INFORMATICS AND DIGITAL EXPERT (INDEX) - VOL. 5 NO. 2 (2023) 51-59
Terbit *online* pada laman web jurnal : https://e-journal.unper.ac.id/index.php/informatics
| ISSN (Print) 2775-2208 | ISSN (Online) 2715-0453 |

Merancang IP Failover & Replikasi Database Menggunakan Heartbeat Pada Komputer Server

(STUDI KASUS : SMK PERIWATAS TASIKMALAYA)

Rangga Agung Nugraha. B.[1], Ir. Ruuhwan, S.T., M.Kom.[2], Rudi Hartono, S.T., M.Kom.[3]

[1]*Fakultas Teknik Universitas Perjuangan*
[1]*Program Studi Teknik Informatika*
[1]*Universitas Perjuangan, Jl. Peta No.177 Kahuripan , Tasikmalaya 46115, Indonesia*
e-mail: [1]*ranggaagungnugraha421@gmail.com*, [2]*1603010009@unper.ac.id*

INFORMASI ARTIKEL

Sejarah Artikel:
Diterima Redaksi : 26 Juli 2023
Revisi Akhir : 01 November 2023
Diterbitkan *Online* : 30 November 2023

***Kata Kunci* :**
Failover Heartbeat, Ubuntu Server, System Jaringan.

Korespondensi :

Telepon / Hp : +62 81905425351
E-mail :
[1]ranggaagungnugraha421@gmail.com
[2]1603010009@unper.ac.id

A B S T R A K

Pada saat ini manfaat jaringan komputer sudah sangat banyak dirasakan manfaatnya, apalagi dalam dunia komunikasi yang serba cepat ini, jaringan komputer berperan begitu sangat vital dalam kegiatan pedistribusian informasi yang cepat. Semua komponen yang tergabung dalam koneksi jaringan komputer tersebut harus mampu saling mendukung untuk menghasilkan sistem yang kokoh dan handal untuk bisa melayani setiap permintaan informasi yang dibutuhkan oleh pengguna. Hal ini menjadi sebuah perhatian di bidang jaringan komputer untuk menjalankan dan menggabungkan teknik failover dalam server real atau virtual sebagai media pendistribusi informasi. SMK Periwatas Kota Tasikmalaya merupakan jenjang sekolah kejuruan yang terletak di Kecamatan Cipedes, Kota Tasikmalaya. Saat ini SMK Periwatas masih menggunakan satu server utama yang dimana permasalahan saat ini adalah apabila server utama mati tidak ada server lain yang membackup, maka dari itu penulis akan membuat dua buah server yang dimana server pertama akan menjadi server utama dan server kedua akan dijadikan server cadangan atau backup yang saling tersinkronisasi dan dapat melakukan failover.

1. PENDAHULUAN

Data dan informasi merupakan sesuatu yang sangat begitu penting saat ini. Agar informasi atau data dapat di akses, maka dibutuhkan *server* untuk mengolah serta menyimpan data. *Server* dituntut harus dapat beroperasi selama 24 jam agar data yang disimpan atau diolah dalam *server* tersebut dapat selalu diakses. Namun terkadang juga *server* yang handal sekalipun tetap dapat mengalami kegagalan. *Server* yang mengalami kegagalan dapat disebabkan oleh berbagai faktor, seperti faktor adanya kegagalan pada *hardware*, *software* dan lain sebagainya [1]

Untuk mengatasi hal tersebut, maka munculah sebuah solusi yaitu dengan adanya implementasi *server* yang memiliki sifat ketersediaan tinggi atau juga dapat disebut dengan *high availability server*. Implementasi *high availability server* dapat dilakukan dengan cara memasang perangkat lunak klaster pada *server* utama dan *server backup*, perangkat lunak heartbeat dapat digunakan untuk implementasi *high availability server*.

High availability server akan dapat terus berjalan untuk melayani permintaan dari pengguna karena menerapkan teknik *failover*. Teknik *failover* diterapkan pada *server*, *server* diduplikasi dan *server* duplikasi tersebut dijadikan satu klaster dengan *server* utama. Tujuan utama dari *failover* itu sendiri adalah agar pengguna tetap dapat mengakses ke *server* walaupun *server* sedang mengalami kegagalan pada *software*, *hardware* ataupun yang lainnya yang bisa mengakibatkan *server* tersebut tidak dapat diakses. Secara sederhana cara kerja dari *failover* itu sendiri adalah jika *server* utama mengalami kegagalan atau gangguan pada sistem maka secara otomatis *server backup* akan mengambil alih tugas dari *server* utama atau *membackupnya*. Dengan adanya sistem implementasi ini dapat menghindari *single point of failure*. *Single point of failure* itu sendiri adalah sebuah komponen penting dalam sistem yang dimana jika komponen ini berhenti maka keseluruhan sistem akan berhenti pula. Sistem yang memeliki sifat ketersediaan tinggi tidak perlu membutuhkan komponen *single point of failure*, ini berarti bahwa jika ada kegagalan pada sebuah komponen tidak akan menyebabkan sistem berhenti beroperasi [2]

Untuk teknologi *failover* yang akan dibahas itu sendiri adalah sebuah teknik untuk menghindari kegagalan pada sistem, dengan adanya implementasi *failover* pada sistem, maka sistem akan terus berjalan untuk melayani pengguna karena *failover* sendiri dapat memindahkan *traffic* dari komponen yang mengalami gangguan atau kegagalan ke komponen *backup*. Proses *failover* sendiri dapat dirancang secepat mungkin setelah saat terjadinya kegagalan, didalam *failover* terdapat proses *failback*. *Failback* sendiri adalah proses kembalinya komponen yang sebelumya mengalami kegagalan atau dari *server backup* dikembalikan ke *server* utama, Dari *failover* sendiri terdapat dua model, yaitu *failover active-active* dan *failover active-standby*.

Untuk menjalankan teknik *failover* itu sendiri harus menggunakan *software* yang bernama *heartbeat*. Dimana *heartbeat* adalah suatu perangkat lunak yang merupakan bagian dari proyek *linux high availability*. *Heartbeat* sendiri merupakan perangkat lunak *failover* berbasis klaster, yang memastikan bahwa *resources* akan selalu tersedia untuk diakses. Maka dalam jangka waktu

tertentu masing-masing *server* dari *heartbeat* akan mengirimkan paket melawati jaringan kepada *server heartbeat* lain sebagai sinyal tetap hidup. Jika tidak ada pesan *heartbeat* dari *server* utama, maka diasumsikan *server* utama telah mati kemudian *server backup* menjalankan teknik *failover* serta mengambil alih peran dari *server* utama [3]

Untuk melakukan proses *backup* sendiri perlu adanya sebuah replikasi *database*, yang dimana replikasi *database* adalah sebagai suatu proses *mengcopy* atau mentransfer data dari suatu *database* ke *database* lain yang tersimpan pada komputer berbeda. Replikasi data sendiri dapat meningkatkan kinerja dan ketersediaan aplikasi, karena adanya berbagai pilihan alternatif terhadap akses data yang disediakan. Ada beberapa contoh yang dapat diambil seperti sebuah aplikasi mungkin biasanya mengakses data secara lokal daripada melalui *server* yang lokasinya jauh dengan tujuan untuk meminimalkan lalu lintas jaringan dan mecapai performa yang maksimal. Jika *server* lokal mengalami kegagalan, maka aplikasi tersebut masih dapat terus berfungsi, karena *server* yang lainnya memiliki replikasi data dari *server* lokal. Maka dari itu dengan adanya fungsi utama *failover* klastering bertujuan untuk membantu menjaga akses *client* ke aplikasi dan sumber daya *server*, bahkan ketika terjadinya kegagalan pada *software*, ataupun kegagalan fungsi *server* yang mengakibatkan *server* berhenti bekerja. Teknologi ini diharapkan dapat menjadi solusi dalam mengatasi kegagalan *server* ketika terjadi gangguan ataupun perawatan dan mengimplementasikan suatu sistem *failover virtual* komputer klaster untuk mengatasi kegagalan *server*.

2. LANDASAN TEORI

2.1 Pengertian Server

Server adalah suatu sistem komputer yang memiliki layanan khusus berupa penyimpanan data. *Server* akan menyimpan beragam jenis dokumen dan menyediakan informasi umtuk pengguna atau pengunjungnya. Secara umum server merupakan tempat yang berisi berbagai macam data atau dokumen yang dibutuhkan klien [4]

Komputer klien membutuhkan *server* agar dapat terhubung pada jaringan atau untuk meminta data yang terdapat pada *server*. Secara umum fungsi utama *server* adalah melayani dan bertanggung jawab penuh terhadap permintaan data dari komputer klien. Selain itu, fungsi *server* juga hak akses ke dalam jaringan yang bisa digunakan oleh komputer klien. Maka dari itu komputer sering juga berisi berbagai data informasi, dimana *server* tersebut memiliki tugas untuk memberikan layanan bagi para klien yang terhubung dengannya.

2.2 Pengertian Failover

Failover sendiri menyediakan solusi *high availability server* dimana jika terjadi kegagalan pada perangkat keras seperti *power supply* mati yang menyebabkan *server* mati total, maka *server* lain *(backup)* yang akan mengambil alih fungsi dari *server* utama yang mati, sehingga komputer klien tidak mengetahui jika terjadi kegagalan pada *server*, karena proses yang dilakukan pada *server* yang mati akan dilanjutkan oleh *server backup*. Konsep konfigurasi *failover clustering* adalah membuat satu *server* sebagai *master server* dan *server* yang lain menjadi *slave server*, dimana saat *server* dalam keadaan normal maka *master server* akan menangani semua permintaan dari klien. *Slave server* akan mengambil alih tugas dari *master server* apabila *master server* tidak berfungsi atau mati [5]

Failover merupakan teknik yang dapat menerapkan beberapa rute untuk dapat mencapai suatu destination *network*. Akan tetapi dalam keadaan standar hanya ada satu *link* atau jalur yang digunakan. *Link* atau jalur yang lain akan berfungsi sebagai cadangan dan hanya akan digunakan apabila link utama telah terputus.

2.3 Pengertian Heartbeat

Perangkat lunak *heartbeat* merupakan bagian dari proyek *linux high availability*. *Heartbeat* merupakan sebuah perangkat lunak *failover* berbasis klaster, yang memastikan bahwa *resources* akan selalu tersedia untuk diakses. Dalam jangka waktu tertentu, masing-masing *node* dari *heartbeat* akan mengirimkan paket melewati jaringan kepada *node heartbeat* lain sebagai sinyal tetap hidup. Jika tidak ada pesan *heartbeat* dari *node master*, maka diasumsikan bahwa *node master* telah mati, kemudian *node backup* akan menjalankan teknik *failover* serta mengambil alih peran dari *node master*.

Heartbeat merupakan sebuah aplikasi yang dapat mendeteksi apabila *server* utama *down* maka *heartbeat* akan secara otomatis mengarahkan peran *server* utama kepada *server backup*. *Heartbeat* menjalankan *script* inisialisasi untuk menjalankan *service* lain saat *heartbeat* dijalankan atau bisa juga mematikan *service* lain saat *heartbeat* dimatikan.

2.4 Pengertian Hight Availability

High availability dalam teknologi informasi dapat diartikan sebagai sebuah sistem atau komponen yang dapat beroperasi secara terus menerus dalam jangka waktu yang lama tanpa mengalami gangguan. Pada dasarnya, *high availability* mengimplementasikan satu atau lebih server *backup* dalam *mode standby*, yang kemudian dapat menjadi siaga dalam beberapa saat ketika ditemukan kegagalan dalam *server master*.

High availability adalah sebuah desain protokol sistem dan implementasi dari sebuah jaminan tentang sebuah kepastian dari kelanjutan operasional dalam sebuah periode yang ditentukan. Kata *availability* mengacu kepada kemampuan sekelompok pengguna agar dapat mengakses ke dalam sistem seperti menambah pekerjaan baru, mengubah pekerjaan yang sebelumnya, atau mengumpulkan hasil dari pekerjaan yang sebelumnya.

2.5 Ubuntu Server

Ubuntu merupakan sebuah sistem operasi yang bersifat *open source* dan dapat dijalankan di *desktop, cloud* atau *IoT.*

Ubuntu 14.04 LTS sendiri adalah rilis Ubuntu yang ke-24 dan merupakan seri *Long Time Support* atau LTS

yang ke-6. LTS adalah versi Ubuntu yang di *support* lebih lama dari versi-versi Ubuntu yang biasa yaitu 4 tahun versi *desktop* dan 5 tahun untuk versi *server*. Namun pada Ubuntu 14.04 ini, seri LTS akan sama-sama di support selama 5 tahun baik itu untuk versi *desktop* maupun *server*. Seri LTS lebih ditujukan untuk kehandalan dan kestabilan sistem, sehingga sangat cocok digunakan untuk komputer *server* [6]

2.6 Virtualbox

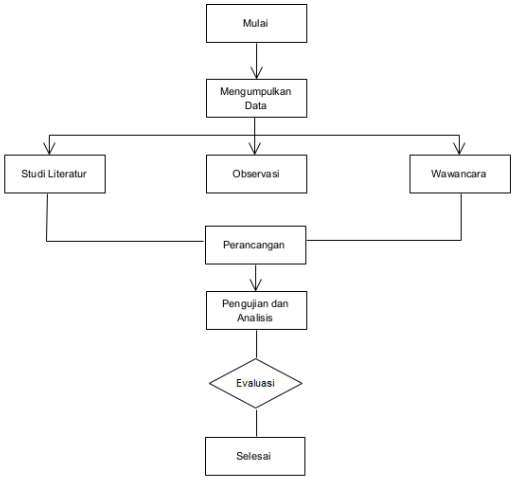
VirtualBox adalah perangkat lunak virtualisasi untuk menginstal sistem operasi. Kata lain dari virtualisasi yaitu mengubah atau mengubah sesuatu menjadi bentuk simulasi dari nyata atau nyata. Misalnya jika komputer telah di *install Windows* pada komputer, maka dari itu juga jika ingin dapat menjalankan sistem operasi lain yang diinginkan dalam sistem operasi *Windows*. Jika pengguna ingin mencoba menginstal sistem operasi lain, maka tidak perlu menginstal ulang komputer, maka hanya memerlukan perangkat lunak *VirtualBox* ini untuk mencoba menginstal sistem operasi. Maka dari itu pengguna bisa menginstal *Linux* di *Windows* secara mudah hanya dengan menggunakan *VirtualBox*.

3. METODOLOGI PENELITIAN

3.1 Pengumpulan Data

Metode yang digunakan dalam penelitian ini adalah metode eksperimental. Metode eksperimen termasuk dalam metode kuantitaf yang dilakukan dilaboratorium dengan adanya perlakuan. Tujuan dari penelitian ini adalah untuk mendapatkan hasil rancangan *failover* yang dapat membackup *server* utama apabila terjadi kerusakan atau kegagalan pada sistem, sehingga adanya teknik *failover* ini dapat membantu *client* yang sedang mengakses komputer tanpa adanya kendala apapun.

Adapun diagram alir *(flow chart)* yang dimaksudkan untuk memberikan gambaran tentang proses pengerjaan pada tugas akhir ini. Diagram alir proses pengerjaan tugas akhir dapat dilihat pada gambar dibawah ini.



Gambar 3.1 Tahapan Penelitian

1. Pengumpulan Data

Metode pengumpulan data adalah teknik atau cara yang dilakukan oleh peneliti untuk mengumpulkan data.

a. Studi Literatur

Tahapan ini adalah cara yang dipakai untuk menghimpun data-data atau sumber-sumber yang berhubungan dengan topic yang diangkat dalam suatu penelitian. Studi literature bisa didapat dari berbagai sumber jurnal, buku, dokumentasi, internet dan pustaka.

b. Observasi

Teknik observasi yang dilakukan melalui peninjauan langsung ke wilayah sekolah SMK Periwatas Kota Tasikmalaya yang akan diteliti. Yakni merancang *IP Failover* dan Replikasi *Database* Menggunakan *Heartbeat* Pada Komputer *Server*.

c. Wawancara

Dengan adanya wawancara dapat dilakukan secara terstruktur maupun tidak terstruktur dan dapat dilakukan melalui tahap. Maka penulis melakukan wawancara dengan Bapak Ino Cahyono selaku ketua laboratorium computer di SMK Periwatas Kota Tasikmalaya dengan pembahasan topic mengenai banyak pengguna *(client)*, model konektivitas jaringan dan pemetaan jaringan untuk kebutuhan sistem *server* yang akan diterapkan pada komputer *server.*

2. Perancangan

Merancang suatu sistem dimulai dari perancangan konfigurasi *IP Address* lalu menginstalasi *software heartbeat*, lalu dilanjutkan dengan membuat program untuk menjalankan sebuah sistem *failover* pada komputer *server*.

3. Pengujian

Dalam tahap ini sistem yang telah dirancang, dilakukan pengujian untuk melihat apakah ada kesalahan-kesalahan sistem pada saat dijalankan. Tujuan dari pengujian ini adalah untuk menemukan kesalahan fungsi pada sistem atau agar bisa dilakukannya sebuah perbaikan pada sistem jika terjadinya kegagalan sistem. Setelah itu menganalisi bertujuan untuk mengetahui proses bagaimana cara bekerjanya sistem *failover* yang dibuat atau untuk mengetahui data-data yang dihasilkan dari proses pengujian sistem *failover* tersebut.

4. Evaluasi

Tahapan evaluasi adalah tahapan akhir dari model diagram alir ini. Evaluasi dilakukan untuk mengevaluasi apakah rancangan yang dibuat sudah sesuai atau tidak dengan yang diharapkan. Hasil dari tahapan ini adalah laporan hasil pengujian sistem dan digunakan

untuk memberi umpan balik kepada pihak sekolah yang mana sebagai tempat untuk sebuah penelitian.

4. HASIL DAN PEMBAHASAN

4.1 Implementasi Pengumpulan Data

Penulis melakukan wawancara dengan pihak laboratorium komputer sekolah berdasarkan masalah yang sedang di hadapi oleh wali kelas, kemudian mencari solusi penyelesaiannya. Untuk memenuhi tujuan tersebut, penulis akan membuat sebuah perbandingan antara perancangan menggunakan *software Heartbeat* untuk pemrograman perangkatnya, serta menggunakan operasi sistem Ubuntu Server versi 14.04 untuk menjalankan *Heartbeat* . Untuk instalasi *Ubuntu Server* disini penulis menggunakan *Virtual Machine* yang bernama *Virtualbox.*

Penelitian ini bertujuan mengimplementasikan sistem *cluster* komputer sebagai salah satu solusi untuk mengatasi kegagalan fungsi dari *server*.dengan menggunakan *Linux* sebagai *platform* simulasinya. *Cluster* komputer yang dibangun terdiri dari dua buah *server* dengan sistem operasi *Ubuntu Server 14.04*. Kedua *server* harus terinstal *software Heartbeat* yang mana berfungsi sebagai *failover* dan replikasi *Distributed Replicated Block Device (DRBD)* yang mana berfungsi sebagai sinkronisasi data [7]

Berdasarkan hal tersebut, maka dalam penelitian ini dilakukan implementasi suatu *server cluster* yang dapat menjamin ketersediaan layanan *web server* untuk *user*, serta dapat dipastikan bahwa *website* dapat diakses walaupun terjadi kerusakan, dikarenakan dalam *server cluster* terdapat dua buah komputer yang bertindak sebagai *server* aktif atau *server* utama dan *server backup* sebagai *server* cadangan.

Maka dengan adanya *server cluster* ini diharapkan mampu memberikan sistem layanan yang terbaik untuk *user* dari sistem *website*. Dalam komputer *cluster* ini, apabila dua buah *server* menjalankan *web application* dan berjalan dengan normal dimana *web application* dijalankan oleh *server* utama dan *server backup* sebagai *server* cadangan maka sistem berjalan dengan normal. Namun jika server utama mengalami kegagalan pada *hardware* maupun *service*, maka secara otomatis *server backup* akan aktif dan menggantikan tugas dari *server* utama [8]

4.2 Implementasi Perancangan

Dalam implementasi *server cluster* ini diperlukan instalasi *web server* pada kedua komputer *server*, instalasi dan konfigurasi *DRBD* dan *Heartbeat* pada kedua *server* sehingga kedua *server* tersebut dapat menjalankan sinkronisasi data dan melakukan *failover*. Konsep dari *server cluster* adalah mem*backup server* pada saat *server* utama mengalami *down*, sehingga *server* cadangan akan menggantikan kerja dari *server* utama. *IP address* yang terdapat pada masing-masing *server* terdiri dari beberapa seperti terlihat pada tabel berikut [9]

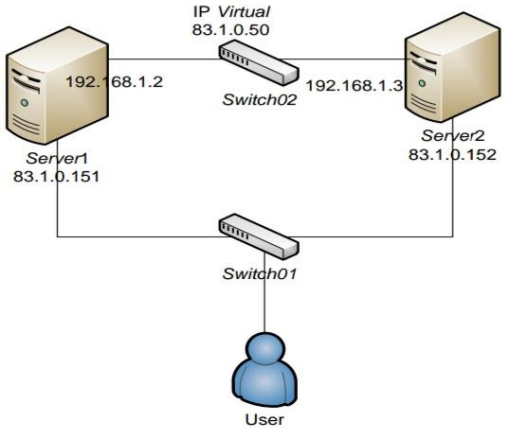
Tabel 4.2 Skema IP Address

Skema	Client	Server 1	Server 2
IP Address	192.168.50.21	192.168.50.22	192.168.50.23
Netmask	255.255.255.0	255.255.255.0	255.255.255.0
Network	192.168.50.1	192.168.50.1	192.168.50.1
Broadcast	192.168.50.254	192.168.50.254	192.168.50.254
IP Virtual	192.168.50.24	192.168.50.24	192.168.50.24

Dari Tabel di atas dapat diketahui bahwa terdapat satu *IP Virtual* yang digunakan dalam implementasi ini.

4.3 Desain Topologi Jaringan

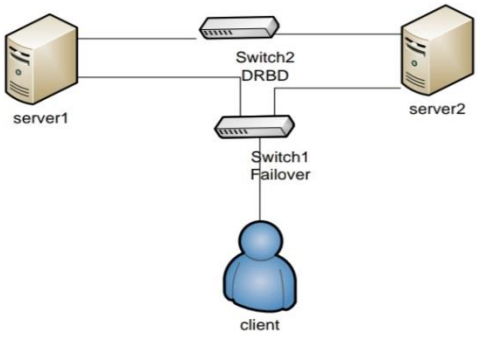
Untuk membangun sebuah jaringan diperlukan desain untuk mengetahui rancangan dari sebuah jaringan yang akan dibangun. Topologi jaringan *server cluster* yang akan dibangun terlihat pada Gambar berikut.



Gambar 4.3 Topologi Server Cluster

4.4 Arsitektur Jaringan

Arsitektur sistem merupakan penjelasan komponen-komponen yang digunakan dalam sebuah jaringan. Dalam implementasi *server cluster* yang akan dibangun mempunyai arsitektur sistem jaringan seperti dalam gambar berikut.



Gambar 4.4 Arsitektur Sistem

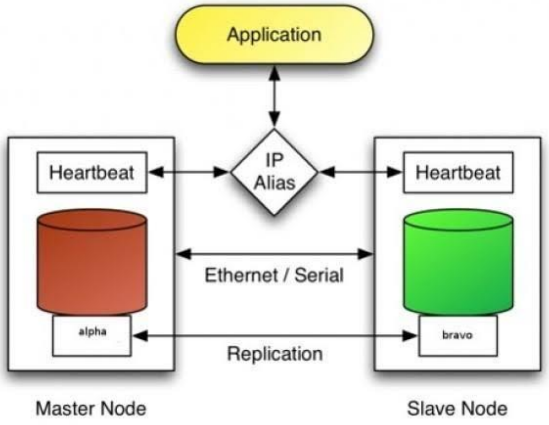
4.5 Instalasi dan Konfigurasi Heartbeat

Sebagai solusi terhadap masalah tersebut, maka sebuah *server* harus mempunyai *server backup* yang mampu meng*handle* tugas dari *server* utama, dan yang

terpenting adalah proses *recovery* sistem berjalan dengan cepat sehingga *client* tidak merasakan bahkan tidak mengetahui adanya kerusakan pada sistem *server*. Maka untuk itu diperlukan sebuah pengetahuan tentang proses *backup failover*, yaitu dimana jika terjadi kendala pada *server* utama maka dapat di *recovery* dengan *server backup* dengan cepat.

Maka pada panduan ini penulis akan membuat sebuah *LAMP (Linux Apache Mysql & PHP) Server*, dimana penulis akan membuat *server* utama dan *server backup* yang saling tersinkronisasi dan dapat melakukan *failover*. Maka dari itu untuk teknologi *failover* yang akan dibahas pada bagian ini, disini penulis akan menggunakan *tools heartbeat*. Sesuai dengan namanya, *heartbeat* akan bekerja layaknya monitor detak jantung, dimana *heartbeat* akan selalu memonitor kondisi dari *server* utama, sehingga apabila terjadi kegagalan atau kerusakan pada *server* utama *heartbeat* akan dengan cepat mengatur *failover* ke *server backup*, dimana semua *server* yang berjalan akan dipindahkan langsung ke *server backup* [10]

Heartbeat akan memanfaatkan satu buah *IP virtual* yang akan digunakan sebagai *IP cluster*, yang dimana *IP* tersebut akan menjadi acuan dalam memindahkan *service* dan data dari *server* utama ke *server backup*. Dibawah ini merupakan proses alur diagram *LAMP Server failover* dan *heartbeat*:



Gambar 4.5 Digram *LAMP Server Failover & Heartbeat*

Diagram di atas menunjukan bahwa *heartbeat* bekerja sebagai pemeriksa kesehatan pada kedua *server*, dimana *heartbeat* yang berada pada *server bravo* atau *server backup* akan selalu memonitor keadaan dari *server alpha* atau *server* utama. Agar tidak bingung disini penulis akan mendeklarasikan *IP* untuk *server alpha* atau *server* utama dengan *IP* 195.168.50.22 dan untuk *IP server bravo* atau *server backup* adalah 195.168.50.23 sedangkan untuk *IP virtual* disini penulis akan memberikan *IP* 195.168.50.24. *IP virtual* ini akan digunakan sebagai *IP cluster*, yang dimana pada saat terjadi *failover IP virtuallah* yang akan di pindahkan dari *server* utama ke server *backup.*

Untuk membangun sistem *failover* seperti ilustrasi diatas, maka harus konfigurasikan *network card* dan instal aplikasi *heartbeat* sebagai aplikasi yang digunakan untuk *failover*. Berikut ini langkah-langkah konfigurasi *failover* :

1. Konfigurasi *network* eth0 pada berkas /etc/network/interfaces.
   :~# nano /etc/network/interfaces
   Isi *file* tersebut dengan konfigurasi dibawah ini:
   auto eth0
   iface eth0 inet static
   address 195.168.50.22
   netmask 255.255.255.0
   gateway 195.168.50.1
   broadcast 195.168.50.254
   Restart service networking dengan cara :
   /etc/init.d/networking restart
   Konfigurasi juga *network* eth0 pada *server bravo* atau *server backup* dengan cara yang sama seperti di atas, berikan *IP address* sesuai dengan *IP address* yang sudah disepakati yaitu 195.168.50.23 untuk *server bravo*.

2. Instal aplikasi *heartbeat* pada masing-masing *server* dengan cara :
   :~# sudo apt-get install heartbeat.

3. Setelah *heartbeat* selesai di instal, maka langkah selanjutnya mengkonfigurasi *heartbeat*. Buatlah berkas baru yaitu ha.cf di /etc/ha.d/ha.cf pada masing-masig *server* dengan perintah berikut ini :
   :~# nano /etc/ha.d/ha.cf
   Isi dari berkas tersebut adalah :
   keepalive 2
   warntime 5
   deadtime 15
   initdead 90
   udpport 694
   auto_failback on
   bcast eth0
   node alpha bravo

   Untuk **node alpha bravo** sesuaikan dengan *hostname* masing-masing, disini penulis memakai *alpha* sebagai *primary* dan *bravo* sebagai *secondary*, penulisan tidak boleh terbalik. Lalu simpan berkas tersebut dengan cara menekan ^o(Ctrl+o), untuk keluar dari editor nano tekan ^x (Ctrl+o).

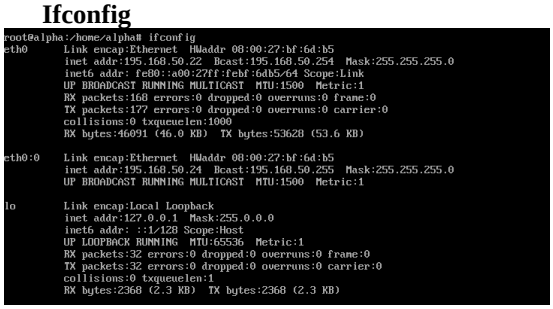
4. Buatlah berkas *authkeys* pada /etc/ha.d/authkeys dengan perintah :
   :~# nano /etc/ha.d/authkeys
   Isi dari berkas tersebut adalah :
   auth 1
   1 sha1 testing01
   Ganti *permision authkeys* dengan cara ketik:
   chmod 0600 /etc/ha.d/authkeys.

5. Buatlah berkas *haresources* pada /etc/ha.d/haresources. Isi dari *file* tersebut dengan :
   Alpha IPaddr::195.168.50.24/24/eth0 apache2
   Script diatas untuk membuat *IP virtual* yang akan digunakan untuk *failover* dimana sebagai *server primary* yaitu *alpha* sedangkan *IP* virtualnya adalah 195.168.50.24. Penulisan berkas diatas bersifat *case sensitive*, perhatikan besar dan kecil hurufnya.

6. Selanjutnya agar tidak repot dan setting kembali pada *server bravo*, maka copykan ketiga berkas tersebut pada *server bravo* dengan cara berikut :
   alpha:~# scp /etc/ha.d/ha.cf 195.168.50.22: /etc/ha.d/ha.cf
   alpha:~# scp /etc/ha.d/authkeys 195.168.50.22: /etc/ha.d/authkeys
   alpha:~# scp /etc/ha.d/haresources 195.168.50.22: /etc/ha.d/haresources.

7. Selanjutnya ketikan perintah ini di kedua *server alpha* dan *bravo* :
   chkconfig heartbeat on
   chkconfig apache2 off
   service apache2 stop
   service heartbeat start
   service apache start

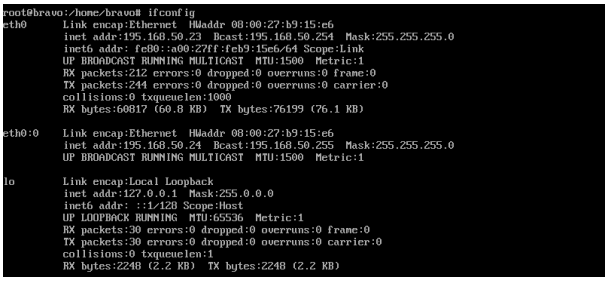
8. Lakukan *testing* pada *server alpha*, ketikan:
   Ifconfig



   Jika eth0:0 sudah ada, menandakan bahwa *IP* settingan sudah benar dan *IP virtual* sudah terbentuk.

9. Selanjutnya lakukan *test failover*, ketikan perintah berikut pada server alpha, jika settingan benar maka eth0:0 akan berpindah ke *server alpha*:
   service heartbeat stop

10. Lalu selanjutnya cek pada *server bravo* dengan cara ketik :
    Ifconfig



   Jika eth0:0 sudah ada, maka menandakan bahwa *IP* settingan sudah benar dan *IP virtual* sudah terbentuk.

11. Terakhir *setting* dari komputer *client* dengan mengakses ke *IP virtual*
    ping 195.168.50.24

12. Selanjutnya *test* dengan mematikan *server alpha.*

13. Jika status *ping* masih *replay* dan kondisi *server alpha* sudah dalam keadaan mati, maka sudah dinyatakan berhasil.
    Sampai disini membuat *server* dapat berpindah otomatis ketika terjadi disaster pada *server* utama telah selesai, namun disini penulis selain membuat *server* dapat melakukan *failover*, data antar kedua *server* juga harus tersinkronisasi. Cara untuk melakukan mensikrinisasikan data antara *server alpha* dan *bravo* adalah sebagai berikut :

4.6 Instalasi dan Konfigurasi DRBD

Pada penjelasan diatas sudah berhasil membuat dua buah *server* dapat melakukan *failover* ketika *server* utama *down*. Namun disini *failover* saja tidak cukup untuk membuat sebuah sistem yang mumpuni, kesamaan data antar dua *server* juga menjadi perhatian penting, akan sangat disayangkan apabila terjadi disaster *server* dapat melakukan *failover* namun data antar kedua *server* tidak sinkron.

Maka dari itulah diperlukannya sebuah teknologi yang dapat mereplikasi data server. Dalam penulisan ini akan dijelaskan bagaimana instalasi dan konfigurasi *DRBD* sebagai teknologi replikasi data.

Untuk itulah diperlukan sebuah teknologi yang dapat mereplikasi antar *server*. Dalam penulisan ini akan dijelaskan bagaimana cara instalasi dan konfigurasi *DRBD* sebagai teknologi replikasi data. Aplikasi *DRBD* sendiri digunakan sebagai perantara antar *server* untuk mereplikasi data. Berikut adalah langkah-langkah instalasi dan konfigurasi *DRBD* seperti dibawah ini :

1. Tambah satu buah *disk* pada kedua *server* dengan ukuran yang sama besar, sebagai contoh penulis akan menambahkan 2Gb pada *server alpha* dan *bravo*. *Disk* ini digunakan sebagai *disk* replikasi. Setelah ditambahkan lalu cek kondisi *disk* dengan perintah berikut :
   fdisk -1

hasil dari command tersebut:
Disk /dev/sda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes /512 bytes
I/O size (minimum/optimal): 512 bytes /512 bytes
Disk identifer: 0x0008760b
Device Boot Start End Blocks Id System
/dev/sda1 * 2048 18874367 9436160 83 Linux
/dev/sda2 18876414 20969471 1046529 5 Extended
/dev/sda5 18876416 20969471 1046528 82 Linux swap / Solaris
Disk /dev/sdb: 2147 MB, 2147483648 bytes
255 heads, 63 sectors/track, 261 cyliders, total 4194304 sectors
Unit = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes /512 bytes
I/O size (minimum/optimal): 512 bytes /512 bytes
Disk identifier: 0x00000000
Disk .dev.sdb doesn't contain a valid partition table
Pada hasil fdisk -1 dijelaskan bahwa disk /dev/sdb tersebut belum memiliki partisi, di karenakan belum melakukan format dan partisi.

2. Lalu instal paket *DRBD* pada masing-masing *server* dengan perintah:
   sudo apt-get install drbd8-utils drbdlinks

3. Lalu edit pada masing-masing *server file* drb.conf di **/etc/drbd.conf**. Lalu isikan dengan perintah berikut:

```
global {
dialog-refresh 1;
usage-count yes;
minor-count 5;
}
common {
syncer {
rate 10M;
}
}
resource r0 {
protocol C;
disk {
on-io-error detach;
}
syncer {
rate 10M;
al-extents 257;
}
on alpha {
device /dev/drbd0;
address 195.168.50.22:7788;
meta-disk internal;
disk /dev/sdb;
}
on bravo {
device /dev/drbd0;
address 195.168.50.23:7788;
meta-disk internal;
disk /dev/sdb;
}
}
```

4. Berikutnya buat *Meta* data *disk* dan jalankan *service DRBD*, jalankan perintah ini pada kedua *server* :
   drbdadm create-md r0
   service drbd start

5. Lalu cek status *DRBD* dengan perintah :
   service drbd status
   Namun kedua server sama-sama bertindak sebagai secondary dan keduanya *inconsistent* (masing-masing node berbeda isi). Itu tidak masalh, yang harus dilakukan adalah membuat *server alpha* menjadi *primary*.

6. Selanjutnya menjadikan *server alpha* sebagai *server primary*, lakukan perintah ini hanya pada *server alpha* saja:
   drbdsetup /dev/drbd0 primary –overwrite-data-of-peer
7. Lalu cek kembali *DRBD* status ketik :
   service drbd status
   Proses sinkronisasi ini ditunjukan dengan persen. Lamanya melakukan sinkronisasi tergantung besar data yang harus di sinkronisasi, tunggu proses hingga 100%, untuk melihat proses sinkronisasi itu sendiri ketik :
   watch service drbd status

8. Jika proses sinkronisasi sudah selesai, selanjutnya buatlah *file* sistem baru untuk *DRBD*, hal ini dilakukan hanya pada server alpha saja karena *server bravo* akan sinkronisasi dengan *server alpha*, ketikkan:
   mkfs.ext3 /dev/drbd0
   Sampai proses ini, instalasi dan konfigurasi *DRBD* sudah selesai. Selanjutnya untuk mengecek dan memastikan bahwa konfigurasi di atas sudah benar, disini penulis akan melakukan *testing DRBD.*

4.7 Melakukan Test DRBD

Setelah melakukan konfigurasi *DRBD* saatnya melakukan test hasil dari konfigurasi yang sudah dibuat, lakukan perintah-perintah berikut untuk melakukan testing Replikasi *DRBD* :

1. *Mounting* satu untuk *device DRBD* pada *server alpha*, buatlah satua buah *file* pada *device DRBD* yang sudah di *mounting*, ketikkan perintah berikut ini :
   mkdir /srv/data-drbd
   mount –t ext3 /dev/drbd0 /srv/data-drbd
   touch /srv/data-drbd/file_test_alpha.txt
   ls /srv/data-drbd/
   Hasil *command* di atas adalah terbentuknya satu buah *file* pada *service DRBD* yang sudah di *mount*.

2. *Unmount device DRBD* pada *server alpha* dan *downgrade* statusnya menjadi *secondary*, dengan ketikkan perintah berikut :
   umount /srv/data-drbd
   drbdadm secondary r0

3. Lalu selanjutnya pada *server bravo* aktifkan *DRBD server bravo* sebagai *primary*, dan *mounting service DRBD* dengan ketik perintah berikut :
   drbdadm primary r0
   service drbd status
   mkdir /srv/data/-drbd
   mount –t ext3 /dev/drbd0 /srv/data-drbd
   ls /srv/data-drbd/
   Server bravo akan otomatis memiliki salinan *file* yang sudah dibuat tadi pada *server alpha*. Jika tidak ada kesalahan pada konfigurasi seharusnya pada *server bravo* akan mendapatkan hasil *file* yang dibentuk pada *server alpha.*

4. Jika *testing* sudah selesai, selanjutnya kembalikan kembali ke kondisi semula yang dimana *server alpha* sebagai *primary* dan *server bravo* sebagai *secondary*.

5. Ketikkan perintah berikut pada *server bravo* :
   umount /srv/data-drbd
   drbdadm secondary r0

6. Ketikkan perintah berikut pada *server alpha* :
   drbdadm primary r0
   service drbd status
   mount –t ext3 /dev/drbd0 /srv/data-drbd

4. Implementasi Heartbeat dan DRBD

Pada bagian diatas sudah mengenal bagaimana menginstal dan konfigurasi *failover* dan replikasi *database*, lalu berikutnya bagaimana mengimplementasikan pada sebuah *service*. Disini penulis akan mengimplementasikan kedua teknologi tersebut untuk membangun sebuah *web server* yang mampu *recovery* dengan cepat dan dengan data yang sama.

1. *Install Apache* sebagai *web server*, ketikkan perintah berikut :
   sudo apt-get install apache2

2. Lalu *install PHP* dengan perintah berikut :
   sudo apt-get install php php-mysql

3. *Install Mysql* sebagai *server database*, ketikkan perintah berikut :
   sudo apt-get install mysql mysql-server

4. *Uninstall apparmor*, hal ini dilakukan agar pembuatan simbol *link* tidak *error*, ketikkan perintah berikut :
   dpkg –purge apparmor apparmor-utils

5. Lalu pindahkan *directory* /var/www ke *device DRBD* yang sudah di *mount* ke /srv/data-drbd. Sebelum dipindahkan pastikan *service apache* di *stop* terlebih dahulu, ketikkan perintah berikut :
   service apache2 stop
   mv /var/www /srv/data-drbd

6. Selanjutnya buat simbol *link* untuk *service apache*, lalu *start service apache* dengan ketikkan perintah berikut :
   ln –s /srv/data-drbd/www /var/www
   service apache2 start

7. Lalu pindahkan *directory* /var/lib/mysql ke device *DRBD* yang sudah di *mount* ke /srv/data-drbd. Sebelum dipindahkan pastikan *service apache* sudah di *stop* terlebih dahulu, ketikkan perintah berikut :
   service mysql stop
   mv /var/lib/mysql /srv/data-drbd

8. Berikutnya buat simbol *link* untuk *service mysql*, lalu *start service mysql*, ketikkan perintah berikut :
   ln –s /srv/data/-drbd/mysql /var/lib/
   service mysql start

9. Data-data yang digunakan untuk *server webserver* sudah dipindahkan ke *device DRBD*, sehingga apabila ingin mengembangkan sebuah *website* di letakkan pada *directory* yang ada pada *device DRBD*.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan uraian dan pembahasan, analisis dan pengujian yang telah dilakukan, maka dapat di ambil kesimpulan terhadap sistem Server Failover pada Ubuntu Server . Evaluasi yang telah dilakukan mengenai *Server Failover* setelah seluruh tahap penelitian dilalui serta mengacu pada rumusan masalah yang telah dipaparkan pada BAB I, maka

kesimpulan yang dapat diambil dari penelitian ini adalah :

1. Dengan menggunakan teknik *failover* pada sistem jaringan koneksi jaringan komputer di SMK Periwatas Tasikmalaya, apabila pada *server* utama terjadi kerusakan, maka *server* cadangan akan menggantikan peranan dari *server* utama untuk menyampaikan paket data.
2. Dengan menggunakan konfigurasi *failover*, yang dikontrol oleh *heartbeat*, pertukaran data dari *server* utama ke *server* cadangan lebih cepat dan jika terdapat gangguan salah satu koneksi jaringan internet terputus dapat di*backup* oleh koneksi yang lain.
3. *Heartbeat* dapat digunakan untuk *failover* sistem menggunakan *IP* pada *node* yang sudah dikonfigurasikan. Jika *node* yang digunakan dua menggunakan versi satu sedangkan bila lebih dari dua maka *heartbeat* versi dua yang digunakan.

5.2 Saran

Berdasarkan kesimpulan dan pembahasan yang telah diuraikan, maka saran yang dapat diberikan antara lain :

1. Pada pengembangan selanjutnya, sistem diharapkan dapat membuat recursive yang dimana bila kesalahan berasal dari pusat bukan dari lokal maka failoverpun akan tetap berjalan.
2. Seiring dengan kemajuan ilmu pengetahuan dan teknologi, maka tidak menutup kemungkinan membangun basis data menggunakan metode replikasi basis data dapat dikembangkan lagi dengan fasilitas-fasilitas yang lebih mumpuni.

DAFTAR PUSTAKA

[1] W. A. Yuliono, *Sinergi Replikasi Server dan Sistem Failover pada Database Server untuk Mereduksi Downtime Disaster Recovery Planing (DRP)*, Vol. 03, No. 5, 2021.

[2] A. I. Harsapranata, *Implementasi Failover Menggunakan Jaringan Vpn Dan Metronet Pada Astridogroup Indonesia,* Vol. 08, No. 2, 2015.

[3] T. Dwi Prayogo, *Sistem Monitoring Jaringan Pada Server Linux Dengan Menggunakan Sms Gateway*, Vol. 02, No. 3, 2010.

[4] A. Rahmatulloh and F. MSN, "Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi," *Jurnal Teknologi dan Sistem Informasi*, Vol. 3, No. 2, pp. 241-248, 2017.

[5] A. Heryanto, *Backup Database Dengan Multi Master Replikasi Pada Kluster Server,* Vol. 06, No. 1, 2020.

[6] M. S. Sungkar, *Konfigurasi Jaringan Local Area Network(Lan) Dan Gateway Internet Menggunakan Router Cisco 2600 Series Di Arg Media Data Brebes*, Vol. 08, No. 2, 2019.

[7] Y. Efendi, *implementasi cluster server berbasis linux sebagai learning management system (lms) di smk muhammadiyah 2 pekanbaru,* vol. 02, no. 2, 2019.

[8] H. Maulana, *analisis dan perancangan sistem replikasi database mysql dengan menggunakan vmware pada sistem operasi open source,* vol. 01, no. 1, 2016.

[9] G. Kustriono, *Pendayagunaan Komputer Lama/Bekas di Sekolah Sekolah dengan Mengimplementasi Linux Terminal Server Project,* Vol. 04, No. 2, 2012.

[10] M. Hidayat darmawan, *fakultas teknik universitas halu oleo,* vol. 04, no. 2, 2018.