasi sistem *high availability* dengan menggunakan *failover clustering* dan *Distributed Replicated Block Device dan m*endapatkan hasil pengujian yang tepat terhadap tingkat *availability.*

1.6 Penelitian Terdahulu

Penelitian Setyorini [5] tentang Rancang Bangun Sistem Informasi Akademik yang *Fault Tolerance*. Penelitian ini membahas perancangan dan pembuatan Sistem Informasi Akademik yang *fault tolerance.* Proses replikasi menggunakan *passive replica* dimana terdapat beberapa *node* dari *server database* yang menjadi *slave* dan melakukan replika terhadap *server master*.

Sedangkan Oei [6] melakukan penelitian terhadap toleransi kegagalan *server database* dilakukan dengan menggunakan *server database slave* yang merupakan replikasi dari *server database* pusat (*master*). Penelitian tentang *database* terpusat juga dilakukan oleh Angraini. Anggraini [7] melakukan penelitian tentang *server database* yang memiliki tingkat ketersediaaan tinggi (*high availability*) dan tingkat keandalan tinggi (*high reliability*) dengan menggunakan konfigurasi *master slave database cluster* dengan mekanisme *failover.*

Afif dan Suryono [8] melakukan penelitian untuk membangun *Disaster Recovery Plan* dengan DRBD. Implementasi dilakukan dengan *server* utama dan *server backup* yang ditempatkan pada *data center* yang berbeda.

Atmaja [1] melakukan penelitian tentang fragmentasi elastis *multi-node* pada *database* non-relasional untuk *website* forum diskusi. Hasil dari penelitian ini adalah sistem *database* yang memiliki *availability* dan *scalability* dengan menggunakan model *database* non-relasional.

Pada penelitian ini dilakukan implementasi *multi-node Distributed Replicated Block Device* dan *failover clustring* yang dimanfaatkan pada *website* untuk meneliti apakah *availability* yang tinggi dapat dicapai dengan menggunakan mekanisme *clustering Distributed Replicated Block Device,* dimana belum digunakan pada penelitian-penelitian sebelumnya.

1.7 Landasan Teori

Landasan teori merupakan teori-teori yang digunakan untuk memecahkan permasalahan. Pada bagian ini akan diuraikan teori-teori yang menjadi penunjang penelitian yang akan digunakan sebagai dasar dari pemecahan masalah.

Availability

Availability merupakan ukuran seberapa lama suatu *service* atau komponen-komponen dari *service* dapat segera digunakan. *Availability* merupakan properti yang digunakan sistem untuk tetap beroperasi bahkan ketika terjadi kegagalan proses [9]. *Availability* dapat juga didefinisikan sebagai kemungkinan bahwa sistem akan beroperasi dengan benar dalam waktu tertentu dan dapat menjalankan fungsinya bagi kepentingan *client* [10].

Availability selalu terkait dengan *Service Level Agreement* (SLA). SLA terdiri dari beberapa bagian yang mendefinisikan tanggung jawab terhadap tingkat ketersediaan layanan. Tingkat ketersediaan sistem diukur dari jumlah angka Sembilan (9). Semakin sedikit angka sembilan pada tingkat ketersediaan sistem maka semakin rendah pula *availability* sistem tersebut. Untuk aplikasi pada umumnya memiliki presentasi *uptime* 99% sudah dapat dianggap layak [11]. Perbandingan *availability* layanan terhadap presentase *uptime* ditunjukkan oleh Tabel 1.

Tabel 1 Tingkat *availability* layanan [11]

Presentase *Uptime* (%)	Presentase Downtime (%)	*Downtime* pertahun	*Downtime* perminggu
98	2	7.3 hari	3 jam 22 menit
99	1	3.65 hari	1 jam 41 menit
99,8	0,2	17 jam 30 menit	20 menit 10 detik
99,9	0,1	8 jam 45 menit	10 menit 5 detik
99,99	0,01	52.5 menit	1 menit
99,999	0,001	5.25 menit	6 detik
99,9999	0,0001	31.5 detik	0.6 detik

Sistem *High Availability*

Sistem *high availability* biasanya dibangun untuk sistem yang memiliki tingkat toleransi kesalahan yang tinggi atau *safety-critical.* Layanan yang disediakan bersifat sangat penting, sehingga apabila terjadi kerusakan tidak akan menghentikan proses dalam sistem. Untuk membangun sistem yang *high availability* tidah hanya dibangun pada komponen yang tunggal akan tetapi melibatkan proses replikasi antara beberapa komponen. Jika ingin dibangun sistem yang *high availability*, maka tidak dapat tergantung hanya pada mesin tunggal, akan tetapi diperlukaan beberapa mesin independen dan dapat dilakukan perawatan secara fisik, pada salah satu anggota dari *critical system* [12].

Network Cluster

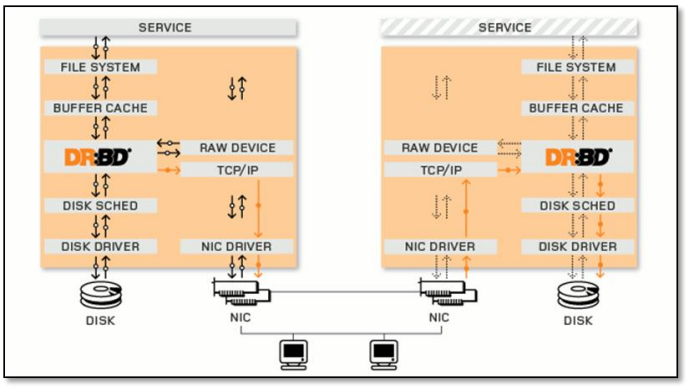
Cluster dapat didefinisikan sebagai sebuah sistem dari dua atau lebih sistem komputer independen dan subsistem penyimpanan yang saling berkomunikasi untuk tujuan membagi dan mengakses sumber daya [13]. Konsep sistem *cluster* didesain untuk meningkatkan tingkat kemampuan layanan. *Cluster* merupakan sekumpulan komputer yang saling berelasi dan bekerja bersama untuk menyelesaikan berbagai macam pekerjaan. Prinsip dari *cluster* adalah bahwa kumpulan komputer tersebut bekerja secara redundan sebagai sistem tunggal yang memiliki kemampuan yang lebih baik dibandingkan dengan komputer tunggal. Terdapat dua pendekatan yaitu *failover cluster* dan *load balancing cluster*.

Failover Cluster

Konsep konfigurasi *failover cluster* adalah membuat satu *server* sebagai *primary server* dan *server* yang lain menjadi *secondary server* dimana saat *server* dalam keadaan normal *primary server* menangani semua *request* dari *client*. *Secondary server* akan mengambil alih tugas *master server* apabila *primary server* gagal berfungsi. Implementasi *cluster* jenis ini akan mencoba untuk menggunakan redudansi komponen *cluster* untuk menghilangkan kegagalan disatu tititk (*single point of failure*). *Failover cluster* mengijinkan sebuah layanan untuk berpindah dari satu *host* ke *host* yang lain apabila terjadi *error*.

Distributed Replicated Block Device

*Distributed Replicated Block Device (*DRDB) merupakan mekanisme yang mirip dengan RAID dimana melakukan *mirroring* atau replikasi terhadap data. Jika RAID melakukan *mirroring* terhadap *device to device* terhadap media penyimpanan maka DRBD melakukan *mirroring* melalui jaringan. DRBD bekerja secara redundan dimana terdapat beberapa *node* melakukan replikasi dengan *failover*. Sinkronisasi antar *node* dilakukan dengan menggunakan mekanisme TCP/IP. Setiap *node* bekerja sebagai *member* dari *cluster* yang saling melakukan sinkronisasi. Singkronisasi dilakukan melalui jaringan. Arsitektur DRBD ditunjukkan pada Gambar 1.



Gambar 1 Arsitektur *Distributed Replicated Block Device* [4]

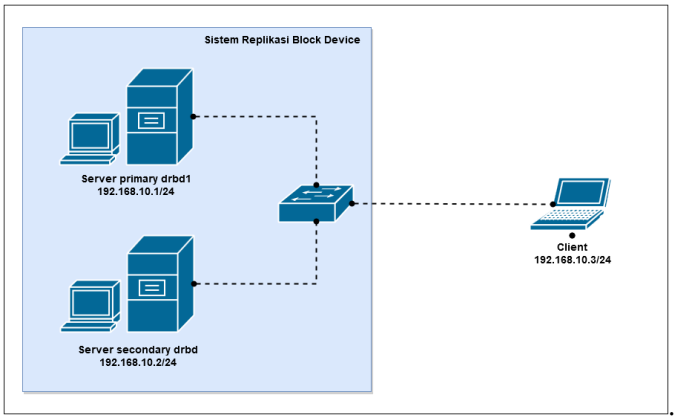
Corosync Pacemaker

Pacemaker bertanggung jawab terhadap proses deteksi adanya *failure* pada *node* dan melakukan *recovery* pada *node*. Pacemaker juga bertanggung jawab terhadap monitoring infrastruktur pada *cluster*. Corosync bertanggung jawab dalam menyediakan layanan *cluster service* dalam bentuk *class-class* yang spesifik yang dapat diakses oleh pengguna melalui *Application Pragrammable Interface* (API). Corosync melakukan abstraksi terhadap *engine cluster* sehingga semua layanan dalam *cluster* dapat bekerja diatas lapisan dari Cluster Engine [14].

2. METODE PENELITIAN

2.1 Analisis

Failover yang akan dibangun pada penelitian ini merupakan *failover cluster*. *Failover cluster* yang dibangun memerlukan infrastuktur baik *hardware* maupun *software*. Topologi fisik sistem disimulasikan pada minimal 2 (dua) buah komputer yang saling terhubung dalam *private Local Area Network* (*LAN*). buah *server* yaitu *server* drbd1 sebagai *node* ke-1 dan *server* drbd2 sebagai *node* ke-2. *Server* drbd1 sebagai *primary* sedangkan *server* drbd2 sebagai *secondary*. N*etwork cluster* yang dibangun antar *node* dengan menggunakan *Corosync Service Engine.* Arsitektur perangkat keras ditunjukkan pada Gambar 2.



Gambar 2 *Arsitektur perangkat keras*

Setiap *node* akan di-*install* Corosyn dan Pacemaker sebagai *software* yang berfungsi untuk membangun *network cluster*. Sedangkan replikasi *block device* dengan memanfaatkan modul dari Linux yaitu *Data Replicated Block Device* (DRBD). Layanan *website* membutuhkan *server* yaitu *web server* dengan memanfaatkan Apache, sedangkan *database server* dengan menggunakan *Database Management System* MySQL.

2.2 Perancangan

Pada tahapan ini dilakukan perancangan dan instalasi komputer-komputer pada suatu jaringan yang menghubungkan satu sama lain. Jaringan bersifat *privat* sehingga hanya menghubungkan komputer saja jaringan lokal. Perancangan pengujian system : Setelah sistem diimplementasikan, dilakukan pengujian untuk menjamin sistem yang dibangun dapat berjalan sebagaimana mestinya. Pengujian yang dilakukan meliputi pengujian tingkat *availability* dan pengujian beban.

2.3 Implementasi

Network cluster diimplementasikan dengan melakukan instalasi Corosync. Corosync berfungsi sebagai *cluster* yang akan melakukan kontrol terhadap *node-node* yang ada didalam sistem. Perlu untuk didefinisikan nama *cluster* dan *node-node* yang ada didalam sistem. Redundansi proses diimplementasikan dengan aktif-pasif *failover cluster* sekaligus berperan sebagai *automatic failover* sistem.

Implementasi *Resource Cluster*

Resource cluster merupakan layanan yang definisikan diatas *cluster*. Layanan dapat berupa aplikasi atau berupa data yang berjalan diatas *cluster*. Pada penelitian ini *resource cluster* yang dibangun antara lain Virtual IP, *Web Server*, MySQL, *Filesystem* dan DRBD. Konfigurasi *resource* pada *cluster* ditunjukkan pada Gambar 3.

```
# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=192.168.10.10
  cidr_netmask=24 op monitor interval=20s
# pcs resource create WebSite ocf:heartbeat:apache
  configfile=/etc/httpd/conf/httpd.conf statusurl="http://localhost/server-
  status" op monitor interval=1min
# pcs resource create DataBase ocf:heartbeat:mysql
  binary=/usr/libexec/mysqld config=/dataweb/datamysql/my.cnf
  datadir=/dataweb/datamysql/data
# pcs resource create WebFS ocf:heartbeat:Filesystem device=/dev/drbd1
  directory=/dataweb/ fstype=ext4 options="noatime,nodiratime"
# pcs resource master WebDataClone WebDRBD master-max=1 master-node-max=1
  clone-max=2 clone-node-max=1 notify=true
# pcs resource create WebDRBD ocf:linbit:drbd drbd_resource="datawebserver"
  op monitor interval="15s"
```

Gambar 3 Konfigurasi *resource cluster*

Implementasi *Failover Cluster*

Pacemaker dilakukan konfigurasi sebagai *Cluster Resource Manager* (CRM). Pada layer ini didefinisikan *resource-resource* yang akan dimonitor oleh Pacemaker. *Resource cluster* merupakan layanan yang definisikan diatas *cluster*. Layanan dapat berupa aplikasi atau berupa data yang berjalan diatas *cluster*. *Resource cluster* yang dibangun antara lain Virtual IP Address, *Web Server* , MySQL, *Filesystem* dan DRBD. Konfigurasi *member* pada *cluster* ditunjukkan pada Gambar 4.

```
totem {
 cluster_name: mycluster
 transport: udpu
}
nodelist {
 node {    ring0_addr: drbd1
    nodeid: 1}
node {     ring0_addr: drbd2
    nodeid: 2} }
```

Gambar 4 Konfigurasi *node* anggota Corosync

Implementasi *Replicated Block Device*

Replikasi *block device* diimplementasikan dengan *menggunakan Data Replicated Block Device* (DRBD). DRBD diimplementasikan sebagai bagian dari *resource agent* pada Pacemaker. DRBD dibangun dengan melakukan replikasi terhadap *block* data dari satu *node* ke anggota *node* lainnya pada *cluster*. Konfigurasi replikasi DRBD ditunjukkan pada Gambar 5.

```
# resource data web server {
volume 0 {
device /dev/drbd1;
disk /dev/fedora-server/data;
meta-disk internal;
}
on drbd1 {
address192.168.10.1:7788;
 }
on drbd2 {
address192.168.10.2:7788;
 }
}
```

Gambar 5 *Script* konfigurasi *DRBD*

Implementasi Pengujian

Pengujian *availability database* dilakukan dengan mengirimkan *query* dari *client.* Pengujian *availability database server* dilakukan dengan memberikan *input* pada *database server* untuk menjalankan *query*. Program dibangun dengan menggunakan bahasa pemrograman Java. Adapun jenis *query* yang dilakukan adalah *insert, update* dan *delete* dengan jenis *query* yang digunakan adalah *write* dan *read*.

3. HASIL DAN PEMBAHASAN

3.1. Hasil Pengujian Availability Database

Pengujian dilakukan ketika *primary server* gagal memberikan layanan sehingga *cluster* akan memindahkan layanan ke *secondary server*. pengujian yang dilakukan adalah transaksi *insert*. Pada transaksi dilakukan pengujian dengan 10 kali pengujian. Pada transaksi *insert* dengan 1.000, 5.000 dan 10.000 transaksi didapatkan tingkat *availability* dari sistem sebesar 98,907, 99,800 dan 99,882. Hasil perbandingan *availability* pada transaksi *insert* ditunjukkan pada Tabel 2.

Tabel 2 Hasil perbandingan *availability* kegagalan transaksi *insert*

Jenis Uji	Jumlah Data Input	Jumlah Eksekusi	Waktu Eksekusi (md)	*Availability*
Insert	1.000	994	91.383,50	98,907
	5.000	4.995	310.414,80	99,800
	10.000	9.994	1.023.374,20	99,882

Pengujian transaksi *update* dilakukan dengan simulasi yang dilakukan tetap sama seperti pengujian *insert*, hanya saja *query* yang dilakukan diubah menjadi *update*. Perbandingan tingkat *availability* yang didapatkan untuk 1.000 adalah 99,026, untuk 5.000 didapatkan 99,808 dan untuk transaksi 10.000 diperoleh 99,896. Hasil perbandingan *availability* pada transaksi *update* ditunjukkan pada Tabel 3.

Tabel 3 Hasil perbandingan *availability* kegagalan transaksi *update*

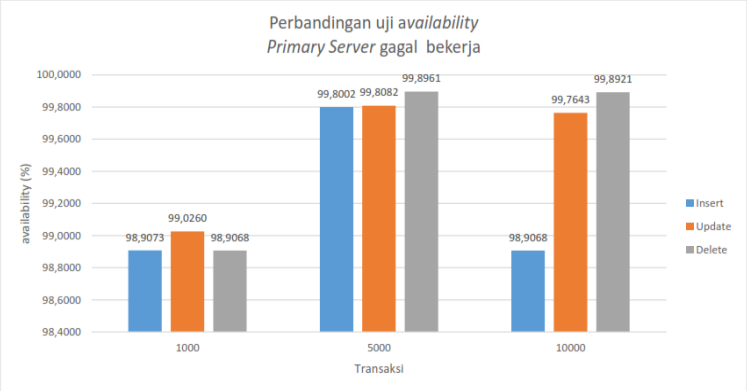
Jenis Uji	Jumlah Data Input	Jumlah Eksekusi	Waktu Eksekusi (md)	*Availability*
Update	1.000	995	40.961,00	99,026
	5.000	4.995	165.473,50	99,808
	10.000	9.994	794.513,10	99,896

Simulasi *query delete* tetap menggunakan 1.000, 5.000 dan 10.000 transaksi. Dengan hasil yang didapatkan yaitu 99,907, 99,764 dan 99,892 untuk pengujian dengan jumlah *query* 1.000, 5.000 dan 10.000 transaksi. Hasil perbandingan *availability* pada transaksi delete ditunjukkan pada Tabel 4.

Tabel 4 Hasil perbandingan *availability* kegagalan transaksi *delete*

Jenis Uji	Jumlah Data Input	Jumlah Eksekusi	Waktu Eksekusi (md)	*Availability*
Delete	1.000	994	113.059,10	98,907
	5.000	4.994	449.439,80	99,764
	10.000	9.994	1.257.266,40	99,892

Pengujian yang dilakukan pada saat terjadi kegagalan pada *primary master*. Semakin besar data yang akan diproses tingkat *availability* yang ditunjukkan sistem menjadi lebih besar, hal ini menujukkan bahwa kegagalan sistem bersifat konstan pada semua pengujian. Grafik perbandingan untuk pengujian *insert, update* dan *delete* dengan jumlah transaksi 1.000, 5.000 dan 10.000 ditunjukkan pada Gambar 6.



Gambar 6 Grafik perbandingan *availability* pada pengujian keseluruhan transaksi pada saat kegagalan sistem

3.2. Hasil Pengujian Beban Database Saat Terjadi Kegagalan

Pengujian beban *database* dilakukan dengan mengukur tingkat kemampuan *server* dalam menangani *query* dalam satuan waktu. Untuk *query insert* didapatkan pada pengujian yang pertama didapatkan tingkat *availability* sebesar 99,9991 sedangkan pada pengujian ke-2 didapatkan tingkat *availability* 99,9995 dengan rerata tingkat *availability* sebesar 99,9993. Hasil perbandingan beban pada transaksi *insert* ditunjukkan pada Tabel 5.

Tabel 5 Hasil pengujian beban *database* dengan transaksi *insert*

No	Waktu Eksekusi (jam)	Berhasil (transaksi)	Gagal (transaksi)	*Availability*
1	24	1.358.490	6	99,99912
2	24	1.395.872	3	99,99957

Untuk pengujian *update* tingkat *availability* yang didapatkan sebesar 99,9995 untuk pengujian pertama dan 99,9993 pada pengujian kedua. Rerata *availability* pengujian sebesar 99,9994. Hasil perbandingan beban pada transaksi *update* ditunjukkan pada Tabel 6.

Tabel 6 Hasil pengujian beban *database* dengan transaksi *update*

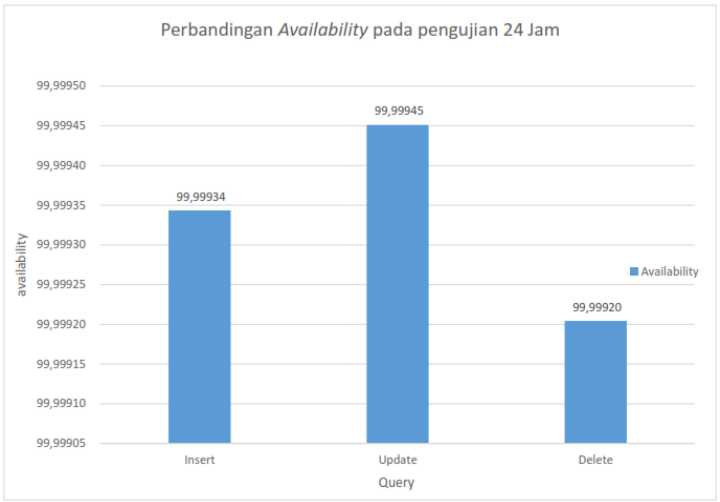
No	Waktu Eksekusi (jam)	Berhasil (transaksi)	Gagal (transaksi)	*Availability*
1	24	1.846.908	4	99,99956
2	24	1.804.853	6	99,99933

Untuk pengujian dengan *query delete* didapatkan hasil untuk pengujian pertama sebesar 99,9991 dan pengujian ke-2 sebesar 99,9992 dengan rerata *availability* pengujian sebesar 99,9994. Hasil perbandingan beban pada transaksi *delete* ditunjukkan pada Tabel 7.

Tabel 7 Hasil pengujian beban *database* dengan transaksi *insert*

No	Waktu Eksekusi (jam)	Berhasil (transaksi)	Gagal (transaksi)	*Availability*
1	24	1.388.880	6	99,99913
2	24	1.374.465	5	99,99927

Dari pengujian beban dapat dilihat perbadingan tingkat *availability* terhadap *query* yang dikerjakan. *Query update* memiliki tingkat *availability* yang sedikit lebih baik dibandingkan dengan *query insert* maupun *delete*. Grafik perbandingan *availability* pada pengujian 24 jam ditunjukkan pada Gambar 7.



Gambar 7 Grafik perbandingan *availability* dengan transaksi *insert*, *update* dan *delete* pada pengujian 24 jam

3.3. Pengujian Availability Web Server

Web server memberikan layanan 100% jika tidak terjadi permasalahan pada *cluster*, sedangkan jika terjadi kegagalan sistem dimana *primary server* gagal maka akan dialihkan ke *secondary server*. Pada percobaan peralihan dari *primary server* ke *secondary server*, dengan percobaan selama 60 menit didapatkan tingkat ketersediaan sebesar 99,85 sedangkan pada

percobaan yang lebih lama yaitu 24 jam, didapatkan hasil tingkat *availability* mencapat 99,992%. Hasil perbandingan *availability webserver* pada transaksi ditunjukkan pada Tabel 8.

Tabel 8 Hasil perbandingan *availability web server* jumlah pengguna konkuren pada *website*

No	Uji Kasus	*user*	Jumlah Transaksi	Transaksi berhasil	*Availability*
1	Pengujian *website* selama 60 Menit	1	7.086	7.086	100
2	Pengujian *website* dengan percobaan konkuren user 50 selama 60 Menit	50	580.068	580.068	100
3	Pengujian *website* dengan percobaan konkuren user 50 selama 60 menit dengan terjadi kegagalan pada sistem	50	340.134	339.880	99,85
4	Pengujian *website* dengan percobaan kegagalan sistem sebanyak 1 kali dalam waktu 24 Jam	50	8.195.072	8.194.752	99,992

Pengujian diatas merupakan pengujian dengan memanfaatkan HTTP *Request* yang bersifat *single request*. Pada web dinamis proses *login* memanfaatkan *session* yang berfungsi untuk autentikasi *user* yang sedang memanfaatkan layanan. Pengujian menunjukkan bahwa failover tidak mempengaruhi session yang berjalan pada client. Pengujian pengaruh failover terhadap session ditunjukkan oleh Tabel 9.

Tabel 9 Hasil pengujian terhadap *session* ketika *login* ke *web server*

No	Skenario Pengujian	*Keterangan*	Status
1	*User login* ke sistem pada kondisi normal	*Session* konsisten	Berhasil
2	*User login* ke sistem pada saat *secondary server* sebagai *standby node*	*Session* konsisten	Berhasil
3	Pengujian ketika terjadi kegagalan sistem, dimana *standby node* berpindah dari *primary server* ke *secondary server*	*Session* konsisten	Berhasil

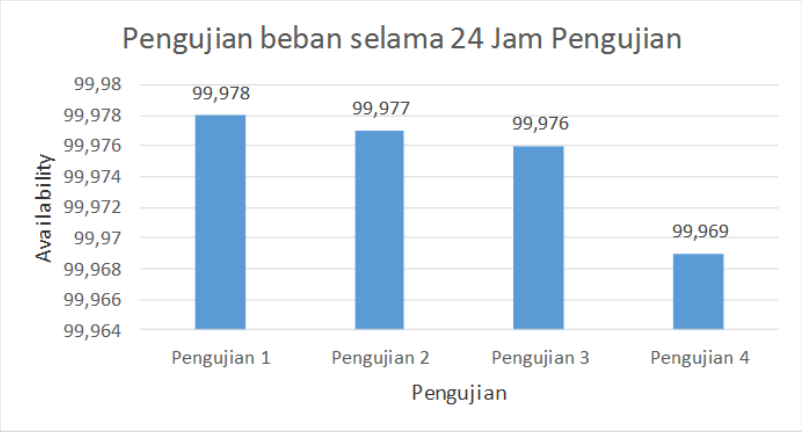
3.4. Pengujian Beban Web Server

Pengujian dilakukan dengan melakukan simulasi dengan menggunakan pengguna yang dapat mengakses kehalaman *website* secara bersama-sama. Hasil pengujian pada sistem selama 24 jam memberikan hasil rata-rata sistem memiliki tingkat ketersediaan 99,9% atau *three fives* pada uji coba HTTP *Request*. Hasil pengujian beban terhadap *web server* ditunjukan oleh Tabel 10.

Tabel 10 Hasil pengujian beban pada *web server*

No.	*Waktu Uji (Jam)*	*HTTP Request*	*Fault*	Data *Hits*	*Availability*
1	24	2.443.104	264	2.442.840	99,978
2	24	2.425.392	278	2.425.114	99,977
3	24	2.471.184	302	2.470.882	99,976
4	24	2.452.752	383	2.452.369	99,969

Hasil pengujian pada sistem selama 24 jam memberikan hasil rata-rata sistem memiliki tingkat ketersediaan 99,9% atau *three fives* pada uji coba HTTP *Request*. Grafik Pengujian yang dilakukan selama 24 jam ditunjukkan oleh Gambar 8.



Gambar 8 Grafik pengujian beban web server

4. KESIMPULAN

Berdasarkan pada pembahasan yang telah diuraikan pada bagaian sebelumnya dapat ditarik kesimpulan yaitu:

- Pengimplementasian *failover cluster* dan replikasi data dapat menghasilkan *Web server* dengan tingkat *availability* yang tinggi, pada penelitian ini pengujian ketersediaan yang dapat dilayanani meliputi *web server* dan *database server*.
- Kegagalan pada pengujian *web server* terjadi pada HTTP *request* yang bersifat *single request* saja, proses *session* pada *website* tidak terpengaruh pada saat *failover* yang terjadi.
- Tingkat ketersediaan *database* mencapai 99,999 (*five nines*) sedangkan tingkat Ketersediaan terhadap *website* yang didapat mencapai 99,97 (*three nines*).

5. SARAN

Berdasarkan pengujian yang telah dilakukan, masih banyak kekurangan pada sistem sehingga perlu dikembangkan lagi agar kinerjanya lebih baik. Perlu untuk memperbaiki kinerja IP Virtual agar *website* dapat bekerja maksimal meskipun terjadi kegagalan sistem. Implementasi *Network Cluster* dan *Distributed Replicated Block Device* belum dapat memenuhi syarat sebagai sistem yang *mission critical* dimana memerlukan *availability* diatas 99,999 atau *five nines.*

DAFTAR PUSTAKA

[1] Atmaja, A. P., 2013, *Tinjauan Implementasi Fragmentasi Elastis pada Database Non-Relational untuk Website Forum Diskusi*, Tesis, Magister Ilmu Komputer FMIPA Universitas Gadjah Mada, Yogyakarta.

[2] Paudyal, U., 2011, *Scalable web application using node.JS and CaouchDB*, Institutionen för informationsteknologi, Department of Information technology, Uppsala: Sweden.

[3] Lin, Z., 2009, Research and Implement of High Availability for *Web* Server Cluster System, *Proceedings of 2009 Conference on Communication Faculty*, Scientific Research, Beijing.

[4] Ellenberg, L., 2009, *DRBD 9 & Device Mapper, Linux Block Level Storage Replication*, LINBIT, Vienna, Austria.

[5] Setyorini, T. A., 2010, *Rancang Bangun Sistem Informasi Akademik yang Fault Tolerance*, Tesis, Magister Ilmu Komputer FMIPA Universitas Gadjah Mada, Yogyakarta.

[6] Oei, S., 2011, *Rancang Bangun Fault Tolerance pada Sistem Database Pada Sistem Database Untuk Aplikasi Point Of Sale,* Tesis, Magister Ilmu Komputer FMIPA Universitas Gadjah Mada, Yogyakarta.

[7] Anggraini, L. H., 2012, *Rancang Bangun Highly Available dan Reliable Server Database dengan Automatic Failover*, Tesis, Magister Ilmu Komputer FMIPA Universitas Gadjah Mada, Yogyakarta.

[8] Afif, M. F., Suryono, T., 2013. Implementasi Disaster Recovery Plan Dengan Sistem Fail over Menggunakan DRBD dan Heartbeat Pada Data Center FKIP UNS. *Indonesian Jurnal on Networking and Security (IJNS) Volume 2 No 2 – April 2013 - ISSN: 2302-5700* (p. 64-69).

[9] Schmidt, K., 2006, *High Availability and Disaster Recovery Concepts, Design, Implementation*, Springer-Verlag, Berlin.

[10] Tanenbaum, A. S. dan Steen, M. V., 2002, *Distributed System Principles and Paradigms*, Prentice Hall.

[11] Marcus, E., Stern, H., 2003, *Blueprints for high availability 2nd Edition*, John Wiley & Sons: Indianapolis.

[12] Depuydt, J., 2014, *Building a high-available failover cluster with Pacemaker*, Corosync & PCS, http://jensd.be/?p=156, Diakses 24 April 2015.

[13] Lowe, D., 2005, *Networking for Dummies*, Seventh Edition, Wiley Publishing: New York.

[14] Dake, C. S., Caulfield, C., Beekhof, A., 2008, the Corosync Cluster Engine, *Linux Symposium, Vol. 1*, Ottawa, Ontario, Canada.