IMPLEMENTASI *HIGH AVAILABILITY WEB SERVER* MENGGUNAKAN METODE *WEB CLUSTER* DAN *HIGH AVAILABILITY PROXY* PADA *SINGLE BOARD COMPUTER* (STUDI KASUS: UD SENDANG PUTRA)

SKRIPSI

Untuk memenuhi Sebagian persyaratan
Memperoleh gelar Sarjana Komputer

Disusun oleh:
Roisul Setiawan
NIM: 165150701111006

PROGRAM STUDI TEKNOLOGI INFORMASI
JURUSAN SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020

PENGESAHAN

IMPLEMENTASI *HIGH AVAILABILITY WEB SERVER* MENGGUNAKAN METODE *WEB CLUSTER* DAN *HIGH AVAILABILITY PROXY* PADA *SINGLE BOARD COMPUTER*
(STUDI KASUS: UD SENDANG PUTRA))

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Roisul Setiawan
NIM: 165150701111006

Skripsi ini telah diuji dan dinyatakan lulus pada
16 Juli 2020
Telah diperiksa dan disetujui oleh:

Pembimbing I

Bayu Rahayudi, S.T., M.T.
NIP: 197407122006041001

Pembimbing II

Dany Primanita Kartikasari, S.T., M.Kom.
NIP: 197711162005012003

Mengetahui
Ketua Jurusan Sistem Informasi

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
FAKULTAS ILMU KOMPUTER
JURUSAN SISTEM INFORMASI

Dr. Eng. Herman Tolle, S.T., M.T.
NIP: 197408232000121001

PERNYATAAN ORISINILITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain dalam kegiatan akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam skripsi ini terbukti terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan, serta diproses sesuai dengan peraturan perun dang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 20 Februari 2020

Penulis
Roisulsetiawan@student.ub.ac.id

ABSTRAK

Roisul Setiawan, Implementasi *High Availability Web Server* Menggunakan Metode *Web Cluster* Dan *High Availability Proxy Pada Single Board Computer* (Studi Kasus: UD Sendang Putra)

Pembimbing: Bayu Rahayudi, S.T., M.T. dan Dany Primanita Kartikasari, S.T., M.Kom.

UD Sendang Putra merupakan usaha dagang yang bergerak dibidang distribusi, penjualan dan jasa bahan pangan. Sejak usaha dagang ini berdiri, semua kegiatan transaksi dan administrasi dilakukan secara manual. Dalam perkembangannya UD Sendang Putra sudah mulai bermigrasi ke sistem informasi, tetapi dalam pengembangannya UD Sendang Putra kesulitan mengimplementasikan sistem tersebut ke *server* dikarenakan lingkungan kerja yang tidak cocok untuk dipasang *server*. untuk mengatasi masalah tersebut maka dibuatlah sebuah *cluster server* dari *single board computer*. Untuk menambah *availability* nya digunakan 4 buah *single board computer* yang didesain dengan metode sistem terdistribusi agar *server* bisa melayani permintaan dengan baik. Dalam pengembangannya *server* ini dibangun menggunakan Raspberry Zero W untuk *load balancer* dan *server*, yang didesain untuk bisa melayani 3 kebutuhan fungsional dan 5 kebutuhan non fungsional. Perangkat lunak yang digunakan dalam pengembangan sistem ini yaitu Keepalived dan Haproxy pada *load balancer*. Nginx PHP dan MariaDB pada sisi *server*. Perangkat lunak tersebut dipakai sesuai dengan studi pustaka yang telah dilakukan. Pengujian fungsional dan non fungsional pada sistem menghasilkan nilai valid terhadap semua kebutuhan. Pengujian *performance* menyatakan bahwa *cluster server* ideal digunakan oleh 110 pengguna karena masih dalam batas aman penggunaan sumberdaya yakni 85%. Berdasarkan hasil pengujian yang dilakukan dapat diambil kesimpulan bahwa *cluster server* layak digunakan di UD Sendang Putra.

Kata Kunci: *server*, *single board computer*, *availability*

ABSTRACT

Roisul Setiawan, Implementation of High Availability Web Server Using Web Cluster Method and High Availability Proxy on Single Board Computer (Case Study: UD Sendang Putra)

Mentors: Bayu Rahayudi, S.T., M.T. dan Dany Primanita Kartikasari, S.T., M.Kom.

UD Sendang Putra is a business engaged in the distribution, sale and service of foodstuffs. Since this business was established, all transaction and administrative activities were carried out manually. In its development UD Sendang Putra has begun to migrate to the information system, but in its development UD Sendang Putra has difficulty implementing the system to the server because the work environment is not suitable for server installations. From this problem a cluster server is created from a single board computer. To increase availability, 4 single board computers are designed using a distributed system method so that the server can serve requests properly. In its development this server was built using 2 Raspberry Zero W for load balancers and servers, which are designed to be able to serve 3 functional requirements and 5 non functional requirements. The software used in the development of this system is Keepalived and Haproxy on load balancers. Nginx PHP and MariaDB on the server side. The software is used in accordance with the literature study that has been done. Functional and non functional testing of the system produces valid values for all requirements. Performance testing stated that the cluster server is ideally used by 110 users because it is still within the safe limit of resource use of 85%. Based on the results of tests conducted, it can be concluded that the server is suitable for use at UD Sendang Putra.

***Keywords**: server, single board computer, availability*

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINILITAS iii
PRAKATA iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL ix
DAFTAR GAMBAR x
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan 2
1.4 Manfaat 2
1.5 Batasan Masalah 3
1.6 Sistematika Pembahasan 3
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Kajian Pustaka 5
2.2 Sistem Terdistribusi 6
2.3 *High Availability* Dan *Fault Tolerance* 7
2.4 Cluster Web Server 12
2.5 Single Board Computer 14
BAB 3 METODOLOGI 16
3.1 Identifikasi Masalah 16
3.2 Analisis Kebutuhan 17
3.3 Perancangan 17
3.4 Implementasi 18
3.5 Pengujian 18
3.6 Hasil 18
3.7 Kesimpulan 19
BAB 4 IDENTIFIKASI MASALAH DAN ANALISIS KEBUTUHAN 20

4.1 Identifikasi Masalah 20
4.2 Analisis Kebutuhan 22
4.2.1 Kebutuhan Perangkat Keras 23
4.2.2 Kebutuhan Fungsional 23
4.2.3 Kebutuhan Non Fungsional 24
BAB 5 Perancangan dan implementasi 25
5.1 Perancangan 25
5.1.1 Perancangan Arsitektur 25
5.1.2 Perancangan Topologi 26
5.1.3 Perancangan Pengujian 28
5.1.4 Perancangan Pengumpulan Data 29
5.2 Implementasi 31
5.2.1 Spesifikasi Perangkat Keras 31
5.2.2 Implementasi Arsitektur dan Topologi 32
5.2.3 Implementasi Perangkat Lunak 34
5.2.4 Implementasi *Failover* dan *Load balancing* 36
5.2.5 Implementasi Replikasi dengan Galera Cluster 38
BAB 6 Pengujian dan hasil 40
6.1 Pengujian 40
6.1.1 Hasil Pengujian Performance 40
6.1.2 Hasil Pengujian Fungsional 42
6.1.3 Menghitung Tingkat *Availability Cluster Server* 44
BAB 7 Penutup 46
7.1 Kesimpulan 46
7.2 Saran 46
DAFTAR REFERENSI 47

DAFTAR TABEL

Tabel 2.1 Perbandingan *Single Board Computer* 14
Tabel 4.1 Hasil Observasi Dan Wawancara 20
Tabel 4.2 Hasil Pengujian *Performance* 1 *Single Board Computer* 22
Tabel 4.3 Kebutuhan Perangkat Keras 23
Tabel 4.4 Kebutuhan Fungsional 24
Tabel 4.5 Kebutuhan Non Fungsional 24
Tabel 5.1 Perangkat Lunak 26
Tabel 5.2 Tabel Alamat IP Perangkat 27
Tabel 5.3 Jenis Pengujian 28
Tabel 5.4 *Performance Testing* 28
Tabel 5.5 *Blacbox Testing* 29
Tabel 5.6 Pengumpulan Data *Performance Testing* 30
Tabel 5.7 Pengumpulan Data *Black Box Testing* 30
Tabel 5.8 Pengumpulan Data Availabilitas Sistem 31
Tabel 5.9 Spesifikasi Perangkat Keras 31
Tabel 5.10 Konfigurasi Alamat IP *Main Load Balancer* 32
Tabel 5.11 Konfigurasi Alamat IP *Slave Load Balancer* 32
Tabel 5.12 Konfigurasi Alamat IP *Server* 1 33
Tabel 5.13 Konfigurasi Alamat IP *Server* 2 34
Tabel 5.14 Konfigurasi Keepalived *Main Load Balancer* 36
Tabel 5.15 Konfigurasi Keepalived *Slave Load Balancer* 37
Tabel 5.16 Konfigurasi Haproxy *Main Load Balancer* 37
Tabel 5.17 Konfigurasi Haproxy *Slave Load Balancer* 38
Tabel 5.18 Konfigurasi *Galera Cluster Server* 1 39
Tabel 5.19 Konfigurasi *Galera Cluster Server* 2 39
Tabel 6.1 Hasil Pengujian *Performance* Cluster Server 40
Tabel 6.2 Hasil Pengujian *Performance* Non Cluster 41
Tabel 6.3 Hasil Pengujian Fungsional 42
Tabel 6.4 Hasil Pengujian Non Fungsional 43
Tabel 6.5 Data Perhitungan *High Availability* 44

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi Load Balancer 7
Gambar 2.2 Replikasi *Synchronous* 11
Gambar 2.3 Replikasi *Asynchronous* 11
Gambar 2.4 Ilustrasi *Web Server* 13
Gambar 3.1 Diagram Alir Metodologi 16
Gambar 5.1 Rancangan Arsitektur 25
Gambar 5.2 Rancangan Topologi 27
Gambar 5.3 Hasil Konfigurasi Alamat IP *Main Load Balancer* 32
Gambar 5.4 Hasil Konfigurasi Alamat IP *Slave Load Balancer* 33
Gambar 5.5 Hasil Konfigurasi Alamat IP *Server* 1 33
Gambar 5.6 Hasil Konfigurasi Alamat IP *Server* 2 34
Gambar 5.7 Haproxy *Main Load Balancer* 34
Gambar 5.8 Keepalived *Main Load Balancer* 35
Gambar 5.9 Haproxy *Slave Load Balancer* 35
Gambar 5.10 Keepalived *Slave Load Balancer* 35
Gambar 5.11 Implementasi Nginx 36
Gambar 5.12 Implementasi MariaDB 36
Gambar 5.13 Implementasi PHP 36
Gambar 6.1 Grafik CPU dan Memory Usage 41

BAB 1 PENDAHULUAN

1.1 Latar Belakang

UD Sendang Putra adalah Usaha dagang perseorangan yang terletak di Desa Pelemwatu, Kecamatan Menganti, Kabupaten Gresik. UD Sendang Putra bergerak dalam bidang perdagangan dan distribusi beberapa komoditas pangan seperti gabah, beras, ketan jagung dan lain lain. Selain distribusi komoditi pangan, UD Sendang Putra juga bergerak dalam bidang pelayanan dan jasa yakni penggilingan bahan baku mentah. Dalam kegiatannya sehari hari UD Sendang Putra memiliki 30 karyawan yang dibagi kedalam beberapa bagian seperti bagian pengeringan, produksi, pemasaran dan administrasi.

UD Sendang Putra berdiri diatas tanah seluas kurang lebih 1 hektar yang dibedakan menjadi 2 bagian yakni bagian *indoor* dan *outdoor*. Bagian indoor memiliki 2 gedung utama yang dipisahkan oleh tempat pengeringan, bagian indoor digunakan untuk melayani jasa penggilingan, jual beli, produksi, penyimpanan dan administrasi. Sedangkan bagian outdoor digunakan sebagai tempat pengeringan bahan.

Dalam perjalanan usahanya UD Sendang Putra juga selalu mengikuti zaman, berawal dari mesin penggilingan konvensional yang menggunakan *diesel* dan solar sebagai bahan bakar kemudian bertransformasi menggunakan penggilingan listrik yang lebih ramah lingkungan. Begitu juga dalam hal administratif seperti jual beli, penyimpanan dan pencatatan sehari hari dan pengelolaan barang yang keluar masuk, UD Sendang putra juga menginginkan perubahan, hal ini disebabkan karena pengelolaan sekarang yang masih manual menggunakan salinan nota dan dokumen kertas lainnya yang dirasa sudah tidak efektif dan efisien, seringnya kehilangan nota nota perdagangan dan juga ketidaksesuaian barang dan nota juga menjadi penyebab utamanya. Sistem Informasi pengelolaan barang dan gudang pun menjadi solusi atas permasalahan permasalahan administratif yang terjadi selama ini terjadi di UD Sendang Putra.

Saat Sistem Informasi Pengelolaan barang dan gudang tersebut akan diimplementasikan muncul masalah lain yang tidak kalah penting yakni terkait infrastruktur pendukung sistem informasi yang akan diimplementasikan. UD Sendang Putra membutuhkan suatu infrastruktur yang dapat melayani kebutuhan akan sistem informasi secara cepat dan juga selalu ada dan siap digunakan saat jam kerja termasuk jika ada kegagalan yang tidak diinginkan. Lingkungan kerja UD Sendang Putra yang panas, sesak dan berdebu tidak memungkinkan adanya instalasi perangkat *server* normal atau hanya sekedar komputer *desktop* biasa.

Semua kegiatan *entry* data di UD Sendang Putra dilakukan menggunakan *smartphone* dan tablet dari karyawan masing-masing dan di tempat kerja masing-masing. *Entry* data pertama dilakukan saat ada komoditi yang masuk, kemudian dilanjutkan pada bagian pengeringan/produksi, setelah produksi selesai maka

komoditi akan di *entry* di gudang tempat penyimpanan untuk menunggu komoditi yang akan dijual keluar.

Untuk mengatasi kondisi lingkungan kerja yang tidak sesuai dengan lingkungan ideal untuk perangkat komputer, maka penelitian ini berupaya untuk mencari alternatif penggunaan perangkat lain yang lebih cocok untuk digunakan di UD Sendang Putra. Dari studi pustaka yang telah dilakukan maka diputuskan menggunakan *single board computer* yakni Raspberry Pi untuk dijadikan sebagai *server* pada infrastruktur yang dikembangkan. Dipilihnya *single board computer* sebagai *server* adalah karena sifatnya yang praktis, *portable,* kemampuan dan dukungan sistem operasi serta aplikasi yang setara dengan komputer pada umumnya yang bisa digunakan sebagai *server*.

Dengan adanya transaksi yang berjalan setiap harinya tentu saja ketersediaan sistem informasi di *server* menjadi prioritas utama dalam mengembangkan sistem ini. Dengan semua kelebihan yang dimiliki oleh *single board computer* tentu saja ada resiko yang timbul dari penggunaannya. Dari percobaan yang dilakukan oleh penelitian ini, 1 perangkat *single board computer* jenis Raspberry Zero W bisa menampung hingga 50 *request* secara konkuren. Batasan ini dicapai dengan mempertimbangkan pengukuran utilisasi *server* yakni performa CPU, memori dan konektivitas respon dari *server*.

Dengan permasalahan diatas penelitian ini mengusulkan untuk memilih metode *web cluster* dan *high availability proxy*. Dengan diimplementasikannya metode *web cluster* dan *high availability proxy* diharapkan efisiensi perangkat bisa ditingkatkan dan mengurangi resiko yang timbul akibat penggunaan *single board computer*.

1.2 Rumusan Masalah

1. Bagaimana respon sistem saat terjadi peningkatan trafik *request entry* sistem?
2. Bagaimana persentase utilitas sistem menggunakan *load balancing*?
3. Bagaimana persentase tingkat availabilitas sistem menggunakan *load balancing*?

1.3 Tujuan

Mewujudkan infrastruktur yang dapat meningkatkan availabilitas sistem menuju sistem dengan *high availability* dengan menggunakan perangkat sumber daya terbatas (constrained device) *single board computer*

1.4 Manfaat

Mendapatkan sistem yang memiliki *availability* tinggi yang dapat digunakan pada lingkungan panas dan berdebu serta dapat melayani kebutuhan bisnis UD Sendang Putra.

1.5 Batasan Masalah

Dalam proposal skripsi ini memiliki beberapa batasan masalah seperti berikut:

1. trafik data yang digunakan hanya transaksi CRUD saja
2. tingkat availabilitas sistem yang diharapkan minimal 85 persen

1.6 Sistematika Pembahasan

Sistematika penulisan laporan bertujuan untuk memberikan uraian serta gambaran dari penulisan penelitian ini yang meliputi beberapa bab, sebagai berikut:

BAB 1 PENDAHULUAN

Bab ini berisi ulasan yang membahas mengenai dasar latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat, dan sistematika pembahasan dalam penelitian pengembangan *high availability web server* dengan sistem terdistribusi pada *single board computer* (studi kasus : UD Sendang Putra).

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi ulasan yang membahas mengenai kajian pustaka dan dasar teori yang berhubungan dengan permasalahan dalam penelitian ini. Kajian pustaka berisi penelitian-penelitian sebelumnya yang berhubungan dengan penelitian ini. Dasar teori meliputi teori yang berhubungan dengan sistem terdistribusi, konsep *high availability* dan *fault tolerance* dan *cluster web server*.

BAB 3 METODOLOGI

Bab ini berisi ulasan yang membahas mengenai metodologi serta langkah kerja yang berhubungan dengan penelitian yang dilakukan. Langkah kerja terdiri dari identifikasi masalah, analisis kebutuhan, perancangan, implementasi, pengujian, hasil dan kesimpulan.

BAB 4 IDENTIFIKASI MASALAH DAN ANALISIS KEBUTUHAN

Bab ini berisi ulasan yang membahas mengenai proses identifikasi masalah dan analisis kebutuhan yang berhubungan dengan penelitian yang dilakukan. Identifikasi masalah membahas mengenai hal yang berhubungan dengan UD Sendang Putra dan kegiatan dalam menjalankan proses bisnisnya. Analisis kebutuhan membahas hal hal yang dibutuhkan untuk mengatasi permasalahan yang terdapat pada proses identifikasi masalah.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini berisi ulasan yang membahas mengenai proses perancangan yang dilakukan sesuai dengan analisis kebutuhan. Proses ini meliputi perancangan arsitektur dan perancangan topologi. Proses implementasi dilakukan setelah proses perancangan selesai. Pada proses ini rancangan yang sudah dibuat akan diimplementasikan langsung terhadap stakeholder.

BAB 6 PENGUJIAN DAN HASIL

Bab ini berisi proses pengujian yang menentukan keberhasilan proses diatasnya. jika pada proses pengujian infrastruktur telah memenuhi kebutuhan yang sudah ditetapkan maka penelitian ini dikatakan berhasil. pada proses terakhir berisi ulasan dan hasil dari proses pengujian.

BAB 7 PENUTUP

Bab ini berisi ulasan kesimpulan yang didapatkan dari mulai proses pendahuluan sampai proses hasil. pada bab ini juga opini penulis berupa saran untuk penelitian selanjutnya dikemukakan.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian "Implementasi *High Availability Web Server* Menggunakan Metode *Web Cluster* Dan *High Availability Proxy* Pada *Single Board Computer* (Studi Kasus: Ud Sendang Putra)" berlandaskan pada penelitian yang telah dilaksanakan oleh peneliti sebelumnya. Penelitian pertama diambil dari jurnal yang ditulis oleh Dawood (2014). Penelitian ini mengangkat masalah mengenai perbandingan *web server* yang digunakan pada *single board computer*. Metode yang digunakan untuk menguji kelayakan *web server* pada raspberry ini adalah dengan menguji kapasitas maksimum *request* yang bisa ditangani oleh Raspberry pi dengan alat uji Jmeter. hasilnya Nginx berhasil mengungguli Apache dan lighttpd dikarenakan Nginx memang cocok digunakan pada *hardware* dengan single thread.

Penelitian selanjutnya diambil dari jurnal yang ditulis oleh Rahmatulloh (2017). penelitian ini berangkat dari permasalahan yang dialami peneliti terhadap *request* yang *overload*. Peneliti kemudian memutuskan menggunakan metode *load balancing* untuk membagi koneksi yang masuk ke sistem informasi akademiknya. Metode penelitian yang dilakukan pada penelitian ini adalah pendekatan model PPDIOO yang diciptakan oleh Cisco untuk membangun suatu infrastruktur jaringan. Penelitian ini menggunakan Haproxy sebagai *software load balancing.* Hasil dari penelitian ini adalah *server* mampu melayani 10.000 *request* tanpa ada *error request* sama sekali dan juga penerapan *clustering server* yang dapat meningkatkan *response time* dan jumlah *current connection* pada *server*.

Penelitian selanjutnya diambil dari penelitian yang ditulis oleh Data, et al., (2019). penelitian ini menjelaskan tentang bagaimana desain dari *high availability dynamic web server cluster*. pada penelitian ini peneliti membuat sebuah rancangan *cluster server* dengan terdiri dari 3 perangkat virtual yang saling terhubung. penelitian ini menggunakan 5 perangkat lunak utama untuk bisa membuat *cluster server* yakni Nginx, PHP, MariaDB, Keepalived dan GlusterFS. Kebutuhan fungsional ini hanya ada 2 yakni *server* mampu berjalan dan memberikan service meskipun salah satu perangkat mengalami kegagalan dan *server* mampu mempertahankan reliabilitas data meskipun salah satu perangkat mengalami kegagalan. *Cluster server* pada penelitian ini diuji menggunakan *blackbox* testing untuk melihat apakah implementasi sudah sesuai dengan kebutuhan. selain menguji kebutuhan fungsional penelitian ini juga menguji *performance* dari *cluster server* dengan membombardir *server* dengan 12000 total koneksi.

Penelitian terakhir diambil dari penelitian yang ditulis oleh Aryal (2017). Penelitian ini menjelaskan mengenai implementasi *cluster system* pada Raspberry pi. pada penelitian ini penulis menggunakan 4 Raspberry pi sebagai *cluster system* dengan terdiri atas 2 *web server* dan 2 *load balancer*. *Cluster system* ini hanya dapat digunakan pada proyek skala kecil dan tidak untuk produksi yang besar. sistem *high availability* pada *cluster system* ini dapat dengan mudah mengurangi

kesalahan dan kegagalan kerja sistem dan juga membuat sistem bekerja mendekati titik kerja yang optimal.

2.2 Sistem Terdistribusi

Menurut (Steen & Tanenbaum, 2016) Pengertian sederhana dari sistem terdistribusi adalah kumpulan dari beberapa *node* atau elemen yang bekerja sama untuk membentuk satu kesatuan yang terlihat sebagai satu elemen. *Node* yang terhubung saling berbagi kondisi atau *state* dan melakukan operasi secara konkuren. Jika salah satu *node* mati maka tidak akan menimbulkan efek terhadap semua sistem besar yang terkoneksi dikarenakan beban operasi tidak hanya dilakukan oleh satu mesin.

Sistem terdistribusi memiliki beberapa karakteristik diantaranya adalah:

1. *Resource Sharing*

   Dalam sistem terdistribusi *node* 1 dengan yang lain saling terhubung dalam suatu jaringan dan akan saling berbagi *resource* yang ditentukan.

2. *Transparent*

   Dalam penerapan sistem terdistribusi pengguna tidak akan menyadari adanya banyak *node* pada sistem. Pengguna hanya akan berhadapan dengan *frontend node.*

3. *Computation Speed Up*

   Sistem terdistribusi dapat meningkatkan kecepatan pemrosesan karena pekerjaan akan dipecah dan dibagikan ke banyak *node*.

4. *Concurrency of Components*

   Pekerjaan yang tersedia akan dilakukan dalam waktu yang bersamaan oleh banyak *node* sehingga pekerjaan bisa lebih cepat selesai

5. *Scalable*

   Seiring berkembangnya waktu sistem terdistribusi dapat dengan mudah mengikutinya dikarenakan skalabilitasnya yang tinggi tanpa harus mengganti semua *node*. *Node* bisa ditambahkan dan bekerja secara langsung

6. *Independent failures of components*

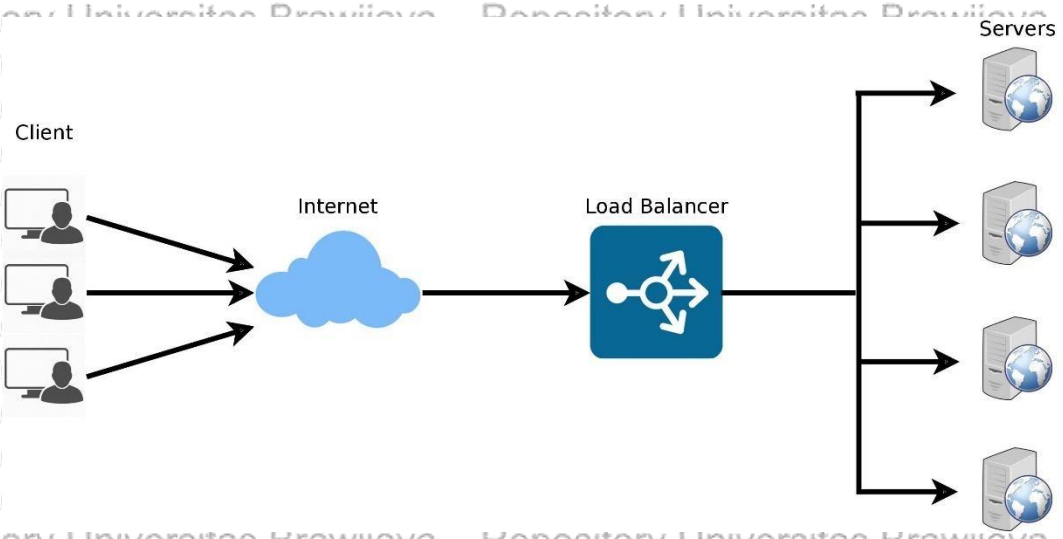
   Kegagalan dari satu *node* pada sistem tidak akan mempengaruhi keseluruhan sistem.

2.3 *High Availability* Dan *Fault Tolerance*

Dalam komputasi terkini *high availability* digunakan untuk mendeskripsikan periode waktu ketika suatu layanan tersedia serta waktu yang dibutuhkan suatu sistem untuk merespon permintaan yang dibuat oleh *user*. Menurut Heidi (2019) *High availability* adalah kualitas dari sistem atau komponen yang menjamin kinerja operasional untuk periode waktu tertentu. *High availability* sering dinyatakan sebagai persentase yang menunjukkan berapa banyak waktu aktif yang diharapkan dari sistem atau komponen tertentu dalam periode waktu tertentu, di mana nilai 100% akan menunjukkan bahwa sistem tidak pernah mengalami kegagalan. Untuk menghitung *availability* dari sebuah layanan, kita harus tau berapa lama layanan digunakan dan berapa lama layanan mengalami kegagalan. Untuk menghitung nilai *availability* didapatkan dengan menggunakan persamaan 2.1 berikut ini.

Availability = Uptime / (Uptime + Downtime) (2.1)

Sistem yang menerapkan mekanisme *high availability* adalah sistem yang dirancang untuk tersedia 99,999% sepanjang waktu atau sedekat mungkin dengannya. Biasanya ini berarti mengkonfigurasi sistem *failover* yang dapat menangani beban kerja yang sama dengan sistem utama. Salah satu teknik yang paling terkenal untuk mencapai sistem *high availability* adalah *load balancing*, *load balancing* adalah salah satu teknik untuk mendistribusikan beban *traffic* pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan *throughput,* memperkecil waktu *response* dan menghindari *overload* pada salah satu jalur koneksi. *Load balancing* digunakan pada saat sebuah *server* telah memiliki jumlah pengguna yang telah melebihi maksimal kapasitasnya. Mengacu pada gambar 2.1 *Load balancing* juga mendistribusikan beban kerja secara merata di dua atau lebih komputer untuk mendapatkan pemanfaatan sumber daya yang optimal.



Gambar 2.1 Ilustrasi Load Balancer

Sumber : (Anderson, 2017)

Seperti yang terlihat pada gambar 2.1 *load balancer* berada di antara *backend server* dan juga *client*. *load balancer* berfungsi untuk menerima dan mendistribusikan *traffic* yang masuk untuk mengurangi beban *server*. *load balancer* bisa berupa *hardware* khusus atau sebuah proses dari sebuah *software*. *load balancer* akan secara berkala melakukan *health check* pada *backend server* untuk menentukan *server* yang siap digunakan dan yang mengalami kegagalan. *load balancer* bekerja dengan memanfaatkan 2 protokol pada OSI *layer* yakni *layer* 4 dan *layer* 7. *Layer* 4 *load balancer* bekerja pada *transport* level, Itu berarti mereka dapat membuat keputusan routing berdasarkan pada port TCP atau UDP yang digunakan paket bersama dengan alamat IP sumber dan tujuan mereka. *layer* 4 *load balancer*s melakukan proses *network address translation* tetapi tidak memeriksa isi sebenarnya dari setiap paket. sedangkan *layer* 7 *load balancer* bekerja pada level aplikasi yakni level tertinggi dari OSI *layer*. karena bekerja pada level aplikasi *layer* 7 dapat mengevaluasi rentang data yang lebih luas daripada *layer* 4, termasuk *header* HTTP, *Session id* dan SSL, ketika memutuskan bagaimana mendistribusikan permintaan di seluruh *server*.

Menurut Anderson (2017) ada beberapa *traffic* yang bisa ditangani oleh *load balancer* yakni HTTP, HTTPS, TCP dan UDP kemudian yang tidak kalah penting adalah algoritma yang dipakai oleh *load balancer*. Terdapat 3 algoritma yang dapat dipakai untuk menentukan *backend* yakni:

1. *Round Robin*

   Round Robin berarti *load balancer* memilih *server* secara sekuensial, artinya *load balancer* akan memilih *server* satu untuk *request* pertama dan *server* dua untuk *request* kedua dan begitu seterusnya.
2. *Least Connection*

   Least Connection berarti *load balancer* akan memilih *server* yang mempunyai sedikit koneksi dan beban. Algoritma ini cocok dipakai untuk tipikal aplikasi yang memiliki *session* yang lama
3. *Source*

   Pada algoritma *source*, *load balancer* akan memilih *server* berdasarkan asal alamat IP klien. Algoritma ini memastikan *user* akan secara konsisten terhubung dengan *server* yang sama.

Contoh produk yang banyak digunakan untuk melakukan *load balancing* adalah Haproxy. Haproxy adalah sebuah *load balancer* berjenis perangkat lunak dan dapat diinstal pada sebagian besar sistem operasi Linux, Solaris dan Freebsd. Haproxy bisa berjalan pada *layer* 4 *load balancer* dan *layer* 7 *load balancer*.

Berbeda dengan *high availability*, Menurut (Nørv˚ag, 2000) *fault tolerance* adalah suatu sistem yang dapat menjalankan tugasnya dengan baik dan benar meskipun terdapat kegagalan yang terjadi. konsep *fault tolerance* berada pada 1 tingkat setelah *high availability*, jika tujuan *high availability* adalah ketersediaan sistem yang mendekati 99,9999% sedangkan pada *fault tolerance* adalah sistem

dapat menjalankan 0% *downtime*, artinya sistem harus bisa sama sekali tidak mengalami status down. *Fault tolerance* menjadi sangat penting di era ini karena kegagalan sekecil apapun akan mengakibatkan kerugian. Untuk mewujudkan sistem yang *fault tolerance*, sistem harus bisa mendeteksi kegagalan dan melakukan tindakan yang diperlukan dengan 2 dasar yakni:

1. *Mask the Fault*

   Teknik ini adalah teknik yang dapat menutupi kegagalan. Semua tugas harus bisa berjalan dan selesai tanpa kehilangan data atau informasi meskipun terjadi penurunan kinerja sistem.

2. *Fail Gracefully*

   Teknik ini adalah teknik untuk membuat suatu mekanisme antisipasi berdasarkan SOP tertentu untuk memungkinkan menghentikan proses dalam waktu yang singkat tanpa harus kehilangan data dan informasi.

Salah satu tujuan *high availability* adalah untuk menghilangkan kesalahan. Kesalahan yang terjadi pada suatu sistem biasanya diakibatkan oleh 4 *major factor* yakni kesalahan spesifikasi (*Specification Mistake*), kesalahan implementasi (*Implementation Mistake*), kerusakan komponen (*Component Defect*) dan gangguan luar (*External Disturbance*). Untuk menghilangkan *failure* bisa dilakukan dengan teknik redundansi. Redundansi berarti memiliki lebih banyak sumber daya yang sama pada sistem untuk menutupi atau mengatasi terjadinya kegagalan. Jenis redundansi yang biasanya dipakai ada 4 macam yakni

1. Redundansi *Hardware*

   Dilakukan dengan cara menambah perangkat keras tambahan kedalam sistem untuk mengatasi atau mengesampingkan kegagalan yang terjadi.

2. Redundansi *Software*

   Dilakukan dengan cara menambah versi dari *software* yang diharapkan untuk tidak mengalami gangguan pada input yang sama dengan *software* utama,

3. Redundansi Informasi

   Dilakukan dengan cara menambah bit tambahan (*bit check*) pada kode program untuk melakukan deteksi dan koreksi dalam memori dan penyimpanan lain untuk kesalahan ringan.

4. Redundansi Waktu

Dilakukan dengan cara mengulang pelaksanaan program yang sama pada *hardware* yang sama. Redundansi ini sangat efektif terhadap kegagalan yang sifatnya sementara.

Salah satu teknik yang digunakan untuk mencapai *fault tolerance* pada penelitian ini adalah replikasi. Replikasi adalah salah satu teknik redundansi *hardware* yang dilakukan pada *server* basis data. Replikasi basis data adalah suatu cara untuk menggandakan dan mendistribusikan basis data baik secara lengkap atau secara parsial di beberapa *node* yang secara fisik terpisah untuk mencegah terjadinya kegagalan dan kehilangan data (Bernstein & Goodman, 1985). Tujuan dari replikasi basis data adalah membuat suatu backup data dari sebuah basis data relasional. Replikasi basis data digunakan untuk menyalin dan mendistribusikan data dari satu basis data ke basis data yang lain. Selanjutnya secara otomatis dilakukan sinkronisasi antar basis data tersebut untuk menjaga konsistensi.

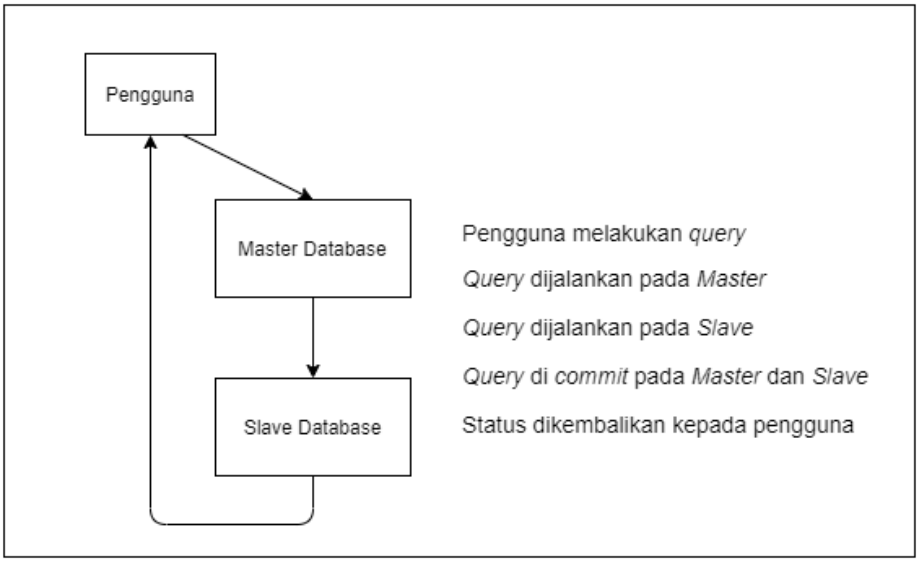
Ada beberapa model umum yang biasanya dilakukan untuk melakukan replikasi yakni

1. *One master, one slave*. yakni replikasi yang hanya memiliki 1 basis data utama dan 1 basis data cadangan
2. *One master, many slaves*. yakni replikasi yang memiliki 1 basis data utama dan lebih dari 1 basis data cadangan
3. *Master/slave circular relationship*. yakni replikasi multi *master* yang memiliki tugas untuk mengupdate status satu sama lain
4. *Master/slave "daisy chain"* yakni replikasi multi *master* yang memiliki lebih dari 2 basis data yang saling mengupdate status dengan berputar antara *node* 1 dengan *node* yang lain

Dalam melakukan transaksi, replikasi dibagi menjadi 2 jenis yakni

1. Replikasi *Synchronous*

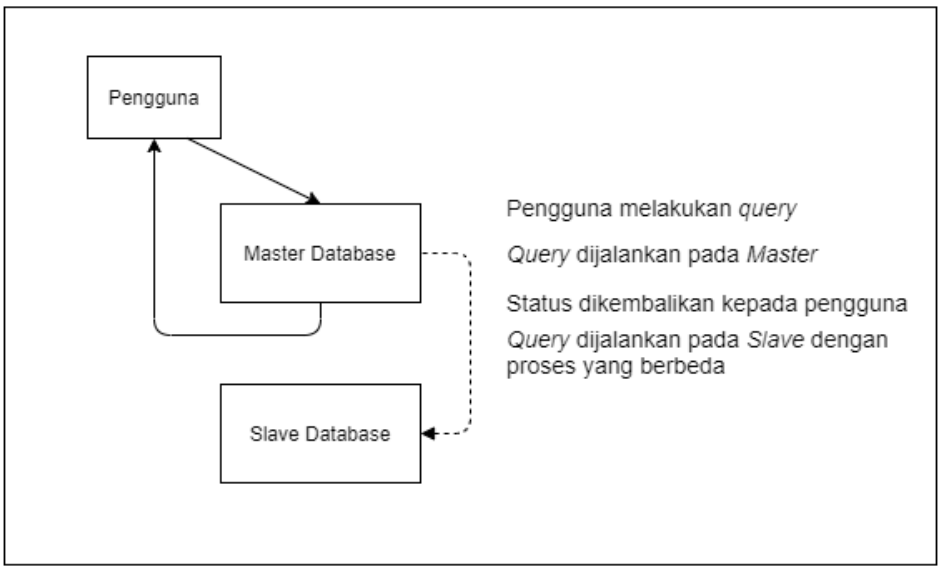
   Pada jenis ini replikasi dilakukan secara *real-time* antara *master* dan *slave*. Keseluruhan proses penulisan *master* dan *slave* harus selesai terlebih dahulu sebelum melanjutkan ke transaksi selanjutnya. Saat menjalankan replikasi ini kebutuhan performa harus dipertimbangkan. Jenis ini memiliki keuntungan yakni data yang selalu konsisten dan terjaga. Berikut ini adalah ilustrasi dari jenis replikasi synchronous yang ditunjukan pada gambar 2.2



Gambar 2.2 Replikasi *Synchronous*

2. Replikasi *Asynchronous*

Pada jenis ini replikasi dilakukan secara *buffer* antara *master* dan *slave*. Replikasi dilakukan secara periodik dalam jangka waktu tertentu antara *master* dan *slave*. Pada replikasi jenis ini keuntungannya adalah efektifitas biaya dari proses transaksi. Namun juga memiliki kekurangan yakni tidak menjamin kesinkronan apabila terdapat kegagalan pada saat replikasi belum selesai dilakukan. Berikut ini adalah ilustrasi dari jenis replikasi *asynchronous* yang terdapat pada gambar 2.3



Gambar 2.3 Replikasi *Asynchronous*

Berbeda dengan *synchronous*, pada jenis *asynchronous,* replikasi dilakukan pada proses yang terpisah sehingga keseluruhan eksekusi akan berjalan lebih cepat karena tidak menunggu data untuk direplikasi terlebih dahulu.

Contoh dari implementasi replikasi yang banyak digunakan saat ini adalah produk dari MariaDB yakni Galera Cluster. Galera Cluster merupakan plug-in replikasi multi *master* untuk *database* dengan *engine* innoDB. berbeda dengan replikasi milik MySQL yang hanya bisa digunakan untuk *master-slave* replikasi, Galera Cluster milik MariaDB bisa digunakan untuk menjalankan multi *master* replikasi.

Dalam pekerjaannya Galera Cluster sangat bergantung pada WSREP dan Rsync untuk membantu pekerjaannya. sebelum mengenal WSREP dan Rsync sebagai mekanisme transfer data pada basis data kita harus mengenal terlebih dahulu mekanisme *state transfer*. *State transfer* adalah teknik yang digunakan untuk mengirimkan data pada replikasi Galera Cluster. *state transfer* dibagi menjadi 2 yakni:

1. *State Snapshot Transfer* (SST)

   Dalam mekanisme ini semua salinan data akan dikirim menjadi 1 berkas kepada *node* lain. mekanisme ini biasanya digunakan saat ada *node* baru yang bergabung pada sebuah *cluster*

2. *Incremental State Transfer* (IST)

   Dalam mekanisme ini yang akan dikirim adalah salinan data yang tidak tersedia saja kepada *node* lain. mekanisme ini biasanya digunakan saat ada transaksi DML atau DDL yang dilakukan pada suatu *cluster* yang sudah memiliki salinan masing masing data.

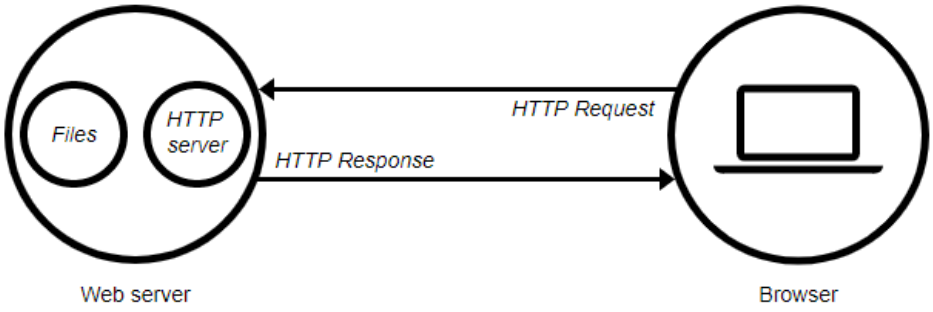
Disinilah peran dari Rsync dan WSREP, Rsync digunakan untuk transfer dengan model SST dan WSREP digunakan pada transfer jenis IST. dalam implementasinya Galera Cluster bisa dipakai pada 2 *node* atau lebih, tetapi disarankan untuk menggunakan *node* berjumlah ganjil agar data memiliki tingkat konsisten yang tinggi. contohnya ada 3 *node* yakni *node* X, *node* Y dan *Node* Z, disaat *node* X dan *node* y memiliki data yang berbeda maka *node* Z akan menjadi patokan untuk mempertahankan konsistensi data pada basis data.

2.4 Cluster Web Server

Dikutip dari Mozilla (2019) *Web server* bisa diartikan sebagai *software* atau *hardware* dan juga keduanya bisa bekerja sama untuk bisa disebut sebagai *web server.* pada definisi sebagai *hardware*, *web server* adalah suatu komputer yang menyimpan perangkat lunak *server web* dan *file* komponen situs *web* (misalnya dokumen HTML, gambar, *stylesheet* CSS, dan *file* JavaScript) dan Terhubung ke suatu jaringan lokal atau Internet yang mendukung pertukaran data fisik dengan perangkat lain yang terhubung ke *web*. Kemudian pada sisi perangkat lunak, *web server* mencakup beberapa bagian yang mengontrol cara pengguna *web* mengakses *file* yang di-*host*, minimal *server* HTTP. *Server* HTTP adalah perangkat

lunak yang memahami dan menggunakan URL (alamat *web*) dan HTTP (protokol yang digunakan browser Anda untuk melihat halaman *web*). yang dapat diakses melalui nama domain dari situs *web* yang disimpannya, dan mengirimkan kontennya ke perangkat pengguna akhir.

Mengacu pada gambar 2.4 Pada tingkat paling dasar, setiap kali *browser* membutuhkan *file* yang di-*host* di *web server, browser* meminta *file* melalui HTTP. Ketika permintaan mencapai *web server* (perangkat keras) yang benar, *server* HTTP (perangkat lunak) menerima permintaan, menemukan dokumen yang diminta (jika tidak maka respon 404 dikembalikan), dan mengirimkannya kembali ke *browser*, juga melalui HTTP.



Gambar 2.4 Ilustrasi *Web Server*

Sumber: (Mozilla, 2019)

Contoh produk yang sering digunakan untuk membuat suatu *web server* adalah Apache, Nginx, Lighttpd dan masih banyak lagi. produk produk *web server* tersebut tentu saja memiliki karakteristik dan kelebihan masing masing.

Sedikit berbeda dari *web server* biasa, *cluster web server* merupakan sekelompok *web server* yang saling terhubung dan bekerja sama untuk untuk meningkatkan kinerja dan ketersediaan suatu layanan. jadi *cluster web server* terdiri dari lebih dari satu perangkat dan bekerja sama untuk mencapai suatu tujuan tertentu, kelebihan dari *cluster web server* adalah mudahnya penambahan perangkat baru untuk scalling, meningkatkan availabilitas dan performa *server*, menghilangkan *single point of failure* dan juga tetap menjaga agar *server* tetap berjalan meskipun ada perangkat yang mengalami kegagalan. Jenis *cluster server* dapat dibedakan menjadi 3 yakni:

1. *Failover Clusters*

   Terdiri dari 2 atau lebih perangkat yang terhubung dalam suatu jaringan dengan koneksi heatbeat yang terpisah antara 2 *host*. Koneksi *heartbeat* antara 2 perangkat digunakan untuk memantau apakah semua layanan masih dapat digunakan. jika terdeteksi suatu layanan pada perangkat mengalami kegagalan maka perangkat lain akan segera mengambil alih layanan tersebut.

2. *Load-Balancing Clusters*

Terdiri dari lebih dari 2 perangkat dan juga *node* khusus yang bertugas sebagai penerima *request* dan mendistribusikan *request* ke *backend server*. Konsep ini mirip dengan konsep *Failover cluster* tetapi ada *node* tambahan yang bertugas untuk mendistribusikan trafik

3. *High Performance Computing Cluster*

Perangkat dikonfigurasikan secara khusus untuk memberikan layanan yang membutuhkan kinerja ekstrem. *Cluster* semacam ini juga memiliki beberapa fitur load-balancing. *cluster* ini mencoba untuk mendistribusikan proses yang berbeda ke lebih banyak perangkat untuk mendapatkan kinerja. Yang membedakannya adalah pada *cluster* ini semua layanan diproses secara paralel dan tidak perlu untuk satu layanan menunggu layanan lainnya.

2.5 Single Board Computer

Single board computer (SBC) adalah gabungan dari mikroprosesor, memori, input, output dan komponen lain yang dibutuhkan oleh suatu komputer untuk bekerja yang dibangun pada satu papan sirkuit (J.Johnston, et al., 2018). Untuk menjelaskan lebih lanjut mari kita lihat perbandingan pada tabel 2.1

Tabel 2.1 Perbandingan *Single Board Computer*

Component	PC	CB	Smartphone	SBC
CPU	DB	Yes	Yes	Yes
GPU	Yes, DB	Yes	Yes	Yes
Memory	DB	Yes	Yes	Yes
LAN	Yes, DB	N/A	N/A	Yes
Video Output	Yes, DB	N/A	N/A	Yes
Storage	ROM, RW-Ext	ROM	ROM, RW-Ext	ROM, RW-Ext
GPIO Header	Yes, (USB)	Yes	N/A	Yes

Sumber : (J.Johnston, et al., 2018)

Bisa dilihat pada perbandingan diatas antara *Personal Computer* (PC), *Controller Board* (CB), *Smartphone* dan *Single Board Computer* (SBC). *Single Board Computer* memiliki fitur dan komponen yang lengkap jika dibandingkan dengan perangkat lain yang serupa.

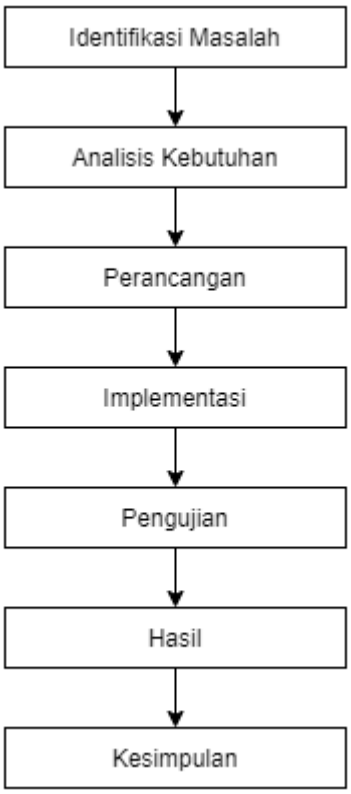
Dalam hal fungsionalitas dan penggunaannya *Single Board Computer* bisa digunakan di hampir semua bidang yang bisa dilakukan oleh *Personal Computer* lebih dari itu *Single Board Computer* juga bisa digunakan sebagai *server* untuk melayani aktivitas dengan skala kecil sampai menengah. Dikutip dalam laman resmi Raspberry, Raspberry Pi merupakan sebuah komputer yang berukuran sebesar kartu kredit yang dikeluarkan oleh Raspberry Foundation yang bisa

digunakan untuk menjalankan tugas tugas sederhana seperti menjelajah internet, menonton video dan bermain *game*. Raspberry Pi juga memiliki input standart seperti USB dan HDMI yang bisa digunakan untuk mengontrol sistem. Lebih hebatnya lagi Raspberry juga bisa berinteraksi dengan dunia luar dan juga sudah digunakan di berbagai proyek proyek besar sebagai alat pemroses data di *outdoor*.

Menurut Damasvara (2010) penggunaan *resource* yang paling optimal untuk sebuah perangkat adalah 10% - 70% dari kapasitas *resource* perangkat tersebut. Penggunaan antara 70% - 80% terbilang masih bisa digunakan dengan layak untuk sebuah perangkat. Apabila perangkat mencapai 90% - 100% maka perlu dilakukan penambahan *resource* baru agar perangkat dapat bekerja secara optimal.

BAB 3 METODOLOGI

Dalam bab ini dijelaskan mengenai metodologi yang akan digunakan dalam penelitian ini. Langkah-langkah tersebut meliputi identifikasi masalah, studi literatur, analisis kebutuhan, perancangan, implementasi, pengujian dan kesimpulan. Berikut ini merupakan diagram alir yang menjelaskan mengenai metodologi yang digunakan seperti pada Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi

3.1 Identifikasi Masalah

Identifikasi masalah merupakan suatu tahapan yang dilakukan untuk mengumpulkan topik permasalahan yang akan digunakan dalam penelitian. Identifikasi masalah didapatkan dari kegiatan kegiatan berikut :

1. Observasi

   Merupakan suatu metode pengumpulan data dengan melakukan pengamatan langsung terhadap masalah dan solusi yang dibutuhkan pada UD Sendang Putra. Tahapan observasi yang ditempuh yakni:

   1. Melakukan seleksi terhadap *setting* penelitian
   2. Mendefinisikan apa yang dapat didokumentasikan dalam observasi
   3. Menerapkan aturan-aturan yang harus ditaati dalam melakukan pengamatan sesuai fokus penelitian

4. memfokuskan observasi pada aspek-aspek yang relevan dengan pertanyaan penelitian
5. Mendokumentasikan berbagai macam hal yang dibutuhkan dalam penelitian
6. Mengakhiri observasi apabila tujuan observasi telah tercapai

2. Wawancara

Merupakan suatu metode pengumpulan informasi yang dilakukan dengan bertanya langsung terhadap stakeholder dan pengguna yang dianggap layak dalam memberikan informasi mengenai permasalahan. Ada beberapa langkah yang ditempuh untuk melakukan proses wawancara yakni:

1. Menetapkan kepada siapa wawancara akan dilakukan
2. Menyiapkan pokok-pokok masalah yang akan menjadi bahan pembicaraan
3. Mengawali atau membuka alur wawancara
4. Melangsungkan wawancara
5. Mengkonfirmasikan ikhtisar hasil wawancara dan mengakhirinya
6. Menuliskan hasil wawancara ke dalam dokumen hasil wawancara
7. Mengidentifikasi tindak lanjut hasil wawancara yang telah diperoleh

3.2 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk menganalisis dan mendapatkan semua kebutuhan yang diperlukan dalam penelitian ini. Tahapan ini dilakukan untuk mengidentifikasi semua kebutuhan dari permasalahan yang sudah ditemukan dalam proses identifikasi masalah. Langkah pertama yang dilakukan pada analisis kebutuhan adalah menentukan kebutuhan fungsional. Kebutuhan fungsional adalah kebutuhan yang berisi proses-proses apa saja / layanan apa saja yang nantinya harus disediakan oleh sistem, mencakup bagaimana sistem harus bereaksi pada input tertentu dan bagaimana perilaku sistem pada situasi tertentu. Kebutuhan fungsional didapatkan pada saat proses observasi dan wawancara yang telah dilakukan. Setelah kebutuhan fungsional didapatkan langkah selanjutnya adalah menentukan kebutuhan non fungsional. Kebutuhan non fungsional adalah kebutuhan yang menitikberatkan pada perilaku sistem dan bagaimana sistem dijalankan. Setelah semua kebutuhan fungsional dan non fungsional ditentukan maka tahapan terakhir adalah menganalisa kebutuhan perangkat keras yang digunakan seperti jumlah perangkat keras, merk, dan spesifikasi.

3.3 Perancangan

Perancangan merupakan tahapan yang digunakan sebagai acuan dalam melakukan implementasi dan pengujian dari sistem yang akan dibangun. Proses pertama dalam tahap ini adalah perancangan arsitektur dimana pada perancangan arsitektur penelitian ini akan menentukan bagaimana nantinya infrastruktur akan dibangun, jumlah klien yang dapat ditampung, jumlah perangkat keras yang dibutuhkan, spesifikasi dari perangkat lunak yang

dibutuhkan dan bagaimana alur data masuk dan keluar. Setelah mendapatkan desain arsitektur yang sesuai dengan analisis kebutuhan selanjutnya adalah perancangan topologi, perancangan ini bertujuan untuk memetakan dan menentukan alamat IP kepada semua perangkat baik *cluster server* maupun klien. Yang terakhir adalah perancangan pengujian sistem, perancangan ini berguna untuk menentukan bagaimana nantinya sistem akan diuji untuk menentukan kualitasnya. Dalam pengujian ini akan berisi pengujian performa untuk menentukan apakah performa yang dimiliki oleh sistem sudah sesuai dengan kebutuhan. Kemudian pengujian *black box* yang akan menentukan apakah semua kebutuhan fungsional dan nn fungsional terpenuhi. Dan yang terakhir adalah menghitung tingkat availabilitas selama sistem digunakan agar sesuai dengan identifikasi masalah

3.4 Implementasi

Implementasi merupakan tahapan penerjemahan proses analisis kebutuhan dan perancangan ke dalam sistem yang sebenarnya. Proses yang dilaksanakan dalam tahapan implementasi sistem adalah menentukan spesifikasi SBC kemudian mengimplementasikan arsitektur yakni memasang *single board computer* sesuai dengan perancangan arsitektur, implementasi topologi yakni memberikan alamat IP kepada semua perangkat sesuai dengan perancangan. Memasang *software* yang dibutuhkan untuk *load balancing*, *web server*, *server* basis data dan replikasi basis data. Melakukan konfigurasi terhadap semua *software* dan memastikan *software* dapat berjalan dengan baik.

3.5 Pengujian

Pengujian sistem pada penelitian ini dibagi menjadi beberapa metode pengujian. Metode pengujian pertama yakni pengujian performa, pengujian performa dilakukan dengan cara mengakses data pada jumlah tertentu secara bersamaan dengan beberapa *scenario* yang bersifat *incremental*. Setiap scenario akan diuji 10 kali dan diambil rata ratanya untuk memperoleh data yang konsisten. Pengujian kedua adalah pengujian *black box* yang dilakukan dengan beberapa *scenario* yakni mengaktifkan dan menonaktifkan beberapa layanan untuk menguji apakah sistem masih dapat melayani kebutuhan pengguna. Scenario yang dimaksud adalah dengan mematikan 1 sampai 2 perangkat untuk melihat bagaimana sistem merespon adanya kegagalan dan mengatasinya. Pengujian terakhir adalah dengan menghitung tingkat availabilitas sistem. Dalam pengujian ini dibutuhkan waktu beberapa bulan untuk mengumpulkan data. Data yang dikumpulkan adalah berapa lama waktu sistem aktif, berapa lama waktu sistem mati dan berapa lama maintenance akan dilakukan.

3.6 Hasil

Hasil dan analisis merupakan tahapan yang dilakukan setelah proses pengujian. Tujuan dari proses ini adalah melakukan verifikasi terhadap hasil dari pengujian dan juga menjelaskan hal hal yang diperoleh dari semua proses pengujian. Pada

tahap ini juga kita akan menentukan berhasil dan tidaknya semua proses penelitian.

3.7 Kesimpulan

Pengambilan kesimpulan dapat dilakukan setelah semua proses tahapan perancangan, implementasi, dan pengujian sistem telah selesai dilaksanakan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Tahapan akhir merupakan saran yang dimaksudkan dapat memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan untuk pengembangan sistem selanjutnya.

BAB 4 IDENTIFIKASI MASALAH DAN ANALISIS KEBUTUHAN

Dalam bab ini dilakukan dua tahap sekaligus yakni identifikasi masalah dan analisis kebutuhan. Langkah tersebut dilakukan secara teliti dan mendalam untuk menghasilkan perancangan yang baik dan sesuai dengan apa yang dibutuhkan.

4.1 Identifikasi Masalah

Identifikasi masalah dilakukan dengan observasi ke lapangan secara langsung dan juga melakukan wawancara kepada bagian elektronika UD Sendang Putra. Dalam hasil observasi dan wawancara didapatkan data yang ditunjukkan dalam tabel 4.1.

Tabel 4.1 Hasil Observasi Dan Wawancara

No.	Analisa	Deskripsi
1.	Waktu Operasional	Hari Kerja: Senin - Sabtu Jam operasional: 07.00 - 16.00 WIB Waktu lembur: 17.00 - 19.00 WIB
2.	Lingkungan Kerja	Luas tanah: 1 Hektar Jumlah Bangunan: 2 buah bangunan Jarak antar bangunan 8 Meter Bangunan 1 luas: 400 M^2 Fungsi bangunan: untuk produksi dan administrasi Bangunan 2 Luas : 400 M^2 Fungsi bangunan: untuk pergudangan, administrasi dan elektronika Lahan Pengeringan Luas : 200 M^2 Fungsi: pengeringan bahan baku produksi Proses penggilingan menyebabkan area di dalam bangunan dan disekitar bangunan kotor dan berdebu.

		Suhu udara sekitar saat panas 38 derajat celcius di kota Gresik.
3.	Mesin Kerja dan Produksi	Ada 6 Mesin produksi yang biasanya dioperasikan baik bersamaan ataupun bergiliran.
4.	Pegawai	Jumlah pegawai: 30 orang Jam kerja: 9 - 12 Jam Pembagian Kerja 1. Administrasi 7 orang 2. Produksi 12 orang 3. Elektronik 3 orang 4. Gudang 8 orang.
5.	Transaksi	Jumlah transaksi 50 - 400 transaksi perhari.
6.	Jumlah User	Jumlah pengguna pencatatan 40 orang.
7.	Riwayat Penggunaan Komputer	Pernah memiliki 2 buah komputer tetapi tidak bertahan lama dan akhirnya dijual

Dari hasil wawancara diatas dapat diketahui bahwa pemakaian sistem informasi ini nantinya akan digunakan pada pukul 07.00 sampai 16.00 dan jika ada lembur maka bisa sampai 19.00. Terdapat 40 *user* yang akan menggunakan sistem informasi ini untuk kegiatan operasionalnya. infrastruktur yang dibangun nantinya harus dapat melayani 50 - 400 transaksi per harinya. Masalah utama terletak pada keadaan lingkungan yang berdebu dan panas serta riwayat penggunaan komputer. Dengan wilayah berdebu dan panas, 2 buah komputer milik UD Sendang Putra harus dijual karena mengalami kerusakan. Dari hasil tersebut mengindikasikan bahwa komputer tidak cocok dengan keadaan lingkungan UD Sendang Putra dan perlu alternatif lain sebagai pengganti komputer *server* yakni:

1. Laptop

   Laptop merupakan sebuah komputer jinjing *portable* yang bisa digunakan dimana saja. laptop berjalan menggunakan tenaga baterai dan listrik secara langsung. Laptop dapat menjadi alternatif untuk komputer karena mereka memiliki fungsi yang sama. Dalam kasus UD Sendang Putra laptop bisa menjadi alternatif sebagai *server* namun karena daya tahannya yang cukup lemah terhadap debu dan panas maka laptop tidak direkomendasikan untuk digunakan oleh UD Sendang Putra.

2. *Single Board Computer*

   Single board computer (SBC) merupakan komputer berukuran kecil dengan fungsional yang mirip dengan komputer biasa, *single board computer* banyak digunakan pada proyek yang menggunakan sensor sebagai

middleware, selain itu *single board computer* juga biasanya digunakan sebagai metode pembelajaran pengenalan komputer. *Single board computer* sangat cocok digunakan untuk proyek dengan skala kecil sampai menengah dan proyek di lokasi yang tidak bisa dijangkau oleh komputer biasa. ada banyak merek atau produk *single board computer* tetapi pada penelitian ini, UD Sendang Putra menggunakan produk Raspberry Pi.

Setelah diputuskan untuk memilih *single board computer* muncul masalah baru yakni kemampuan *single board computer* untuk melakukan prosesing data. dari masalah di atas, penelitian ini mencoba melakukan pengujian terhadap kemampuan *processing single board computer* untuk menjalankan *web server* dan *database server*. Hasil dari pengujian 1 perangkat *single board computer* bisa dilihat pada tabel 4.2

Tabel 4.2 Hasil Pengujian *Performance* 1 *Single Board Computer*

No.	Total Koneksi	Total Konkurensi	Jumlah Data Pada Basis Data	CPU Usage (%)	Memory Usage (%)
1	10	10	7000	43.1	71
2	20	20	7000	48.6	79
3	30	30	7000	51.3	82
4	40	40	7000	53.6	87
5	50	50	7000	57.4	93

Dari tabel 4.2 didapatkan data bahwa 1 *single board computer* dapat melayani 50 koneksi dengan utilitas 57.4% CPU dan 93% RAM. dari hasil tersebut penggunaan *resource* memory sudah melebihi batas aman penggunaan *single board computer* yakni sebesar 80%. penggunaan di atas batas aman akan meningkatkan resiko kegagalan *hardware* karena *overheating*. Untuk itu penelitian ini memutuskan untuk menggunakan metode *cluster server* agar penggunaan *resource* tetap dibawah batas aman yakni 80%.

4.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan setelah proses identifikasi masalah. Dalam melakukan analisis kebutuhan, penelitian ini selalu bersandar pada hasil observasi dan wawancara pada tabel tabel 4.1. Berdasarkan pada hasil observasi dan wawancara 4.1, didapatkan beberapa informasi yang bisa digunakan sebagai pedoman untuk menganalisis kebutuhan yakni waktu operasional pukul 07.00 sampai 16.00 (19.00 jika lembur) yang berarti sistem hanya akan beroperasi selama 9-12 jam. Total pengguna sistem informasi 40 *user*, keadaan yang berdebu dan panas, riwayat penggunaan komputer dan kondisi geografi UD Sendang Putra.

setelah dilakukan identifikasi maka diputuskan penelitian ini menggunakan *single board computer* dengan metode *cluster server* yakni metode untuk menggabungkan lebih dari 1 perangkat ke dalam sebuah *cluster* untuk bisa mengurangi resource setiap perangkat. Metode ini dipilih untuk menjaga agar penggunaan *resource* dari 1 perangkat SBC bisa tetap dibatas aman yakni 80%. pada metode *cluster server* dibutuhkan lebih dari 1 perangkat *single board computer*. Dilihat dari identifikasi masalah, perangkat yang dibutuhkan adalah 2 *backend server* yang bertugas untuk memproses data (*web server* + basis data) dan juga *load balancer* yang bertugas untuk mendistribusikan permintaan yang masuk. untuk menjaga *load balancer* tetap dibatas aman dan mengatasi *single point of failure* maka perlu untuk menambahkan 1 perangkat tambahan yang berfungsi sebagai backup dari *load balancer* agar saat *load balancer* mengalami kegagalan *request* tetap bisa ditangani dengan baik. untuk mempermudah untuk menganalisa kebutuhan yang dibutuhkan untuk membangun infrastruktur UD Sendang Putra maka penelitian ini akan mengelompokkan kebutuhan menjadi 3 yakni kebutuhan fungsional, kebutuhan non fungsional dan kebutuhan perangkat keras.

4.2.1 Kebutuhan Perangkat Keras

Untuk dapat menjalankan sistem untuk memenuhi kebutuhan maka penulis akan menggunakan *single board computer* yakni Raspberry dengan jumlah tertentu. Untuk melihat rincian jumlah yang dibutuhkan dalam penelitian ini dapat dilihat pada tabel 4.3

Tabel 4.3 Kebutuhan Perangkat Keras

No.	Jumlah	Jenis Kebutuhan	Deskripsi
1.	2	*Backend Server* (BS)	Digunakan untuk melayani permintaan dari klien dan memproses semua pemrosesan sistem informasi.
2.	1	*Main Load Balancer* (MLb)	Digunakan sebagai *load balancer* utama yang nantinya akan digunakan untuk mendistribusikan *trafik*.
3.	1	*Slave Load Balancer* (SLb)	Digunakan sebagai *load balancer* cadangan yang nantinya akan melakukan *take over* jika *load balancer* utama mengalami *failure*.

Sumber: Penulis

4 perangkat tersebut akan dijadikan menjadi 1 *cluster server* yang berhubungan satu sama lain untuk memenuhi kebutuhan pengguna.

4.2.2 Kebutuhan Fungsional

Kebutuhan Fungsional adalah kebutuhan yang berisi mengenai apa apa saja yang nantinya harus disediakan oleh sistem. Pada penelitian ini kebutuhan

fungsional yang dibutuhkan berjumlah tiga, adapun rincian dari kebutuhan fungsional bisa dilihat pada tabel 4.4

Tabel 4.4 Kebutuhan Fungsional

No.	Kebutuhan Fungsional
KF1.	*Cluster server* mampu beroperasi dan melayani permintaan *user* meskipun ada perangkat yang mengalami kegagalan
KF2.	*Cluster server* mampu menyimpan data secara konsisten meskipun ada perangkat yang mengalami kegagalan
KF3.	*Cluster server* mampu berada pada batas aman penggunaan *resource* yakni 80%

4.2.3 Kebutuhan Non Fungsional

Kebutuhan Non Fungsional adalah kebutuhan yang menitikberatkan pada perilaku yang dimiliki oleh sistem. kebutuhan non fungsional juga sering disebut sebagai batasan layanan atau fungsi yang ditawarkan sistem seperti batasan waktu, batasan pengembangan proses, standarisasi dan lain lain. Pada penelitian ini kebutuhan non fungsional yang dibutuhkan berjumlah 5. Adapun rinciannya bisa dilihat pada tabel 4.5.

Tabel 4.5 Kebutuhan Non Fungsional

No.	Kebutuhan Non Fungsional
KNF1.	*Cluster server* mengandung *software load balancing* (Haproxy) yang digunakan untuk mendistribusikan trafik
KNF2.	*Cluster server* mengandung *software failover* (Keepalived) yang digunakan untuk mencegah single point of failure
KNF3.	*Cluster server* mengandung *software web server* (Nginx) yang digunakan untuk melayani kebutuhan *web* pengguna
KNF4.	Cluster *Server* mengandung *software* basis data (MariaDB dengan Galera Cluster) untuk menyimpan dan mereplikasi data
KNF5.	*Cluster server* mengandung *software* scripting (PHP dengan PHP-FPM) untuk menjalankan sistem informasi

BAB 5 PERANCANGAN DAN IMPLEMENTASI

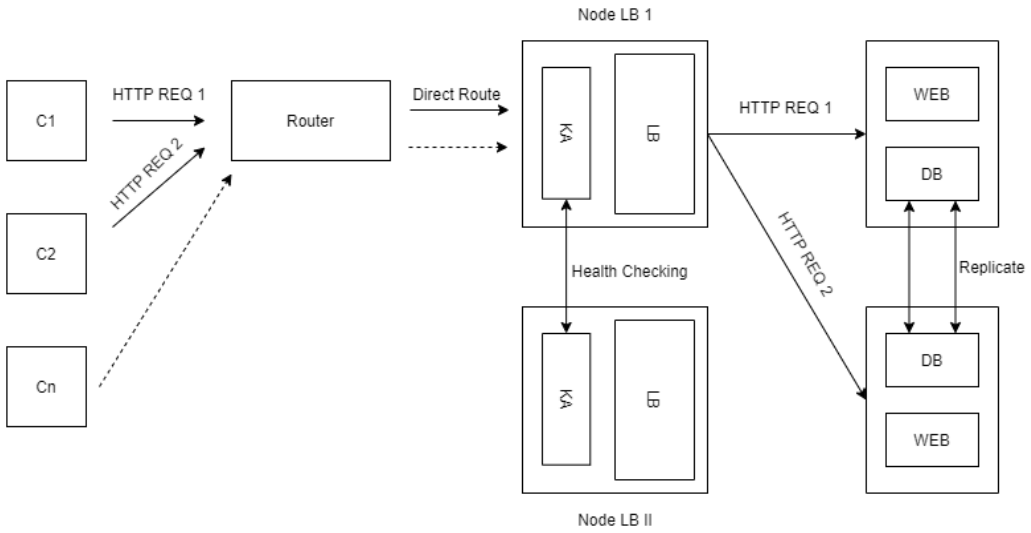
Perancangan dan Implementasi merupakan tahapan penterjemahan proses analisis kebutuhan ke dalam suatu rancangan dan mengimplementasikannya ke sistem yang sebenarnya.

5.1 Perancangan

Perancangan merupakan tahapan yang digunakan sebagai acuan dalam melakukan implementasi dan pengujian dari sistem yang akan dibangun. Proses yang dilaksanakan dalam tahap ini adalah perancangan arsitektur, topologi dan pengujian.

5.1.1 Perancangan Arsitektur

Sistem yang dibangun akan menggunakan arsitektur klien-*server* yang terhubung pada jaringan lokal. Pada sisi klien pengguna bisa mengakses sistem menggunakan perangkat *smartphone* dan *tablet.* Sedangkan pada sisi *server* sistem menggunakan konsep *cluster* dengan satu *load balancer* utama, 1 *load balancer* cadangan dan juga 2 *server*. Perancangan arsitektur yang dibangun bisa dilihat pada gambar 5.1.



Gambar 5.1 Rancangan Arsitektur

Seperti yang terlihat pada gambar 5.1 pengguna akan melakukan akses dengan jaringan lokal melalui router. Router kemudian akan mengarahkan ke *main load balancer* (MLb) dan selanjutnya MLb akan mendistribusikan permintaan kepada *backend server*. MLb akan terus berkoordinasi dengan *slave load balancer* (SLb) melalui *health check* yang dilakukan oleh keepalived secara periodik. Keepalived juga secara dinamis melakukan *health check* dari layanan spesifik pada *server* melalui tiga metode *health check* bawaan yakni koneksi TCP sederhana, HTTP, dan HTTPS. Untuk koneksi TCP, perangkat yang aktif akan secara berkala memeriksa apakah ia dapat terhubung ke perangkat lain pada port tertentu. Untuk

HTTP dan HTTPS, perangkat aktif akan secara berkala mengambil URL di perangkat lain dan memverifikasi kontennya.

Slave load balancer menjalankan peran sistem siaga. Kegagalan perangkat akan ditangani oleh VRRP. Saat memulai, MLb dan SLb akan bergabung dengan grup multicast. Grup *multicast* ini digunakan untuk mengirim dan menerima *advertisement* VRRP. Karena VRRP adalah protokol berbasis prioritas, perangkat dengan prioritas tertinggi dipilih sebagai *master*. Setelah perangkat terpilih sebagai *master*, perangkat akan bertanggung jawab untuk mengirim *advertisement* VRRP secara berkala ke grup *multicast*.

Jika MLb gagal menerima *advertisement* dalam periode waktu tertentu (berdasarkan interval), *master* baru akan dipilih. *Master* baru akan mengambil alih virtual IP dan mengirim message Address Resolution Protocol (ARP). Ketika MLb kembali ke layanan aktif, itu bisa menjadi cadangan atau *master*. tergantung oleh prioritas perangkat. Untuk menjaga konsistensi data pada 2 basis data maka dilakukan namanya replikasi. replikasi dilakukan dengan Galera Cluster milik MariaDB. Galera Cluster akan mengupdate data satu sama lain dengan menggunakan algoritma Incremental yakni IST dengan WSREP.

Penggunaan perangkat lunak cluster server dapat dilihat pada tabel 5.1

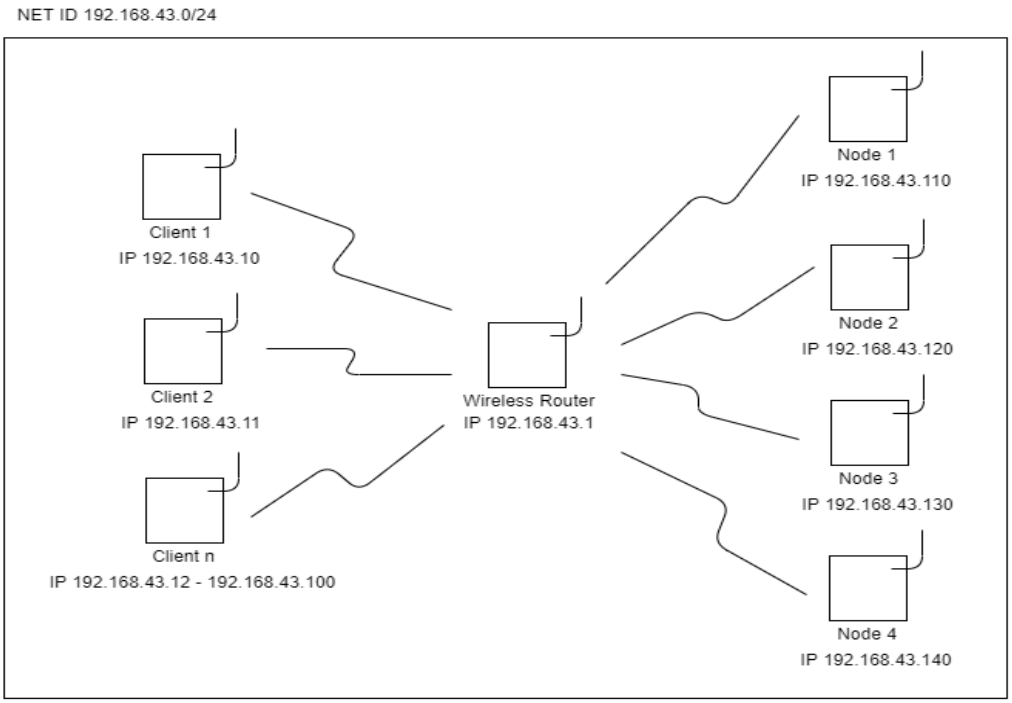
Tabel 5.1 Perangkat Lunak

No.	Perangkat Lunak	Versi	Perangkat
1.	Raspberry Pi OS (32-bit) Lite	Jan 2020	Semua Perangkat
2.	Haproxy	2.1	*Load Balancer*
3.	Keepalived	2.0.19	*Load Balancer*
4.	Nginx	1.18.0	*Backend Server*
5.	PHP	7.3	*Backend Server*
6.	MariaDB	10.3.15	*Backend Server*
7.	Rsync	3.0	Semua Perangkat

Pemilihan dari versi perangkat lunak yang digunakan berdasarkan versi yang paling stabil (*Stable Version*). Jika ada versi baru dari perangkat lunak maka pembaruan perangkat lunak akan dilaksanakan pada proses *maintenance.*

5.1.2 Perancangan Topologi

Setelah perancangan arsitektur dilakukan maka tahapan selanjutnya adalah perancangan topologi beserta detail detail alamat yang akan digunakan. Rancangan topologi bisa dilihat pada gambar 5.2.



Gambar 5.2 Rancangan Topologi

Untuk memudahkan dalam membaca gambar diatas maka rincian topologi dapat dilihat pada tabel 5.2

Tabel 5.2 Tabel Alamat IP Perangkat

No.	Perangkat	Alamat IP
1.	Wireless Router	192.168.43.1
2.	Client	192.168.43.10 - 192.168.43.100
3.	*Node* 1 (MLb)	192.168.43.110
4.	*Node* 2 (SLb)	192.168.43.120
5.	*Node* 3 (BS1)	192.168.43.130
6.	*Node* 4 (BS2)	192.168.43.140
7.	Floating IP (IP Virtual / tidak memiliki perangkat keras)	192.168.43.200

Pada rincian tersebut terdapat satu alamat IP yang tidak terafiliasi secara fisik pada *hardware* manapun yakni IP virtual milik keepalived. Alamat ini yang nantinya akan digunakan untuk mengakses *cluster server*.

5.1.3 Perancangan Pengujian

Pada bagian perancangan pengujian penelitian ini menentukan bagaimana infrastruktur diuji dan diambil datanya. berikut ini adalah jenis pengujian yang akan dilakukan pada penelitian yang bisa dilihat pada tabel 5.3.

Tabel 5.3 Jenis Pengujian

No	Jenis Pengujian	Tujuan Pengujian	Cara Melakukan Pengujian
1	*Performance Testing*	Mendapatkan data utilitas sistem yakni penggunaan CPU dan Memori	Melakukan *request* dengan total koneksi dan konkurensi incremental pada data dengan jumlah tertentu
2	*Black Box Testing*	Mendapatkan nilai kevalidan terhadap kebutuhan fungsional dan non fungsional	Melakukan *shutdown* terhadap 1 atau lebih perangkat untuk menguji fungsional dan non fungsional
3.	Menghitung Availabilitas	Mendapatkan persentase tingkat availabilitas selama sistem berjalan	Memantau sistem yang telah berjalan pada periode tertentu

Jenis pengujian pertama adalah *performance testing*. Pengujian ini dilakukan dengan menggunakan data *dummy* yang telah disesuaikan dengan kolom transaksi pada sistem informasi. Pengujian ini menggunakan data yang telah ditambahkan pada MariaDB dan sudah tereplikasi untuk *server* 1 dan *server* 2. *Testing scenario* dalam melakukan pengujian performa dapat dilihat pada tabel 5.3

Tabel 5.4 *Performance Testing*

No.	Total Koneksi	Total Konkurensi	Jumlah Data Pada Basis Data
1	10	10	7000
2	20	20	7000
3	30	30	7000
4	40	40	7000
5	50	50	7000
6	60	60	7000
7	70	70	7000

8	80	80	7000
9	90	90	7000
10	100	100	7000
11	110	110	7000
12	120	120	7000
13	130	130	7000
14	140	140	7000
15	150	150	7000

Pada pengujian performa, penelitian ini menggunakan 7000 baris data yang akan diakses secara incremental mulai dari total koneksi 10 sampai 150. Pengujian kedua adalah pengujian model *black box testing. Testing scenario* dari *black box testing* dapat dilihat pada tabel 5.5.

Tabel 5.5 *Blacbox Testing*

No	*Main load balancer*	*Slave load balancer*	*Backend server 1*	*Backend server 2*
1	Non Aktif	Aktif	Aktif	Aktif
2	Aktif	Non Aktif	Aktif	Aktif
3	Aktif	Aktif	Non Aktif	Aktif
4	Aktif	Aktif	Aktif	Non Aktif
5	Non Aktif	Aktif	Non Aktif	Aktif
6	Non Aktif	Aktif	Aktif	Non Aktif
7	Aktif	Non Aktif	Aktif	Non Aktif
8	Aktif	Non Aktif	Non Aktif	Aktif

Pengujian *black box* dilakukan dengan cara mengaktifkan atau menonaktifkan 1 atau lebih perangkat untuk melihat apakah *server* tetap berjalan dengan normal atau tidak normal.

5.1.4 Perancangan Pengumpulan Data

Setelah *scenario* pengujian telah ditentukan, tahap selanjutnya adalah rancangan untuk pengumpulan data. Tabel rancangan pengumpulan data pada *performance testing* dapat dilihat pada Tabel 5.6

Tabel 5.6 Pengumpulan Data *Performance Testing*

Percobaan X Koneksi		
Percobaan Nomor	Penggunaan CPU	Penggunaan Memori
Percobaan 1	X %	Y %
Percobaan 2	X %	Y %
Percobaan 3	X %	Y %
Percobaan 4	X %	Y %
Percobaan 5	X %	Y %
Percobaan 6	X %	Y %
Percobaan 7	X %	Y %
Percobaan 8	X %	Y %
Percobaan 9	X %	Y %
Percobaan 10	X %	Y %
Rata Rata Penggunaan	Z %	Z %

Pada Tabel 5.6, 1 scenario akan dijalankan 10 kali kemudian diambil reratanya untuk digunakan pada hasil pengujian. Untuk *black box testing* rancangan pengumpulan datanya dapat dilihat pada Tabel 5.7

Tabel 5.7 Pengumpulan Data *Black Box Testing*

Percobaan Scenario N				
Percobaan No	Tetap Dapat Melayani U*ser*	Data Konsisten	Haproxy, Keepalived dan Replikasi Tetap Berjalan	Nginx, MariaDB, dan PHP Tetap Berjalan
Percobaan 1				
Percobaan 2				
Percobaan 3				
Percobaan 4				
Percobaan 5				
Percobaan 6				
Percobaan 7				
Percobaan 8				
Percobaan 9				
Percobaan 10				

Pada Tabel 5.7, 1 *scenario* dijalankan 10 kali untuk melihat konsistensi dan validitas pengujian berdasarkan kebutuhan fungsional dan non fungsional. Untuk menghitung availabilitas sistem rancangan pengumpulan datanya dapat dilihat pada Tabel 5.8

Tabel 5.8 Pengumpulan Data Availabilitas Sistem

No	Minggu (Tgl-Tgl)	Waktu Aktif (Jam)	Waktu Tidak Aktif (Jam)	Waktu Maintenance (Jam)
1	Minggu 1	X Jam	Y Jam	Z Jam
2	Minggu 2	X Jam	Y Jam	Z Jam
3	Minggu 3	X Jam	Y Jam	Z Jam
7	Minggu N	X Jam	Y Jam	Z Jam
Jml	X Minggu	X Jam	Y Jam	Z Jam

5.2 Implementasi

Implementasi merupakan tahapan penerjemahan proses analisis kebutuhan dan perancangan ke dalam sistem yang sebenarnya. Proses yang dilaksanakan dalam tahapan implementasi sistem adalah menentukan spesifikasi perangkat keras, mengimplementasikan arsitektur, topologi dan semua perangkat lunak yang dibutuhkan.

5.2.1 Spesifikasi Perangkat Keras

Setelah jumlah perangkat ditentukan maka kita akan mencari *single board computer* yang sesuai dengan kebutuhan. Adapun spesifikasi yang dibutuhkan dalam penelitian ini dapat dilihat pada tabel 5.9

Tabel 5.9 Spesifikasi Perangkat Keras

No.	Jenis Kebutuhan	Nama Perangkat	Spesifikasi
1	*Load Balancer* *Backend Server*	Raspberry Zero W	1GHz, single-core CPU, 512MB RAM Mini HDMI and USB On-The-Go ports Micro USB power HAT-compatible 40-pin header, Composite video and reset headers,CSI camera connector

Dari tabel 5.6 penelitian ini menggunakan 1 tipe Raspberry Pi yakni Pi Zero W sebagai *load balancer* dan juga sebagai *backend server*. yang membedakan spesifikasi dari *load balancer* dan *backend server* adalah kapasitas storage nya. Pada *load balancer* kapasitas storage nya adalah 16 GB sedangkan pada *backend server* adalah 32 GB. *backend server* harus memiliki kapasitas yang besar karena digunakan sebagai tempat penyimpanan basis data dan juga sistem informasi.

5.2.2 Implementasi Arsitektur dan Topologi

Berdasarkan perancangan pada bab 5.1.2 maka *main load balancer* akan mendapatkan alamat IP statis 192.168.43.110/24. Konfigurasi alamat IP disimpan dalam *file* /etc/dhcpcd.conf dan untuk konfigurasinya bisa dilihat pada tabel 5.10

Tabel 5.10 Konfigurasi Alamat IP *Main Load Balancer*

```
1  Interface wlan0
2  static ip_address=192.168.43.110/24
3  static routers=192.168.43.1
4  static domain_name_servers=192.168.43.1 8.8.8.8
```

Sumber: Perancangan

Berdasarkan tabel 5.10 *interface* yang digunakan pada *main load balancer* adalah *interface wireless* (wlan0). Alamat IP diatur menjadi statis sesuai dengan perancangan dan perlu juga untuk menambahkan alamat *router* dan juga *DNS*. Untuk hasil konfigurasinya bisa dilihat pada gambar 5.3

```
pi@balancer:~ $ ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.110  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::6866:1f14:2e2e:724f  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:74:6b:7f  txqueuelen 1000  (Ethernet)
        RX packets 290  bytes 27822 (27.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 442  bytes 45254 (44.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Gambar 5.3 Hasil Konfigurasi Alamat IP *Main Load Balancer*

Sumber: Perancangan

Dapat dilihat pada gambar 5.3 bahwa alamat IP telah berhasil dipasang dan sudah dapat berjalan sebagaimana semestinya yakni pada alamat 192.168.43.110 dengan netmask 255.255.255.0.

Berdasarkan perancangan pada bab 5.1.2 maka *slave load balancer* akan mendapatkan alamat IP statis 192.168.43.120/24. Konfigurasi alamat IP disimpan dalam *file* /etc/dhcpcd.conf dan untuk konfigurasinya bisa dilihat pada tabel 5.11

Tabel 5.11 Konfigurasi Alamat IP *Slave Load Balancer*

```
1  Interface wlan0
2  static ip_address=192.168.43.120/24
3  static routers=192.168.43.1
4  static domain_name_servers=192.168.43.1 8.8.8.8
```

Sumber: Perancangan

Berdasarkan tabel 5.11 *interface* yang digunakan pada *slave load balancer* adalah *interface wireless* (wlan0). Alamat IP diatur menjadi statis sesuai dengan perancangan dan perlu juga untuk menambahkan alamat *router* dan juga *DNS*. Untuk hasil konfigurasinya bisa dilihat pada gambar 5.4

```
pi@balancer2:~ $ ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.120  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::bcf5:20aa:7b7a:7dcb  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:f8:1b:8f  txqueuelen 1000  (Ethernet)
        RX packets 1433  bytes 137199 (133.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2500  bytes 225111 (219.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Gambar 5.4 Hasil Konfigurasi Alamat IP *Slave Load Balancer*

Sumber: Perancangan

Dapat dilihat pada gambar 5.4 bahwa alamat IP telah berhasil dipasang dan sudah dapat berjalan sebagaimana semestinya yakni pada alamat 192.168.43.120 dengan netmask 255.255.255.0

Berdasarkan perancangan pada bab 5.1.2 maka *server* 1 akan mendapatkan alamat IP statis 192.168.43.130/24. Konfigurasi alamat IP disimpan dalam file /etc/dhcpcd.conf dan untuk konfigurasinya bisa dilihat pada tabel 5.12

Tabel 5.12 Konfigurasi Alamat IP *Server* 1

```
1  Interface wlan0
2  static ip_address=192.168.43.130/24
3  static routers=192.168.43.1
4  static domain_name_servers=192.168.43.1 8.8.8.8
```

Sumber: Perancangan

Berdasarkan tabel 5.12 *interface* yang digunakan pada *server* 1 adalah *interface wireless* (wlan0). Alamat IP diatur menjadi statis sesuai dengan perancangan dan perlu juga untuk menambahkan alamat *router* dan juga *DNS*. Untuk hasil konfigurasinya bisa dilihat pada gambar 5.5

```
pi@server1:~ $ ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.130  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::a88a:81bc:7d78:c5e5  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:76:a0:bb  txqueuelen 1000  (Ethernet)
        RX packets 134  bytes 10937 (10.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 74  bytes 12308 (12.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Gambar 5.5 Hasil Konfigurasi Alamat IP *Server* 1

Sumber: Perancangan

Dapat dilihat pada gambar 5.5 bahwa alamat IP telah berhasil dipasang dan sudah dapat berjalan sebagaimana semestinya yakni pada alamat 192.168.43.130 dengan netmask 255.255.255.0

Berdasarkan perancangan pada bab 5.1.2 maka *server* 2 akan mendapatkan alamat IP statis 192.168.43.140/24. Konfigurasi alamat IP disimpan dalam file /etc/dhcpcd.conf dan untuk konfigurasinya bisa dilihat pada tabel 5.13

Tabel 5.13 Konfigurasi Alamat IP *Server* 2

```
1  Interface wlan0
2  static ip_address=192.168.43.140/24
3  static routers=192.168.43.1
4  static domain_name_servers=192.168.43.1 8.8.8.8
```

Sumber: Perancangan

Berdasarkan tabel 5.13 *interface* yang digunakan pada *server* 1 adalah *interface wireless* (wlan0). Alamat IP diatur menjadi statis sesuai dengan perancangan dan perlu juga untuk menambahkan alamat *router* dan juga *DNS*. Untuk hasil konfigurasinya bisa dilihat pada gambar 5.6

```
pi@server2:~ $ ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.140  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::a88a:81bc:7d78:c5e5  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:76:a0:bb  txqueuelen 1000  (Ethernet)
        RX packets 83  bytes 7860 (7.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 76  bytes 12611 (12.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Gambar 5.6 Hasil Konfigurasi Alamat IP *Server* 2

Sumber: Perancangan

Dapat dilihat pada gambar 5.6 bahwa alamat IP telah berhasil dipasang dan sudah dapat berjalan sebagaimana semestinya yakni pada alamat 192.168.43.140 dengan netmask 255.255.255.0

5.2.3 Implementasi Perangkat Lunak

Pada bagian implementasi perangkat lunak, penelitian ini mengimplementasikan semua perangkat lunak yang dibutuhkan oleh *load balancer* dan *backend server*. pada *load balancer* hanya membutuhkan 2 perangkat lunak yakni Haproxy dan Keepalived.

```
pi@balancer:~ $ service haproxy status
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor preset: enabled
   Active: active (running) since Thu 2020-03-26 02:22:16 WIB; 4 months 14 days ago
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
  Process: 409 ExecStartPre=/usr/sbin/haproxy -f $CONFIG -c -q $EXTRAOPTS (code=exited,
 Main PID: 414 (haproxy)
   Memory: 3.3M
   CGroup: /system.slice/haproxy.service
           ├─414 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
           └─418 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
```

Gambar 5.7 Haproxy *Main Load Balancer*

Sumber: Perancangan

Dapat dilihat pada gambar 5.7 yakni proses pengecekan status, Haproxy otomatis berjalan setelah proses instalasi.

```
pi@balancer:~ $ service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-03-26 02:22:15 WIB; 4 months 14 days ago
 Main PID: 406 (keepalived)
    Memory: 6.9M
    CGroup: /system.slice/keepalived.service
            ├─406 /usr/sbin/keepalived --dont-fork
            └─417 /usr/sbin/keepalived --dont-fork
```

Gambar 5.8 Keepalived *Main Load Balancer*

Sumber: Perancangan

Keepalived status menunjukkan aktif seperti terlihat pada gambar 5.8 tetapi belum dapat berfungsi. Hal ini wajar terjadi mengingat belum ada konfigurasi yang dilakukan pada Keepalived.

```
pi@balancer2:~ $ service haproxy status
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor preset: enabled
   Active: active (running) since Thu 2020-03-26 02:22:30 WIB; 4 months 14 days ago
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
  Process: 323 ExecStartPre=/usr/sbin/haproxy -f $CONFIG -c -q $EXTRAOPTS (code=exited,
 Main PID: 332 (haproxy)
    Memory: 3.3M
    CGroup: /system.slice/haproxy.service
            ├─332 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
            └─339 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
```

Gambar 5.9 Haproxy *Slave Load Balancer*

Sumber: Perancangan

Dapat dilihat pada gambar 5.9 yakni proses pengecekan status, Haproxy otomatis berjalan setelah proses instalasi tetapi belum dapat melakukan *load balancing* dikarenakan perlu ada pengaturan tambahan untuk bisa berfungsi menjadi *load balancer*.

```
pi@balancer2:~ $ service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset:
   Active: active (running) since Sun 2020-08-09 11:04:12 WIB; 4s ago
 Main PID: 711 (keepalived)
    Memory: 1.1M
    CGroup: /system.slice/keepalived.service
            ├─711 /usr/sbin/keepalived --dont-fork
            └─712 /usr/sbin/keepalived --dont-fork
```

Gambar 5.10 Keepalived *Slave Load Balancer*

Sumber: Perancangan

Keepalived status menunjukkan aktif seperti terlihat pada gambar 5.10 tetapi belum dapat berfungsi. Hal ini wajar terjadi mengingat belum ada konfigurasi yang dilakukan pada Keepalived.

Untuk *backend server* ada 3 perangkat lunak utama yang digunakan yakni Nginx, PHP dan MariaDB serta beberapa add on tambahan yang sudah include di

dalam paket instalasi ketiga *software* tersebut seperti Rsync, Galera Cluster dan PHP-FPM. Untuk hasil implementasi bisa dilihat pada gambar 5.11 sampai dengan 5.13

```
pi@server1:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-03-23 16:02:56 WIB; 4 months 17 days ago
     Docs: man:nginx(8)
  Process: 11468 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited,
  Process: 11469 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0
 Main PID: 11470 (nginx)
    Tasks: 2 (limit: 1150)
   Memory: 3.5M
   CGroup: /system.slice/nginx.service
           ├─11470 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           └─11471 nginx: worker process
```

Gambar 5.11 Implementasi Nginx

Sumber: Perancangan

```
pi@server1:~$ systemctl status mariadb
● mariadb.service - MariaDB 10.3.15 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-03-17 23:41:55 WIB; 4 months 22 days ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
 Main PID: 2638 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 36 (limit: 1150)
   Memory: 178.5M
   CGroup: /system.slice/mariadb.service
           └─2638 /usr/sbin/mysqld --wsrep-new-cluster --wsrep_start_position=00000000-0000-
```

Gambar 5.12 Implementasi MariaDB

Sumber: Perancangan

```
pi@server1:~$ systemctl status php7.3-fpm.service
● php7.3-fpm.service - The PHP 7.3 FastCGI Process Manager
   Loaded: loaded (/lib/systemd/system/php7.3-fpm.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-03-19 17:05:48 WIB; 4 months 21 days ago
     Docs: man:php-fpm7.3(8)
 Main PID: 10704 (php-fpm7.3)
   Status: "Processes active: 0, idle: 2, Requests: 250, slow: 0, Traffic: 0req/sec"
    Tasks: 3 (limit: 1150)
   Memory: 16.8M
   CGroup: /system.slice/php7.3-fpm.service
           ├─10704 php-fpm: master process (/etc/php/7.3/fpm/php-fpm.conf)
           ├─10705 php-fpm: pool www
           └─10706 php-fpm: pool www
```

Gambar 5.13 Implementasi PHP

Sumber: Perancangan

5.2.4 Implementasi *Failover* dan *Load balancing*

Implementasi selanjutnya adalah implementasi untuk *failover* dan *load balancing*. implementasi ini dilakukan hanya pada MLb dan juga SLb. Tabel implementasi dapat dilihat pada Tabel 5.14 dan 5.15.

Tabel 5.14 Konfigurasi Keepalived *Main Load Balancer*

```
1  vrrp_instance SP_1 {
2      state MASTER
3      interface eth0
4      virtual_router_id 101
5      priority 101
```

```
6       advert_int 1
7       authentication {
8           auth_type PASS
9           auth_pass sendangputra
10      }
11      virtual_ipaddress {
12          192.168.43.200
13      }
14  }
```

Sumber: Perancangan

Tabel 5.15 Konfigurasi Keepalived *Slave Load Balancer*

```
1   vrrp_instance SP_1 {
2       state MASTER
3       interface eth0
4       virtual_router_id 101
5       priority 50
6       advert_int 1
7       authentication {
8           auth_type PASS
9           auth_pass sendangputra
10      }
11      virtual_ipaddress {
12          192.168.43.200
13      }
14  }
```

Sumber: Perancangan

Pada Keepalived terdapat konfigurasi untuk menentukan mana yang menjadi *load balancer* utama dan *load balancer* cadangan yakni pada command priority. *Load balancer* dengan priority tertinggi akan menjadi yang utama. Selain itu juga yang tidak kalah penting adalah alamat IP virtual kita harus menyamakan pengaturan untuk virtual *router id* dan alamat IP virtual dengan *load balancer* cadangan agar saat *load balancer* utama mati maka *load balancer* cadangan akan langsung mengambil alih peran *load balancer* utama seperti yang terlihat pada tabel 5.14 dan 5.15. untuk mengakses *server cluster* kita hanya harus mengakses alamat IP virtual.

Tabel 5.16 Konfigurasi Haproxy *Main Load Balancer*

```
1   frontend lb1
2       bind *:80
3       mode http
4       default_backend lbb1
5
6   backend lbb1
7       mode http
8       balance leastconn
9       http-request set-header X-Forwarded-For %[src]
10      option httpchk HEAD / HTTP/1.0
11      server sp1 192.168.43.130:80 check fall 3 rise 2 inter
12  1s
13      server sp2 192.168.43.140:80 check fall 3 rise 2 inter
14  1s
15
16  listen stats
```

```
17          bind *:8181
18          stats enable
19          stats refresh 30s
20          stats show-node
21          stats auth admin:admin
22          stats uri /stats
```

Sumber: Perancangan

Tabel 5.17 Konfigurasi Haproxy *Slave Load Balancer*

```
1   frontend lb2
2           bind *:80
3           mode http
4           default_backend lbb2
5
6   backend lbb2
7           mode http
8           balance leastconn
9           http-request set-header X-Forwarded-For %[src]
10          option httpchk HEAD / HTTP/1.0
11          server sp1 192.168.43.130:80 check fall 3 rise 2 inter
12  1s
13          server sp2 192.168.43.140:80 check fall 3 rise 2 inter
14  1s
15
16  listen stats
17          bind *:8181
18          stats enable
19          stats refresh 30s
20          stats show-node
21          stats auth admin:admin
22          stats uri /stats
```

Sumber: Perancangan

Dapat dilihat pada tabel 5.16 dan 5.17 terdapat tiga konfigurasi utama pada Haproxy yakni *frontend backend* dan juga *statistic*. Konfigurasi pertama adalah konfigurasi *frontend*. *Frontend* digunakan untuk melakukan *handling* semua *request* yang masuk ke *load balancer.* Pada kasus ini *frontend* berjalan pada mode http dan *listen* di *port* 80*.* Konfigurasi kedua adalah *backend, backend* digunakan untuk melakukan *handling* terhadap *server*. Pada kasus ini *backend* berinteraksi dengan 2 *server* yakni sp1 dan sp2 dengan metode *least connection.* Setiap 1 detik *load balancer* melakukan *health check* pada kedua *server*, jika dalam 3 kali pengecekan *server* gagal membalas maka *server* dinyatakan mati dan tidak akan melayani *request*. Yang terakhir adalah statistik yang digunakan untuk memantau *load balancer,* Statistik dapat diakses pada port 8181

5.2.5 Implementasi Replikasi dengan Galera Cluster

Implementasi berikutnya merupakan implementasi untuk replikasi Galera Cluster. Implementasi ini hanya dilakukan pada *backend server*. Berikut ini merupakan hasil implementasinya dapat dilihat pada Tabel 5.18 dan 5.19.

Tabel 5.18 Konfigurasi *Galera Cluster Server* 1

```
1  [galera]
2  wsrep_on = on
3  wsrep_provider = /lib/galera/libgalera_smm.so
4  wsrep_cluster_address =
5  gcomm://192.168.43.130,192.168.43.140
6  wsrep_cluster_name = galera_cluster_0
7  default_storage_engine = InnoDB
8  innodb_autoinc_lock_mode = 2
9  innodb_doublewrite = 1
10 binlog_format = ROW
```

Sumber: Perancangan

Tabel 5.19 Konfigurasi *Galera Cluster Server* 2

```
1  [galera]
2  wsrep_on = on
3  wsrep_provider = /lib/galera/libgalera_smm.so
4  wsrep_cluster_address =
5  gcomm://192.168.43.130,192.168.43.140
6  wsrep_cluster_name = galera_cluster_0
7  default_storage_engine = InnoDB
8  innodb_autoinc_lock_mode = 2
9  innodb_doublewrite = 1
10 binlog_format = ROW
```

Sumber: Perancangan

Pada Tabel 5.18 dan 5.19 terdapat konfigurasi yang dipasang pada masing masing *server*. Pertama penulis harus memasang *galera cluster* pada mode hidup dan juga mengarahkan ke *library* galera-3. Konfigurasi selanjutnya adalah menentukan alamat dari basis data yang ingin dilakukan replikasi yakni 192.168.43.130 dan 192.168.43.140. setelah menentukan alamat maka kita juga harus menentukan *default storage engine* yang digunakan basis data dan jenis log formatnya. Saat semuanya telah terpasang maka *galera cluster* di *server* 1 akan berjalan dan otomatis *server* basis data di *server* 2 juga akan berjalan. Galera Cluster menggunakan WSREP sebagai protokol untuk mengirimkan transaksi dengan cara *incremental* (IST).

BAB 6 PENGUJIAN DAN HASIL

6.1 Pengujian

Pada tahap pengujian penelitian ini melakukan pengujian dengan skenario yang telah disiapkan pada perancangan pengujian yakni menguji *performance* dari *cluster server*, menguji kebutuhan fungsional dan non fungsional serta menghitung tingkat availabilitas dari *server cluster*.

6.1.1 Hasil Pengujian Performance

Dalam pengujian performance penelitian ini melakukan 10 kali pengujian pada 1 scenario dan melakukan penghitungan rata rata dari hasil pengujian tersebut dari hasil pengujian tersebut dihasilkan data dapat dilihat pada tabel 6.1 dan 6.2.

Tabel 6.1 Hasil Pengujian *Performance* Cluster Server

No.	Total Koneksi	Total Konkurensi	Jumlah Data Pada Basis Data	CPU Usage (%)	Memory Usage (%)
1	10	10	7000	18.5	49
2	20	20	7000	22.3	51
3	30	30	7000	25.2	57
4	40	40	7000	29.1	62
5	50	50	7000	35.5	64
6	60	60	7000	39.8	60
7	70	70	7000	41.3	63
8	80	80	7000	42.6	65
9	90	90	7000	47.7	72
10	100	100	7000	50.1	75
11	110	110	7000	51.3	79
12	120	120	7000	54.7	81
13	130	130	7000	60.3	83
14	140	140	7000	63.3	85

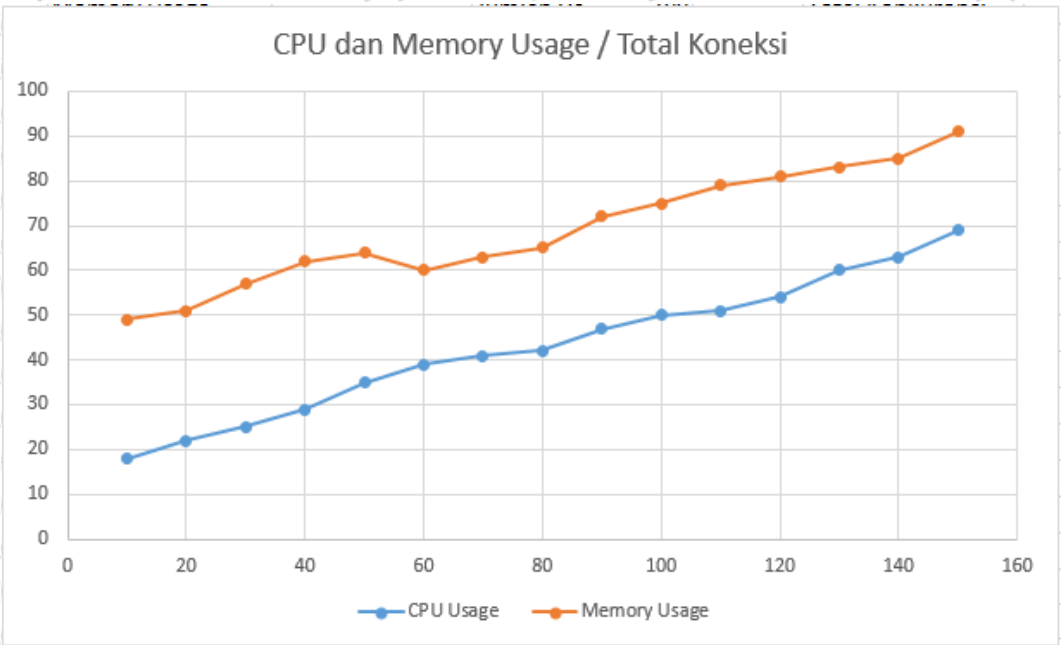
15	150	150	7000	69.7	91

Tabel 6.2 Hasil Pengujian *Performance* Non Cluster

No.	Total Koneksi	Total Konkurensi	Jumlah Data Pada Basis Data	CPU Usage (%)	Memory Usage (%)
1	10	10	7000	43.1	71
2	20	20	7000	48.6	79
3	30	30	7000	51.3	82
4	40	40	7000	53.6	87
5	50	50	7000	57.4	93

Pada hasil pengujian 1 perangkat didapatkan hasil yang bervariasi mulai dari 71 % penggunaan memori sampai 93%. Penggunaan paling optimal untuk 1 perangkat non *cluster* adalah pada total 20 koneksi. Dari sini sudah terlihat perbandingan dan perbedaan hasil dari penggunaan *cluster server* dan 1 perangkat.

jika digambarkan dalam grafik maka data yang terbentuk adalah seperti pada gambar 6.1



Gambar 6.1 Grafik CPU dan Memory Usage

Pengujian tersebut dimulai dengan menguji 10 koneksi secara konkuren dan didapatkan hasil dari penggunaan *resource* yakni 18.5% CPU dan 49% Memori. pengujian dilakukan dengan kelipatan 10. pada pengujian ke 5 dengan total

koneksi 50 *cluster server* ini menggunakan 35.5% CPU dan 64% RAM, hal ini jelas lebih baik dari pengujian 1 perangkat yang dilakukan oleh penelitian ini yang menghasilkan penggunaan Memori melebihi batas aman yakni 80%. pengujian diteruskan sampai dengan total koneksi yakni 150 dengan penggunaan CPU mencapai 69.7% dan Memori 91% jelas dikatakan sudah melewati batas aman. Penggunaan paling ideal adalah pada total koneksi 110 dengan penggunaan CPU sebesar 51.3 dan memori 79% yang belum menembus batas aman penggunaan perangkat yakni 80%. Dari hasil tersebut terlihat kecenderungan yakni semakin banyak total koneksi dan konkurensi maka akan semakin tinggi pula penggunaan *resource* CPU dan memori

6.1.2 Hasil Pengujian Fungsional

Pengujian pada Pengujian fungsional dan non fungsional dilakukan dengan metode *black box* berdasarkan tabel 5.4 yakni dengan mengaktifkan dan menonaktifkan 1 sampai 2 perangkat. Untuk memastikan hasil dari pengujian black box, penelitian ini melakukan pengujian *black box* secara berulang. Hasil dari pengujian fungsional menunjukkan bahwa semua kebutuhan telah terpenuhi. matinya salah satu perangkat tidak berefek pada keseluruhan *cluster server*. Hasil dari pengujian dapat dilihat pada tabel 6.3 dan 6.4

Tabel 6.3 Hasil Pengujian Fungsional

No	Nama Tes	Hasil Yang Diharapkan	Hasil Yang Diperoleh	Status
KF1	*Cluster server* mampu beroperasi dan melayani permintaan *user* meskipun ada perangkat yang mengalami kegagalan	*Cluster server* tetap dapat melayani permintaan meskipun ada perangkat yang mengalami kegagalan	*Cluster server* tetap dapat melayani permintaan meskipun ada perangkat yang mengalami kegagalan	Valid
KF2	*Cluster server* mampu menyimpan data secara konsisten meskipun ada perangkat yang mengalami kegagalan	*Cluster server* tetap dapat menyimpan data dengan konsisten meskipun ada perangkat yang mengalami kegagalan	*Cluster server* tetap dapat menyimpan data dengan konsisten meskipun ada perangkat yang mengalami kegagalan	Valid
KF3	*Cluster server* mampu berada pada batas aman	*Cluster server* dapat menangani permintaan pada	*Cluster server* dapat menangani permintaan pada	Valid

	penggunaan *resource* yakni 80%	batas aman penggunaan *resource* perangkat yakni 80%	batas aman penggunaan *resource* perangkat yakni 80%	

Tabel 6.4 Hasil Pengujian Non Fungsional

No	Nama Tes	Hasil Yang Diharapkan	Hasil Yang Diperoleh	Status
KNF1	*Cluster server* mengandung *software load balancing* (Haproxy) yang digunakan untuk mendistribusikan trafik	Sistem mampu menjalankan Haproxy dan mendistribusikan trafik yang masuk	Sistem mampu menjalankan Haproxy dan mendistribusikan trafik yang masuk	Valid
KNF2	*Cluster server* mengandung *software failover* (Keepalived) yang digunakan untuk mencegah single point of failure	Sistem mampu menjalankan Keepalived dan mencegah *single point of failure*	Sistem mampu menjalankan Keepalived dan mencegah *single point of failure*	Valid
KNF3	*Cluster server* mengandung *software web server* (Nginx) yang digunakan untuk melayani kebutuhan *web* pengguna	Sistem mampu menjalankan Nginx dan melayani kebutuhan *web* pengguna	Sistem mampu menjalankan Nginx dan melayani kebutuhan *web* pengguna	Valid
KNF4	Cluster *Server* mengandung *software* basis data (MariaDB dengan Galera Cluster) untuk menyimpan dan mereplikasi data	Sistem mampu menjalankan MariaDB dan Galera Cluster untuk menyimpan dan mereplikasi data	Sistem mampu menjalankan MariaDB dan Galera Cluster untuk menyimpan dan mereplikasi data	Valid
KNF5	*Cluster server* mengandung	Sistem mampu menjalankan PHP	Sistem mampu menjalankan PHP	Valid

	software scripting (PHP dengan PHP-FPM) untuk menjalankan sistem informasi	untuk melakukan prosesing sistem informasi	untuk melakukan prosesing sistem informasi	

6.1.3 Menghitung Tingkat *Availability Cluster Server*

Pada pengukuran tingkat availabilitas penelitian ini menggunakan data yang diambil dari UD Sendang Putra selama 112 Hari. Data dapat dilihat pada tabel 6.5.

Tabel 6.5 Data Perhitungan *High Availability*

No	Minggu (Tgl-Tgl)	Waktu Aktif (Jam)	Waktu Tidak Aktif (Jam)	Waktu Maintenance (Jam)
1	1 (23-29) Jan	63	0	0
2	2 (30-5) Feb	63	0	0
3	3 (6-12) Feb	63	0	0
4	4 (13-19) Feb	63	0	0
5	5 (20-26) Feb	63	3	3
6	6 (27-4) Mar	63	0	0
7	7 (5-11) Mar	63	0	0
8	8 (12-18) Mar	63	0	0
9	9 (19-25) Mar	63	3	3
10	10 (26-1) Apr	63	0	0
11	11 (2-8) Apr	63	0	0
12	12 (9-15) Apr	63	0	0
13	13 (16-22) Apr	60	3	3
14	14 (23-29) Apr	63	0	0
15	15 (30-6) Mei	63	0	0
16	16 (7-13) Mei	63	6	0
Jml	112	1008	15	9

Berdasarkan tabel hasil perhitungan 6.5 maka tingkat availabilitas sistem dihitung berdasarkan rumus 2.1. Hasil perhitungannya adalah sebagai berikut:

Availability = 1008 / (1008+15)

Availability = 1008 / 1023

Availability = 0.9853

Availability = 98.53 %

Dengan waktu tingkat *availability* sebesar 98.53 % dalam waktu 112 hari adalah hasil yang memuaskan dan sudah menjawab pertanyaan dari Batasan masalah yang diangkat yakni minimal tingkat *availability* sebesar 85 %.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan maka dapat ditarik kesimpulan sebagai berikut.

1. Saat terjadi peningkatan trafik yang masuk ke dalam sistem maka sistem mengalami peningkatan penggunaan resource yakni CPU dan Memori. Dari hasil pengujian terhadap 1 perangkat, sistem hanya dapat melayani 20 koneksi secara bersamaan. Saat jumlah koneksi melebihi 20 maka penggunaan resource akan melebihi batas aman yakni 80%. Dengan begitu peningkatan tersebut akan meningkatkan resiko kegagalan sistem.
2. Dari hasil pengujian sistem yang telah menerapkan *load balancing* didapatkan hasil bahwa sistem mampu menangani permintaan dengan total koneksi 110. Selebihnya *cluster server* akan melebihi batas tidak aman penggunaan *resource* yakni lebih dari 80%.
3. Sistem yang diusulkan mampu memberikan tingkat availabilitas sebesar 98.53 persen. Hal ini sudah memenuhi batasan permasalahan sebesar 85 persen %.

7.2 Saran

Berdasarkan penelitian yang sudah dilakukan berikut ini merupakan saran yang dapat digunakan pada penelitian lebih lanjut untuk sistem di UD Sendang Putra.

1. Perlu dilakukan kajian tambahan mengenai algoritma *load balancing* yang cocok untuk digunakan dan memiliki performa paling baik dalam mendistribusikan trafik.
2. Perlu dilakukan integrasi antar file *system* pada 2 *backend server* agar konsistensi file pada 2 *backend server* tetap terjaga.

DAFTAR REFERENSI

Anderson, M., 2017. *What is Load Balancing?*. [Online] Available at: https://www.digitalocean.com/community/tutorials/what-is-load-balancing [Accessed 18 Januari 2020].

Aryal, S., 2017. *Highly Available Web Service Using the Raspberry Pi Cluster.* Helsinki, Helsinki Metropolia University of Applied Sciences.

BERNSTEIN, P. A. & GOODMAN, N., 1985. Serializability Theory for Replicated Databases*. *JOURNAL OF COMPUTER AND SYSTEM SCIENCES,* Volume 31, pp. 355-374.

damasvara, 2010. *Optimum percentage of ram usage.* [Online] Available at: tomshardware.com/threads/optimum-percentage-of-ram-usage [Accessed 01 July 2020].

Data, M., Kartikasari, D. P. & Bhawiyuga, A., 2019. *The Design of High Availability Dynamic Web Server Cluster.* Lombok, IEEE.

Dawood, R., Qiana, S. F. & Muchallil, S., 2014. Kelayakan Raspberry Pi sebagai Web Server Perbandingan Kinerja Nginx, Apache, dan Lighttpd pada Platform Raspberry Pi. *Jurnal Rekayasa Elektrika,* 11(1), pp. 25-29.

Heidi, E., 2019. *What is High Availability?*. [Online] Available at: https://www.digitalocean.com/community/tutorials/what-is-high-availability [Accessed 18 Januari 2020].

Hidayat, E. W. & Rahmatulloh, A., 2014. Optimasi Server SIMAK Menggunakan Memcached dan Mirror Server Untuk Meningkatkan Kecepatan Akses Layanan Akademik Universitas Siliwangi. *Jurnal Ilmu Komputer dan Sains Terapan ,* 5(2).

J.Johnston, S. et al., 2018. Commodity single board computer clusters and their applications. *Future Generation Computer Systems ,* Volume 89, pp. 201-212.

Kozlovski, S., 2018. *A Thorough Introduction to Distributed Systems.* [Online] Available at: https://www.freecodecamp.org/news/a-thorough-introduction-to-distributed-systems-3b91562c9b3c/ [Accessed 19 Januari 2020].

Mozilla, 2019. *What is a web server?*. [Online] Available at: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server [Accessed 19 Januari 2020].

Nørv˚ag, K., 2000. *An Introduction to Fault-Tolerant Systems,* Trondheim: Department of Computer and Information Science Norwegian University of Science and Technology.

Rahmatulloh, A. & MSN, F., 2017. Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi. *Jurnal Nasional Teknologi dan Sistem Informasi* , 3(2), pp. 241-248.

Raspberry, n.d. *What is a Raspberry Pi?*. [Online] Available at: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/ [Accessed 13 April 2020].

Steen, M. v. & Tanenbaum, A. S., 2016. *A brief introduction to distributed systems.* Amsterdam, Springer.