TUGAS AKHIR - KI141502

RANCANG BANGUN SISTEM LOG SERVER BERBASIS SYSLOG DAN CASSANDRA UNTUK MONITORING PENGELOLAAN JARINGAN DI ITS

THIAR HASBIYA DITANAYA
NRP 5112 100 083

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2016

TUGAS AKHIR - KI141502

RANCANG BANGUN SISTEM LOG SERVER BERBASIS SYSLOG DAN CASSANDRA UNTUK MONITORING PENGELOLAAN JARINGAN DI ITS

THIAR HASBIYA DITANAYA
NRP 5112 100 083

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2016

UNDERGRADUATE THESIS - KI141502

DESIGN AND IMPLEMENTATION OF SYSLOG SERVER USING CASSANDRA TO MANAGE NETWORKS IN ITS

THIAR HASBIYA DITANAYA
NRP 5112 100 083

Supervisor I
Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD

Supervisor II
Ir. Muchammad Husni, M.Kom.

Department of INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2016

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM LOG SERVER BERBASIS SYSLOG DAN CASSANDRA UNTUK MONITORING PENGELOLAAN JARINGAN DI ITS

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur dan Jaringan Komputer
Program Studi S1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

THIAR HASBIYA DITANAYA
NRP: 5112 100 083

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD. .....
NIP: 197708242006041001 (Pembimbing 1)

Ir. Muchammad Husni, M.Kom. .....
NIP: 196002211984031001 (Pembimbing 2)

SURABAYA
JUNI 2016

RANCANG BANGUN SISTEM LOG SERVER BERBASIS SYSLOG DAN CASSANDRA UNTUK MONITORING PENGELOLAAN JARINGAN DI ITS

Nama	:	**THIAR HASBIYA DITANAYA**
NRP	:	**5112 100 083**
Jurusan	:	**Teknik Informatika FTIf**
Pembimbing I	:	**Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD**
Pembimbing II	:	**Ir. Muchammad Husni, M.Kom.**

Abstrak

Semakin berkembangnya teknologi informasi menuntut semakin banyaknya penggunaan perangkat berupa komputer atau perangkat jaringan. Komputer yang digunakan sebagai server harus memiliki spesifikasi yang tinggi. Server berspesifikasi tinggi terkadang masih belum bisa memenuhi permintaan yang sangat besar, sehingga dibutuhkan teknologi pembagian beban dan topologi jaringan yang handal. Teknologi pembagian beban dan topologi jaringan yang handal membutuhkan banyak perangkat komputer agar dapat berjalan dengan baik dan benar. Semakin banyak perangkat komputer yang digunakan akan membutuhkan biaya perawatan yang lebih besar, terutama ketika sistem administrator harus memeriksa pesan log semua perangkat komputernya.

Syslog server adalah sebuah server yang menyimpan data syslog berbagai macam perangkat komputer dan jaringan secara terpusat. Syslog server harus memiliki ketersediaan tinggi untuk melayani penyimpanan syslog setiap perangkat komputer dan jaringan.

Semakin banyak perangkat komputer dan jaringan yang harus dicatat lognya secara realtime menjadi tantangan

tersendiri. Log tersebut juga harus dapat ditampilkan kembali dengan baik, agar sistem administrator dapat menganalisa log dengan mudah dan cepat.

Muncul gagasan penggunaan Cassandra sebagai basis data penyimpanan syslog server. Cassandra sebagai penyimpanan utama syslog server didukung dengan nodeJS sebagai middleware yang digunakan untuk berinteraksi dengan cassandra diharapkan dapat menjadi sebuah lingkungan log server yang dapat menyimpan semua syslog secara terpusat. Namun dengan beragammnya perangkat komputer dan jaringan yang ada dibutuhkan standar pemformatan syslog yang dikirimkan sesuai rfc3164 dan rfc5424.

Muncul gagasan lain untuk mengembangkan penerjemah pesan syslog secara modular. Dengan bantuan pustaka syslogparser yang ditulis dalam bahasa GO, semua pesan syslog dari berbagai perangkat komputer dan jaringan dapat diterjemahkan dan di proses oleh middleware nodeJS.

Hasil membuktikan penggunaan Cassandra sebagai penyimpanan data Syslog dapat membantu proses pengembangan sistem menjadi skala yang lebih besar dengan cepat dan tanpa adanya penghentian pada sistem yang sedang berjalan, Sehingga sistem dapat menangani lebih banyak perangkat.

***Kata-Kunci**: Syslog, Cassandra, Scalability, NodeJS.*

DESIGN AND IMPLEMENTATION OF SYSLOG SERVER USING CASSANDRA TO MANAGE NETWORKS IN ITS

Name	:	**THIAR HASBIYA DITANAYA**
NRP	:	**5112 100 083**
Major	:	**Informatics FTIf**
Supervisor I	:	**Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD**
Supervisor II	:	**Ir. Muchammad Husni, M.Kom.**

Abstract

The continued development of information technologies require higher usage of the computer or network device. Computers are used as servers should have a high specification. High specification servers sometimes still can not meet the huge demand, so it takes a load balance technology and reliable network topology. But it require a lot of computer to run properly. The more devices that use computers will require greater maintenance costs, for example system administrator must check log message from all computers.

Syslog server is a centralized server that stores syslog message from computing devices and network device. Syslog server must have high availability to serve as storage syslog.

There is an idea to use cassandra as storage database for syslog message and NodeJS as middleware to communicate. However, it hard to translate syslog message without proper parser function.

To overcome this problem, there is a syslog parser written in GO to translate syslog message that formatted with rfc3164 and rfc5424. Using syslog parser to translate syslog message and process it with NodeJS middleware before store it in Cassandra.

The results prove the use Cassandra as Syslog data storage

system can help the process of developing into a larger scale quickly and without downtime of the current system, so the system can handle more devices
***Kata-Kunci**:Syslog, Cassandra, Scalability, NodeJS.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BAB 1

PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dam batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Semakin berkembangnya teknologi informasi menuntut semakin banyaknya penggunaan perangkat berupa komputer atau perangkat jaringan. Komputer yang digunakan sebagai server harus memiliki spesifikasi yang tinggi. Server berspesifikasi tinggi terkadang masih belum bisa memenuhi permintaan yang sangat besar, sehingga dibutuhkan teknologi pembagian beban dan topologi jaringan yang handal. Teknologi pembagian beban dan topologi jaringan yang handal membutuhkan banyak perangkat komputer agar dapat berjalan dengan baik dan benar. Semakin banyak perangkat komputer yang digunakan akan membutuhkan biaya perawatan yang lebih besar, terutama ketika sistem administrator harus memeriksa pesan *log* semua perangkat komputernya.

Syslog server adalah sebuah server yang menyimpan data syslog berbagai macam perangkat komputer dan jaringan secara terpusat. Syslog server harus memiliki ketersediaan tinggi untuk melayani penyimpanan syslog setiap perangkat komputer dan jaringan.

Semakin banyak perangkat komputer dan jaringan yang harus dicatat lognya secara *realtime* menjadi tantangan tersendiri. *log* tersebut juga harus dapat ditampilkan kembali dengan baik, agar sistem administrator dapat menganalisa *log* dengan mudah dan cepat.

Perancangan syslog server yang memiliki ketersediaan tinggi harus menggunakan teknik penyimpanan dan topologi jaringan

yang baik. Syslog server harus memiliki toleransi ketika terjadi kegagalan pada salah satu perangkat dalam topologi jaringannya, sehingga syslog server tetap dapat melayani permintaan penyimpanan dari berbagai perangkat komputer dan jaringan.

Oleh karena itu dibangunlah sistem ini, dengan menerapkan penyimpanan *realtime* yang memiliki ketersediaan tinggi pada syslog server diharapkan dapat melayani setiap permintaan pencatatan *log* secara *realtime* dan menampilkan kembali *log* tersebut kepada sistem administrator secara mudah dan cepat.

1.2 Rumusan Masalah

Berikut beberapa hal yang menjadi rumusan masalah dalam tugas akhir ini:

1. Bagaimana membangun sistem penyimpanan syslog secara terpusat ?
2. Bagaimana membangun sistem penyimpanan syslog yang memiliki ketersediaan tinggi ?
3. Bagaimana menyajikan data syslog dengan baik untuk keperluan analisis ?

1.3 Batasan Masalah

Dari permasalahan yang telah diuraikan di atas, terdapat beberapa batasan masalah pada tugas akhir ini, yaitu:

1. Perangkat yang dapat ditangani adalah perangkat yang mendukung protokol syslog.
2. Perangkat di lingkungan jaringan ITS.

1.4 Tujuan

Tugas akhir dibuat dengan beberapa tujuan. Berikut beberapa tujuan dari pembuatan tugas akhir:

1. Mampu mengimplementasikan sistem pencatatan syslog yang terpusat.
2. Mampu mengimplementasikan sistem pencatatan syslog yang memiliki ketersediaan tinggi.
3. Mampu menyajikan data syslog dengan baik untuk keperluan analisis.

1.5 Manfaat

Dengan dibangunnya sistem pencatatan syslog ini diharapkan sistem administrator dapat mengetahui kondisi semua perangkat yang dikelolanya dengan mudah dan cepat untuk keperluan analisis dan pengambilan keputusan.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

- **Penyusunan Proposal Tugas Akhir**
  Penyusunan proposal Tugas Akhir dilaksanakan untuk merumuskan masalah serta melakukan penetapan desain dasar sistem yang akan dikembangkan dalam pelaksanaan Tugas Akhir ini.
- **Studi Literatur**
  Untuk membantu proses pengerjaan Tugas Akhir, diperlukan studi lebih lanjut mengenai penggunaan komponen-komponen terkait dengan sistem yang akan dibangun.
- **Desain dan Perancangan**
  Dalam rangka memerinci lebih jauh mengenai bagaimana memanfaatkan komponen-komponen sistem untuk membangun sistem secara utuh, diperlukan proses desain dan perancangan dari sistem. Hasil analisis dan desain

kemudian ditetapkan menjadi rancangan dasar implementasi sistem.

- **Implementasi Sistem**
  Hasil analisis dan desain kemudian diimplementasikan melalui komponen-komponen perangkat lunak pendukung.
- **Uji Coba dan Evaluasi**
  Untuk mengukur kemampuan sistem untuk menangani pengguna dari segi fungsionalitas dan performa, dilakukan proses uji coba dan evaluasi untuk mengetahui sejauh mana sistem dapat berjalan sesuai dengan yang diharapkan.

1.7 Sistematika Laporan

Buku Tugas Akhir ini terdiri dari beberapa bab yang dijelaskan sebagai berikut:

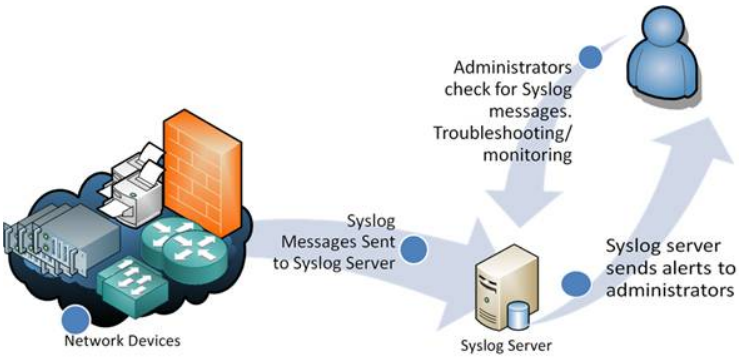
- **Bab 1. Pendahuluan**. Bab ini berisikan latar belakang, rumusan masalah, tujuan, manfaat, metodologi dan sistematika penulisan yang digunakan sebagai dasar penyusunan Tugas Akhir.
- **Bab 2. Tinjauan Pustaka**. Bab ini berisikan teori penunjang yang berhubungan dengan Tugas Akhir.
- **Bab 3. Desain dan Perancangan**. Bab ini membahas desain dasar dari sistem dan perangkat lunak yang akan dirancang sebagai bagian dari pengerjaan Tugas Akhir.
- **Bab 4. Implementasi**. Bab ini membahas hasil implementasi dari rancangan sistem beserta tekniknya.
- **Bab 5. Pengujian dan Evaluasi**. Bab ini membahas mengenai teknik uji coba dan hasil keluarannya sebagai bahan evaluasi terhadap hasil Tugas Akhir.
- **Bab 6. Penutup**. Bab ini berisi kesimpulan dari hasil uji coba serta saran untuk pengembangan Tugas Akhir selanjutnya.

BAB 2

TINJAUAN PUSTAKA

2.1 Syslog

Syslog adalah sebuah standard untuk message logging yang digunakan untuk mengkategorikan *log* file yang dilaporkan oleh aplikasi atau sistem operasi. Syslog dapat mengkategorikan laporan-laporan tersebut berdasarkan *facility* dan *severity* level [1].



Gambar 2.1: Contoh Penggunaan Syslog Protocol

2.2 Node JS

Merupakan sebuah *platform* yang dibangun di atas Chrome's JavaScript runtime dengan teknologi V8 yang mendukung proses server yang bersifat long-running. Tidak seperti platform modern yang mengandalkan multithreading, NodeJS memilih menggunakan asynchronous I/O eventing. Karena inilah NodeJS mampu bekerja dengan konsumsi memori rendah [2][3].

Teknologi yang tidak memanfaatkan *multi-thread* ini memudahkan pengembang yang terkadang kesulitan mengatur sumber daya yang digunakan *thread*. Karena tidak mungkin ada

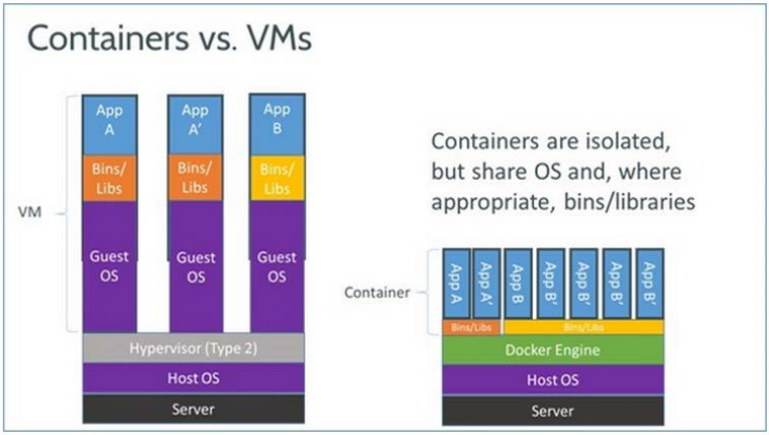
sumber daya yang terkunci karena thread yang berjalan. Akhirnya banyak yang memanfaatkan kemampuan dasar NodeJS sebagai web server atau layanan REST API. Dengan adanya callback untuk setiap penggunaan fungsi, memungkinkan setiap pemanggilan fungsi yang tidak menghasilkan apapun, NodeJS akan *sleep*.

2.3 Angular JS

Angular JS adalah kerangka kerja *frontend* pada aplikasi berbasis web. Angular JS menggunakan bahasa javascript dan berjalan pada peramban. Angular JS menggunakan pola desain MVW (*Model View Whatever*) yang membuatnya lebih fleksibel ketika digunakan. Angular JS membantu dalam pembangunan halaman HTML menjadi lebih dinamis. Merupakan sebuah kumpulan alat bantu yang mampu bekerja baik dengan pustaka lainnya. Setiap fitur dapat dimodifikasi sesuai dengan kebutuhan aplikasi. Menjadi salah satu kerangka kerja yang memfasilitasi pembangunan aplikasi kompleks yang terorganisir dan mudah dirawat. Angular JS lebih dikenal pada kemampuannya melayani sebuah situs web yang menggunakan halaman tunggal untuk penyajian data yang beragam [6][7].

2.4 Docker

Docker adalah virtualisasi menggunakan teknologi linux kontainer. Docker berjalan pada lapisan kernel, sehingga memungkinkan untuk menjalankan lebih banyak virtualisasi dengan sumber daya yang terbatas.Semua virtualisasi docker berjalan pada sistem operasi yang sama dan saling berbagi sumber daya. Dengan menggunakan docker dapat mempermudah proses *deployment*. Docker sangat populer digunakan dalam layanan-layanan berbasis *cloud* [8][9].



Gambar 2.2: Perbandingan docker dan virtual machine

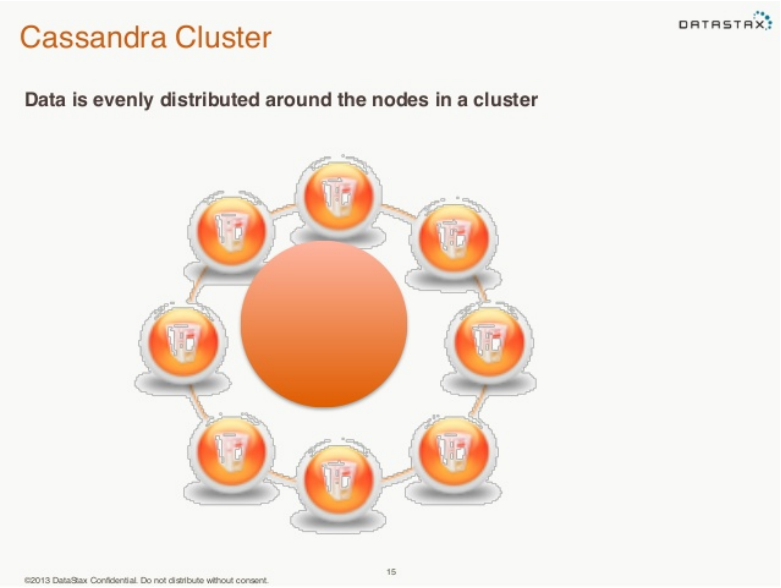
Docker dapat dijalankan di berbagai sistem operasi, pengembang dapat dengan mudah menggunakan layanan docker melalui `https://hub.docker.com` untuk mengunduh *images* yang diinginkan.

2.5 Cassandra

Apache Cassandra merupakan basis data salah satu NoSQL (Not only SQL) terkenal yang dirancang untuk mendukung penyimpanan data dalam ukuran besar. Beberapa fitur yang diunggulkan oleh apache cassandra antara lain :

1. *Decentralized*
   Semua node pada klaster cassandra memiliki tugas dan kemampuan yang sama oleh karena itu tidak cassandra tidak memiliki SPOF (*Single Point Of Failure*) .
2. *Scalability*
   Kemampuan sistem dapat ditingkatkan sewaktu-waktu dengan menambah perangkat baru tanpa ada *downtime* atau gangguan pada aplikasi.

3. *Fault-tolerant*
   Kegagalan pada salah satu *node* tidak akan menggangu kinerja sistem, dan *node* tersebut dapat digantikan dengan *node* baru tanpa terjadi *downtime* atau gangguan pada aplikasi.
4. *Query language*
   Cassandra mendukung *query language* seperti pada basis data relasional menggunakan CQL(Cassandra Query Language)[5].



Gambar 2.3: Contoh Cluster Cassandra

2.6 Mysql

Mysql adalah basis data relasional dengan kemampuan *multithread* dan *multi-user*. Mysql mendukung transaksi pada basis data relasional. Transaksi ini digunakan untuk memastikan

serangkaian perintah berhasil dijalankan , jika ada satu dari serangkaian perintah yang gagal dijalankan maka mysql akan melakukan *roolback*. Kemampuan mysql untuk melakukan *roolback* sangat dibutuhkan pada sistem yang memiliki desain transaksional. Mysql adalah perangkat lunak sumber terbuka dibawah lisensi GPL, sehingga dapat digunakan secara bebas.

2.7 Socket.IO

Socket.IO adalah pustaka dalam bahasa javascripts untuk mendukung komunikasi *realtime* pada aplikasi web. Socket.IO berjalan pada sisi *client* dan *server*. Pada sisi *client* Socket.IO berjalan di peramban milik pengguna, sedangkan pada sisi *server* Socket.IO berjalan sebagai pustaka Node JS. Socket.IO menggunakan teknologi *websocket* sebagai media transportasinya. *Websocket* adalah teknologi web yang memungkinkan *client* dan *server* melakukan pertukaran data secara *realtime*, teknologi ini memungkinkan *server* memulai pengiriman data terlebih dahulu tanpa ada permintaan dari *client*. Teknologi *websocket* sering digunakan sebagai *push notification*.

2.8 Golang

Golang adalah bahasa pemrogramman dengan lisensi sumber terbuka yang dikembangkan oleh Google. Golang termasuk bahasa *statically typed* seperti C. Golang akan di kompilasi menjadi bahasa sistem terlebih dahulu, sehingga performa bahasa Golang lebih baik dan cepat dari pada bahasa yang menggunakan *interpreter* seperti php,ruby dan python. Walaupun Golang termasuk bahasa *statically typed*, Golang sudah mendukung fitur-fitur seperti pada bahasa pemgrogramman yang lebih maju. Golang mendukung fitur-fitur antara lain *garbage collection* ,*memory safety* dan *concurrent programming* .

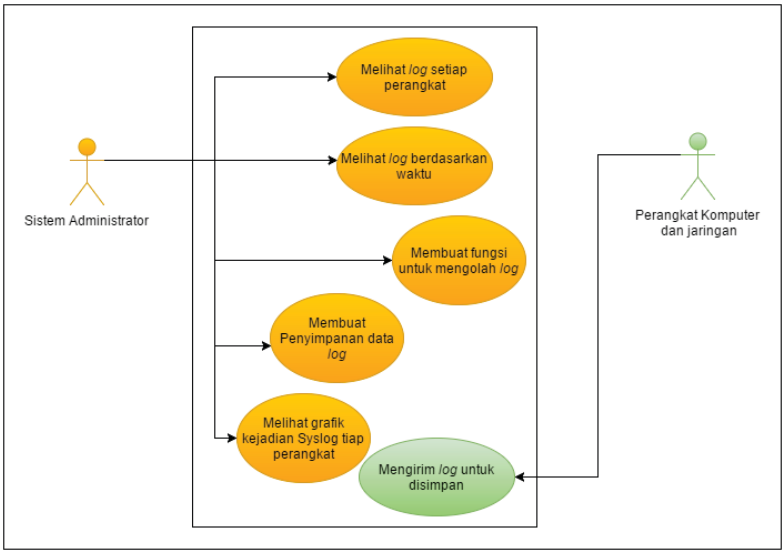
BAB 3

DESAIN DAN PERANCANGAN

Pada bab ini dibahas mengenai analisis dan perancangan sistem.

3.1 Kasus Penggunaan

Terdapat dua aktor dalam sistem yaitu sistem administrator dan perangkat komputer dan jaringan. Sistem administrator adalah aktor yang mengoperasikan sistem, sedangkan perangkat komputer dan jaringan adalah aktor yang menggunakan layanan sistem. Diagram kasus penggunaan menggambarkan kebutuhan-kebutuhan yang harus dipenuhi sistem. Diagram kasus penggunaan digambarkan pada Gambar 3.1.



Gambar 3.1: Digram Kasus Penggunaan

Digram kasus penggunaan pada Gambar 3.1 dideskripsikan masing-masing pada Tabel 3.1.

Tabel 3.1: Daftar Kode Kasus Penggunaan

Kode Kasus Penggunaan	**Nama Kasus Penggunaan**	**Keterangan**
UC-0001	Melihat *log* setiap perangkat.	Sistem administrator dapat melihat *log* dari semua perangkat yang telah didaftarkan.
UC-0002	Melihat *log* berdasarkan waktu.	Sistem administrator dapat melihat *log* dari semua perangkat yang telah didaftarkan berdasarkan waktu.
UC-0003	Membuat fungsi untuk mengolah *log*.	Sistem administrator dapat membuat fungsi baru untuk mengolah data *log*.
UC-0004	Membuat penyimpanan *log* data.	Sistem administrator dapat membuat penyimpanan data *log* baru.
UC-0005	Melihat grafik kejadian Syslog tiap perangkat.	Sistem administrator dapat melihat grafik kejadian Syslog tiap perangkat berdasarkan waktu.
UC-0006	Mengirim *log* untuk disimpan.	Perangkat komputer dan jaringan mengirimkan syslog ke sistem sesuai format rfc3164 dan rfc5424.

3.2 Arsitektur Sistem

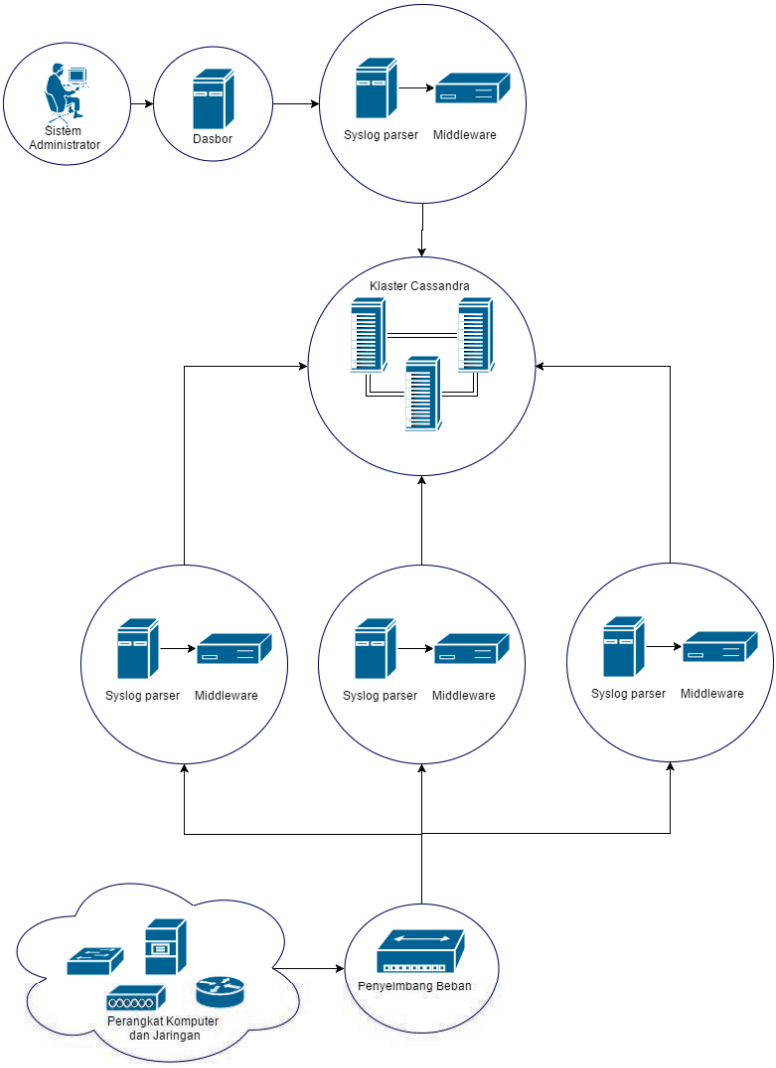
Pada sub-bab ini, dibahas mengenai tahap analisis dan desain dari sistem yang akan dibangun. Pembahasan akan dilakukan pada tiap-tiap komponen yaitu: penyeimbang beban, parser, middleware, penyimpanan data Cassandra dan Dasbor.

3.2.1 Desain Umum Sistem

Sistem *log* server yang akan dibangun terdiri dari beberapa komponen yang telah disusun secara modular. Komponen-komponen yang dibangun antara lain:

1. Penyeimbang Beban
   Penyeimbang beban akan membagi beban ke komputer pekerja secara *round-robin*.
2. Penyimpanan Data
   Penyimpanan data adalah sebuah klaster basis data yang befungsi sebagai penyimpanan terpusat semua pesan syslog yang diterima dari perangkat komputer dan jaringan.
3. Middleware
   Middleware adalah penghubung antara dasbor dengan penyimpanan data, atau parser dengan penyimpanan data. Middleware berfungsi sebagai perantara untuk penyimpanan atau pengambilan data.
4. Dasbor
   Dasbor adalah halaman yang digunakan sistem administrator untuk mengelola dan menampilkan data syslog.
5. Parser
   Parser dibangun secara terpisah dan tidak dapat langsung menyimpan data kedalam Cassandra.

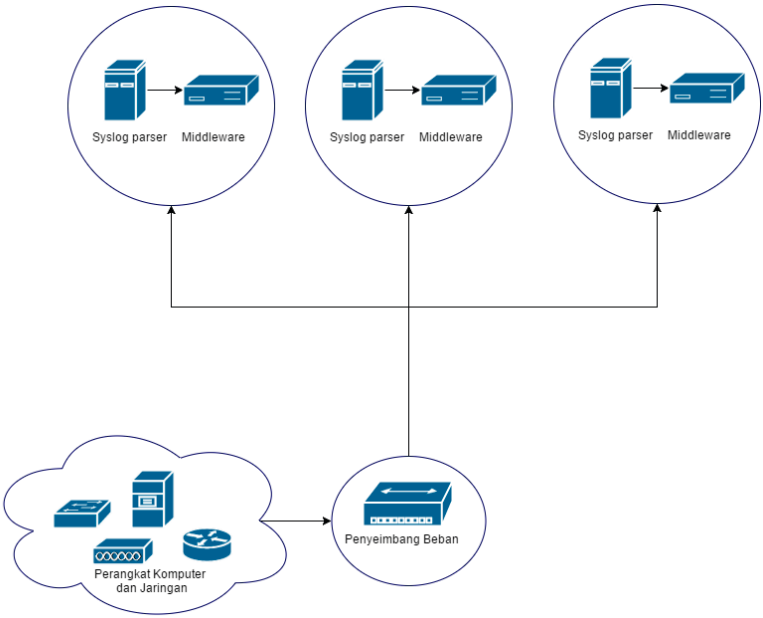
Arsitektur sistem secara umum digambarkan pada diagram arsitektur 3.2.



Gambar 3.2: Desain Sistem Secara Umum

3.2.2 Desain Penyeimbang Beban

Untuk membuat sistem yang memiliki ketersediaan tinggi. Sistem harus mendukung skalabilitas. Skalabilitas dibutuhkan untuk memperbesar sistem ketika sistem menerima permintaan yang terlalu besar. Skalabilitas dilakukan dengan memperbanyak Syslog parser dan Middleware. Ketika Syslog parser dan middleware di perbanyak, maka dibutuhkan penyeimbang beban untuk membagi beban. Penyeimbang beban di letakkan di depan Syslog parser. Penyeimbang beban yang akan di akses komputer dan perangkat jaringan secara langsung. Penyeimbang beban akan membagi permintaan ke masing-masing Syslog parser secara bergantian. Diagram arsitektur sistem dengan penyeimbang beban tertera pada gambar 3.3



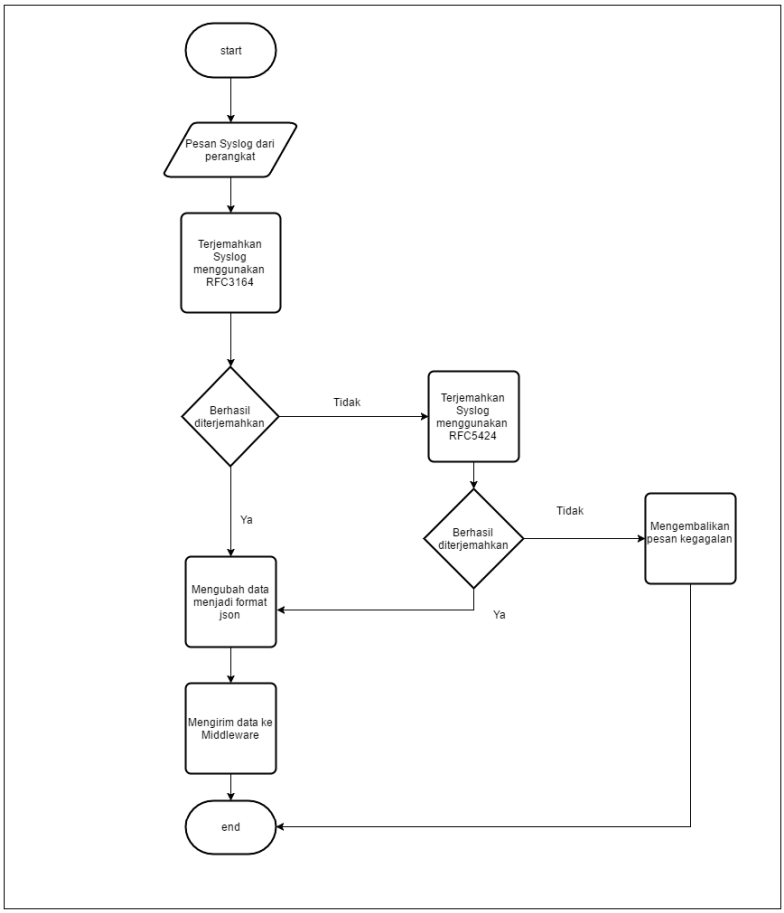
Gambar 3.3: Desain Arsitektur Penyeimbang Beban

3.2.3 Desain Syslog Parser

Syslog parser adalah aplikasi modular yang digunakan untuk menerjemahkan pesan Syslog. Syslog parser dibangun menggunakan bahasa GO. Untuk membangun parser yang dapat menerima pesan Syslog dari berbagai perangkat, maka parser harus dapat berjalan menggunakan protokol *User Datagram Protocol(UDP)* dan *Transmission Control Protocol(TCP)*. Syslog parser harus berjalan menggunakan protokol UDP dan TCP karena perangkat-perangkat komputer mengirimkan pesan syslog nya menggunakan protokol TCP sedangkan preangkat jaringan menggunakan UDP sebagai protokol pengirimannya. Syslog parser akan berjalan pada lingkungan Docker sebagai layanan. Syslog parser menerima pesan Syslog dari perangkat komputer dan jaringan. Pesan Syslog yang diterima akan diterjemahkan menjadi kata kunci dan nilai dalam format JSON. Data yang telah di format dalam JSON akan diteruskan ke *middleware* untuk disimpan. Diagram arsitektur Syslog parser tertera pada gambar 3.4.

Gambar 3.4: Desain Arsitektur Syslog Parser

Syslog parser memiliki beberapa tahapan dalam penerjemahan pesan syslog. Tahapan-tahapan proses penerjemahan pesan Syslog di gambarkan menggunakan diagram alur kerja. Diagram alur kerja Syslog parser tertera pada gambar 3.5.

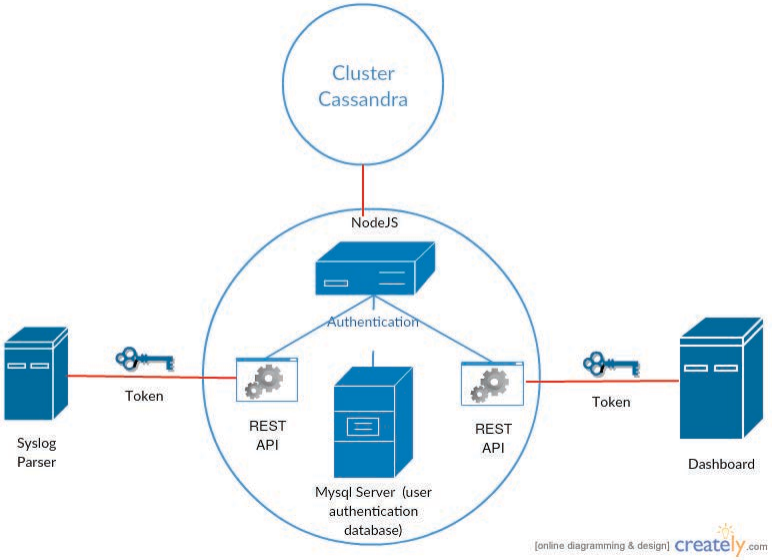


Gambar 3.5: Diagram Alur Penerjemahan Pesan Syslog

3.2.4 Desain Middleware

Middleware adalah penghubung antara dasbor dengan penyimpanan data atau parser dengan penyimpanan data. Middleware berfungsi sebagai perantara untuk penyimpanan atau pengambilan data. Middleware berupa layanan REST API yang

dibangun menggunakan NodeJS sebagai REST API server dan menggunakan JSON(*Javascript Object Notation*) sebagai pemformatan data. Middleware menggunakan token sebagai salah satu faktor untuk melakukan autentikasi. Autentikasi diperlukan untuk membatasi akses pada Middleware. Digram arsitektur Middleware tertera pada Gambar 3.6.



Gambar 3.6: Desain Arsitektur Middleware

REST API menggunakan rute-rute pada URI untuk melakukan komunikasi dan pertukaran data. Karena REST API bersifat *stateless*, maka REST API tidak mendukung penyimpanan sesi pengguna. Oleh karena itu autentikasi dan autorisasi pengguna dilakukan menggunakan token. Token adalah informasi-informasi terkait pengguna dan sumber daya yang dapat dikelolanya. Token akan dikodekan sebelum diberikan kepada klien. Hal ini dimaksudkan agar klien tidak mengetahui isi token dan mengurangi kemungkinan pemalsuan

token. Daftar rute yang digunakan dalam REST API *middleware* dijelaskan pada Tabel 3.2

Tabel 3.2: Rute pada REST API Middleware

No	Rute	Metode	Aksi
1	/parser/:token/:db	GET	Mendapatkan data dari penyimpanan data sesuai dengan token dan nama penyimpanan data
2	/parser/:token/:db	POST	Memasukkan data kedalam penyimpanan data sesuai dengan token dan nama penyimpanan data
3	/storage/	GET	Mendapatkan daftar nama penyimpanan data yang ada.
4	/storage/	POST	Membuat penyimpanan data baru
5	/storage/	PUT	Memperbarui penyimpanan data
6	/function/plugin/	GET	Mendapatkan daftar *plugin* yang telah dibuat dan tersimpan sebelumnya.
7	/function/plugin/:name	GET	Menampilkan kode *plugin* sesuai dengan nama yang disertakan pada *URI*.

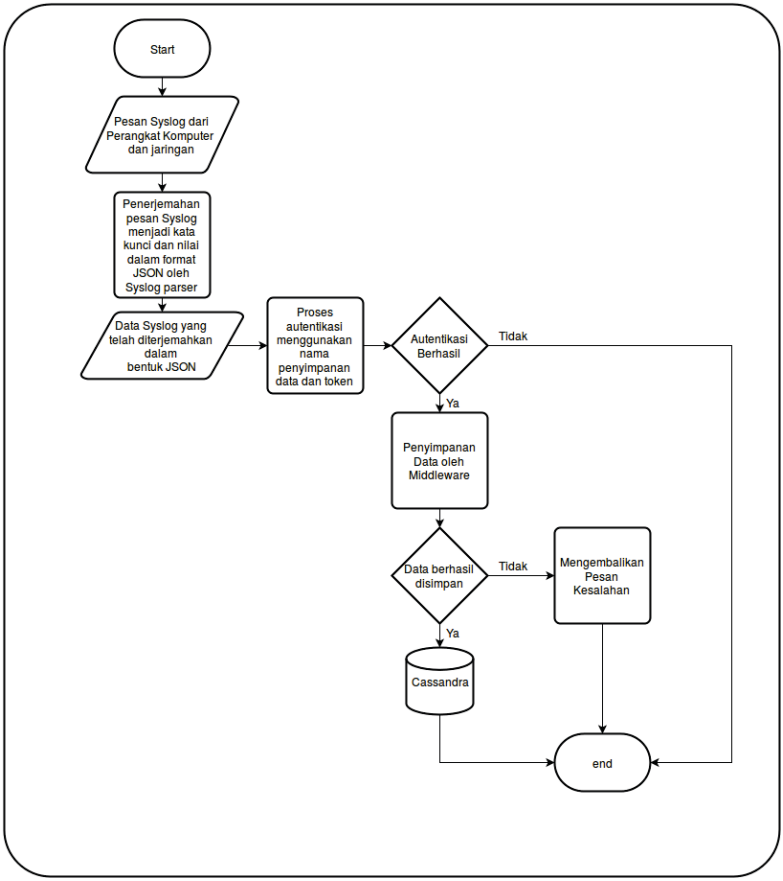
Tabel 3.2: Rute pada REST API Middleware

No	Rute	Metode	Aksi
8	/function/	PUT	Menyimpan kode *plugin* dan *Query Profile* yang telah dibuat di menu fungsi pada aplikasi Dasbor, kemudian menjalankan kode tersebut dengan data yang telah diambil berdasarkan *Query Profile* dalam lingkungan yang terisolasi dan mengembalikan keluaran dari kode tersebut.
9	/function/query	GET	Mendapatkan daftar *Query Profile* yang telah dibuat dan tersimpan sebelumnya.
10	/function/query/:name	GET	Menampilkan *Query Profile* sesuai dengan nama yang disertakan pada *URI*.

3.2.4.1 Desain Metode Penulisan Data Syslog

Data dari Syslog parser akan diterima melalui rute `/parser/:token/:db`. Kemudian dilakukan proses autentikasi oleh pengendali parser. Jika proses autentikasi berhasil, maka data akan disimpan pada penyimpanan data Cassandra. Gambar

diagram alur kerja penulisan data syslog tertera pada gambar 3.7.

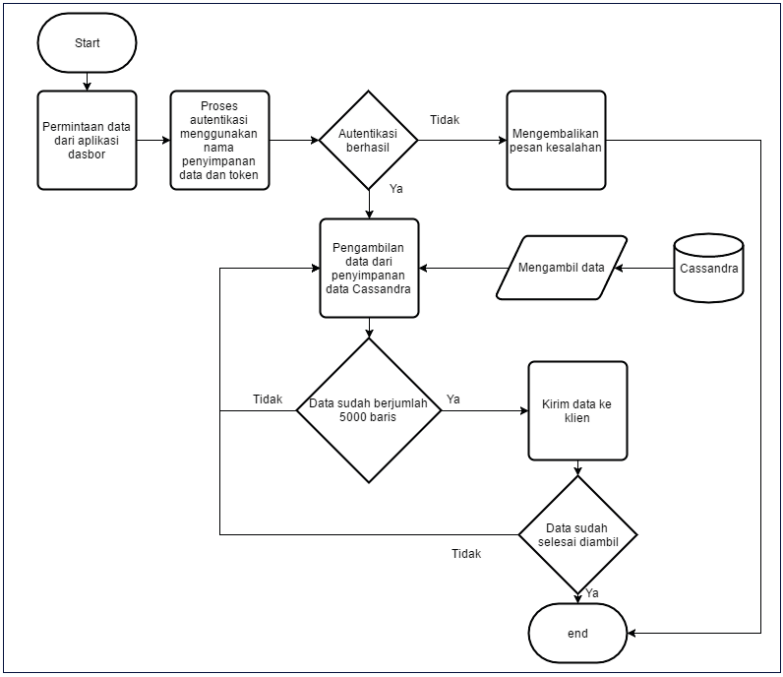


Gambar 3.7: Diagram Alur Penulisan Data Syslog

3.2.4.2 Desain Metode Pengambilan Data Syslog

Cassandra tidak mendukung pengambilan data menggunakan filter tertentu layaknya yang di lakukan pada basis data relasional. Hal ini bertujuan agar pengambilan data menjadi lebih cepat. Data

yang di dapat dari Cassandra akan disimpan pada variabel. Ketika variabel telah berisi 5000 baris data, data akan dikirimkan ke klien menggunakan Socket.IO. Klien hanya melakukan sekali *request* saja. Kemudian pengiriman data dilakukan secara bertahap hingga seluruh data selesai dikirimkan. Diagram alur kerja pengambilan data tertera pada gambar 3.8.

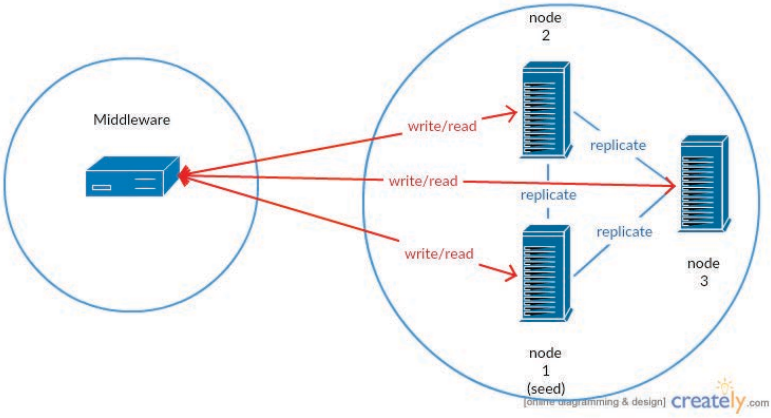


Gambar 3.8: Diagram Alur Pengambilan Data Syslog

3.2.5 Desain Penyimpanan Data

Penyimpanan data adalah sebuah lingkungan basis data yang befungsi sebagai penyimpanan terpusat semua pesan *log*. Penyimpanan data berupa sebuah klaster Cassandra yang telah terintegrasi. Klaster Cassandra menggunakan *Consistency level*

quorum sehingga dibutuhkan setidaknya n/2 + 1 *node* menyala untuk dapat beroperasi. Jumlah *node* Cassandra yang akan digunakan adalah 3. Penyimpanan data hanya dapat diakses oleh *middleware*, sehingga dapat mengurangi tingkat pencurian data. Penyimpanan data menerima masukan data berupa perintah CQL (*Cassandra Query Language*). Skema penyimpanan data dapat bertambah sesuai definisi yang ditentukan oleh sistem administrator. Pembuatan skema dapat dilakukan menggunakan aplikasi dasbor. Skema yang dibuat akan didistribusikan ke semua *node* dalam klaster. Ketika terdapat *node* baru yang akan bergabung, *node* tersebut akan mengkopi semua skema yang sudah ada sebelumnya pada klaster. Diagram arsitektur penyimpanan data tertera pada Gambar 3.9.



Gambar 3.9: Desain Arsitektur Penyimpanan Data

3.2.6 Desain Dasbor

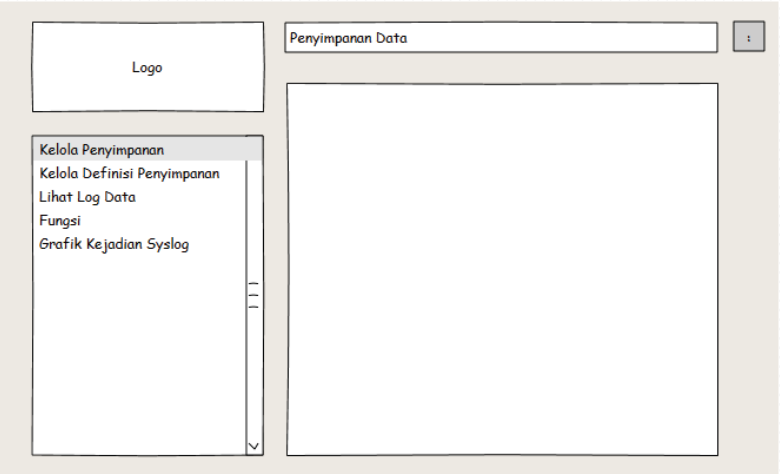
Dasbor adalah halaman yang digunakan sistem administrator untuk mengelola dan menampilkan data *log*. Dasbor adalah aplikasi berbasis web yang dibangun menggunakan NodeJS

sebagai *web server* dan AngularJS sebagai tampilan depan halaman (*frontend*). Halaman depan menggunakan Angular Material untuk mendapatkan tampilan yang sederhana dan nyaman digunakan. Dasbor digunakan oleh sistem administrator untuk berinteraksi dengan sistem. Dasbor memiliki menu-menu yang memudahkan pengelolaan sistem. Menu-menu yang terdapat pada dasbor antara lain:

- Kelola Penyimpanan
  Menu yang digunakan sistem administrator untuk mengelola penyimpanan data Cassandra. Pengelolaan antara lain membuat dan mengedit penyimpanan data. Pembuatan penyimpanan data baru menyertakan nama basis data penyimpanan data dan nama pengembang. Kemudian sistem akan membuatkan penyimpanan data baru sesuai dengan nama penyimpanan data yang disertakan dan membuatkan token.
- Kelola Definisi Penyimpanan
  Menu yang digunakan sistem administrator untuk mengelola definisi penyimpanan data Cassandra. Definisi yang dikelola terkait nama kolom dan tipe kolom.
- Lihat Log Data
  Menu yang digunakan sistem administrator untuk melihat data *log* yang tersimpan pada penyimpanan data. Data *log* ditampilkan sesuai penyimpanan data dan rentang waktu yang dipilih.
- Grafik Kejadian Syslog
  Menu yang digunakan sistem administrator untuk melihat kejadian syslog dalam bentuk grafik. Kejadian dikelompokkan berdasarkan *severity*. Kejadian ditampilkan berdasarkan tanggal pada bulan yang dipilih.
- Fungsi
  Menu yang digunakan sistem administrator untuk menjalankan fungsi baru. Fungsi baru tersebut digunakan

untuk mengolah data syslog. Fungsi yang dibuat harus menggunakan bahasa javascripts.

Desain tampilan antar muka dasbor tertera pada Gambar 3.10.



Gambar 3.10: Desain Antar Muka Tampilan Dasbor

BAB 4

IMPLEMENTASI

Bab ini membahas implementasi sistem *log* Server secara rinci. Pembahasan dilakukan secara rinci untuk setiap komponen yang ada yaitu: penyeimbang beban, parser, middleware, penyimpanan data Cassandra dan Dasbor. Implentasi dilakukan pada lingkungan komputer dan virtual. Penjelasan lingkungan implementasi akan dijelaskan kemudian.

4.1 Lingkungan Implementasi

Lingkungan implementasi dan pengembangan dilakukan menggunakan virtualisasi Proxmox dengan spesifikasi *Host* komputer adalah Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz dengan memori 8 GB di Laboratorium Arsitektur dan Jaringan Komputer, Teknik Informatika ITS dan komputer fisik dengan spesifikasi Intel(R) Core(TM) i3-2120 CPU @ 3.20GHz dengan memori 4 GB di lantai 4 Teknik Informatika ITS. Perangkat lunak yang digunakan dalam pengembangan adalah sebagai berikut :

- Sistem Operasi Linux Ubuntu Server 14.04.2 LTS
- Editor teks nano
- Editor teks Sublime Text 2
- Golang versi 1.2.1 untuk pengembangan parser
- Git versi 1.9.1 untuk pengolahan versi program
- TortoiseGit versi 1.8.12.0 sebagai antar muka git
- NodeJS versi 4.2.3 untuk pengembangan aplikasi
- Express versi 4.13.1 untuk web server halaman admin
- Angular JS versi 1.5.0
- Python versi 2.7 untuk pembuatan penyeimbang beban
- Cassandra versi 3.0 sebagai penyimpanan data
- Mysql server versi 5.5
- Paket Latex untuk pembuatan buku tugas akhir
- Editor teks Texstudio untuk menjalankan paket Latex

- Peramban *web* Mozilla Firefox dan pustaka Firebug

4.2 Implementasi Penyeimbang Beban

Penyeimbang beban dibangun menggunakan Python 2.7. Penyeimbang beban membagi beban kepada komputer-komputer pekerja secara bergantian. Komputer-komputer pekerja telah di definisikan sebelummnya pada kode program penyeimbang beban. Kode sumber penyeimbang beban tertera pada gambar B.18.

4.3 Implementasi syslog parser

Syslog parser dibangun menggunakan bahasa GO, sehingga membutuhkan paket GO Lang sebagai kompilator. Syslog parser terdiri dari 2 aplikasi yaitu parser menggunakan TCP dan parser menggunakan UDP. Syslog parser akan dipasang pada server dengan alamat IP `10.151.38.181` dan berjalan pada *port* 9000 baik TCP maupun UDP.

4.3.1 Pembangunan Syslog parser penerjemah pesan Syslog

Pesan Syslog yang dikirmkan memiliki format sesuai dengan RFC 5424 dan RFC 3164. Format pesan Syslog adalah |HEADER|MESSAGE|. HEADER terdiri dari PRI, VERSION, TIMESTAMP, HOSTNAME, PROCID, MSGID dan APP-NAME. PRI terdiri dari gabungan kode *severity* dan *facility* yang telah dikodekan. VERSION adalah versi dari Syslog yang digunakan. TIMESTAMP adalah waktu dikirimnya pesan Syslog tersebut. HOSTNAME adalah nama dari komputer yang mengirimkan pesan Syslog. APP-NAME adalah nama aplikasi yang mengirimkan pesan Syslog.PROCID dan MSGID adalah penomoran yang digunakan sebgai penanda pesan dan penanda kelanjutan pesan Syslog. Kemudian MESSAGE adalah isi dari

pesan Syslog yang dikirmkan. Berikut contoh pesan Syslog yang dikirimkan menggunakan RFC 5424 `<34>1 2003-10-11T22:14:15.003Z mymachine.example.com su - ID47 - BOM'su root' failed for lonvick on /dev/pts/8` dan RFC 3164 `<34>Oct 11 22:14:15 mymachine su: 'su root' failed for lonvick on /dev/pts/8`. Perbedaan antara RFC 5424 dan RFC 3164 sebagai berikut:

- RFC 5424 mendukung pengiriman struktur data sedangkan RFC 3164 tidak mendukung.
- RFC 5424 menggunakan pemformatan waktu sesuai format ISO 8601 sedangkan RFC 3164 tidak menggunakan pemformatan waktu.

Syslog parser bekerja dengan cara menerjemahkan pesan Syslog yang telah di format ke dalam bentuk RFC 5424 atau RFC 3164 menjadi kata kunci dan nilai. Rancangan alur kerja Syslog parser dijelaskan pada *pseudocode* 4.1.

```
data <- pesan syslog dari perangkat
data <- Terjemahkan menggunakan RFC3164
    IF tidak berhasil diterjemahkan
        data <- Terjemahkan menggunakan RFC5424
        IF tidak berhasil diterjemahkan
            Kembalikan pesan kegagalan
            Exit
data <- Ubah data ke dalam format JSON
Kirim data ke middleware
```

Gambar 4.1: Pseudocode Alur Kerja Syslog parser

Pembangunan Syslog parser dilakukan sesuai langkah-langkah berikut:

- Unduh pustaka `syslogparser` menggunakan perintah `go get github.com/jeromer/syslogparser/rfc3164` dan `go get github.com/jeromer/syslogparser/rfc5424`. Pustaka yang diunduh akan masuk ke dalam direktori yang

sudah diatur pada GOPATH.

- Setelah pustaka berhasil diunduh, gunakan pustaka tersebut dan beberapa pustaka penunjang lainnya menggunakan perintah *import* sesuai dengan Kode sumber B.5.
- Setelah pustaka berhasil dimasukkan, struktur data untuk menyimpan variabel dan nilai dari pesan Syslog harus dibuat, struktur data ini digunakan untuk membentuk data *JSON* yang nantinya akan dikirim ke Middleware menggunakan protokol *HTTP REST API*. Pembuatan struktur data dapat dilakukan seperti pada kode program B.6
- Setelah pembentukan struktur data selesai, untuk melakukan penerjemahan data syslog cukup membuat objek `syslogparser` menggunakan pustaka `syslogparser` yang telah dimasukkan sebelumnya. Setelah pembuatan objek selesai, cukup memanggil fungsi *Parse* untuk melakukan penerjemahan. Setelah fungsi *Parse* dipanggil maka fungsi *Dump* dapat dijalankan untuk mendapatkan variabel dan nilai dari pesan Syslog tersebut secara iterasi. Kode sumber untuk melakukan penerjemahan tertera pada kode programB.7. Karena tidak dapat di prediksi apakah perangkat yang mengirimkan pesan Syslog menggunakan RFC5424 atau RFC3164, maka dilakukan 2 kali penerjemahan. Penerjemahan pertama dilakukan menggunakan RFC3164 karena perangkat lama masih banyak yang menggunakan RFC3164. Kemudian jika terjadi kegagalan, pesan akan diterjemahkan menggunakan RFC5424.

4.3.2 Menjalankan Syslog parser sebagai layanan menggunakan protokol UDP dan TCP

Perangkat jaringan seperti *router* dan *access point* menggunakan protokol UDP sebagai media pengiriman pesan Syslognya. Protokol UDP sangat cepat dan tidak membebani pengirim, walaupun data yang dikirimkan tidak dapat dipastikan sampai ke tujuan. Oleh karena itu perangkat-perangkat jaringan menggunakan protokol UDP sebagai media pengiriman, karena tidak terlalu membebani perangkat. Langkah-langkah pembuatan layanan yang berjalan menggunakan protokol UDP adalah sebagai berikut:

- Membuat variabel konstan untuk keperluan pembuatan layanan dan koneksi ke Middleware. Pembuatan variabel sesuai dengan kode sumber B.8
- Setelah variabel konstan telah dibuat. Langkah selanjutnya adalah membuka koneksi UDP pada server. Untuk membuka koneksi UDP menggunakan kode program B.9 Untuk menerima data melalui protokol UDP menggunakan kode program B.10. Untuk melakukan ujicoba pada layanan, dapat menggunakan perintah `nc -u 127.0.0.1 9000` kemudian mengetikkan pesan yang akan dikirimkan pada layanan.

Perangkat komputer dan server menggunakan protokol TCP. Sehingga perlu dibangun layanan serupa yang berjalan pada protokol TCP.Langkah-langkah pembuatan layanan pada protokol TCP hampir sama dengan layanan pada protokol UDP. Perbedaanya terdapat pada variabel konstan dan cara membuka koneksi. Variabel konstan yang digunakan pada protokol TCP terdapat pada kode program B.12. Untuk membuka koneksi TCP terdapat pada kode program B.11. Untuk menerima data melalui protokol TCP menggunakan kode program B.13

4.3.3 Mengirimkan pesan Syslog yang telah diterjemahkan ke Middleware

Setelah mendapatkan pesan dari kode sumber B.13 atau B.10 dan diterjemahkan oleh kode sumber B.7, data akan dikirimkan ke Middleware menggunakan *HTTP REST API* melalui rute `/parser/:token/:db`. Data yang akan dikirimkan harus dalam format *JSON*, sehingga dibutuhkan objek struktur data pada kode sumber B.6. Penggunaan objek struktur data terdapat pada kode program B.14. Setelah menjalankan kode B.14 maka variabel `data` akan berisi JSON dari variabel-variabel Syslog yang telah diterjemahkan. Untuk mengirim JSON ke Middleware menggunakan kode program B.15

4.4 Implementasi Middleware

Middleware dibangun sebagai perantara antara parser,Dasbor dan klaster penyimpanan data. Middleware akan dipasang pada komputer dengan alamat IP `10.151.36.99`. Middleware dibangun menggunakan Node JS dan kerangka kerja Expressjs. Untuk menjalankan Node JS, terlebih dahulu paket Node JS harus terpasang pada komputer server.

4.4.1 Skema Basis Data Autentikasi Menggunakan Mysql

Untuk melakukan proses autentikasi, data pengguna yang valid harus tersimpan dalam media penyimpanan. Media penyimpanan yang paling tepat adalah basis data relasional salah satunya yaitu Mysql. Mysql server yang digunakan adalah versi 5.5. Mysql server memiliki definisi tabel `auth`, `column_mapping` dan `cassandra_type`. Tabel `auth` digunakan untuk proses autentikasi. Proses autentikasi menggunakan 2 faktor autentikasi, yaitu nama basis data dan token. Berikut definisi tabel `auth` pada tabel 4.1

Tabel 4.1: Tabel auth

No	Kolom	Tipe	Keterangan
1	id	int	Sebagai primary key pada tabel, nilai awal adalah `AUTO_INCREMENT`.
2	token	varchar(255)	token untuk autentikasi hasil hash dari kolom maintainer, db dan created_at/updated_at.
3	maintainer	varchar(255)	nama pengembang yang bertanggung jawab.
4	db	varchar(255)	nama penyimpanan data
5	created_at	datetime	waktu pembuatan penyimpanan data.
7	updated_at	datetime	waktu penyimpanan data di perbarui.
6	deleted_at	datetime	waktu penyimpanan data dihapus, nilai awal adalah `null`.

Tabel `column_mapping` digunakan untuk memetakan nama penyimpanan data pada Mysql ke penyimpanan data pada klaster Cassandra. Definisi tabel column_mapping seperti pada tabel 4.2

Tabel 4.2: Tabel column_mapping

No	Kolom	Tipe	Keterangan
1	id	int	Sebagai primary key pada tabel, nilai awal adalah `AUTO_INCREMENT`.
2	auth_id	int	sebagai *foreign key* dari tabel `auth`.

Tabel 4.2: Tabel column_mapping

No	Kolom	Tipe	Keterangan
3	column_name	varchar(255)	nama kolom sesuai dengan penyimpanan data pada klaster Cassandra.
4	column_type	int	sebagai *foreign key* dari tabel `cassandra_type`
5	created_at	datetime	waktu pembuatan data.
7	updated_at	datetime	waktu data di perbarui.
6	deleted_at	datetime	waktu data dihapus, nilai awal adalah `null`.

4.4.2 Implementasi pengendali parser

Pengendali parser digunakan sebagai perantara antara klaster penyimpanan data dengan Syslog parser. Autentikasi menggunakan 2 faktor, yaitu nama penyimpanan data dan token. Pengendali parser ditulis ke dalam berkas pengendali bernama `parser.js` pada direktori `controller`. Langkah-langkah penerimaan data hingga data disimpan sebagai berikut :

- Melakukan koneksi ke basis data autentikasi untuk mengetahui apakah pengguna valid atau tidak. Koneksi ke basis data Mysql menggunakan pustaka `knex` dan `bookshelf`. Dengan kerangka kerja Express-MVC hanya perlu mengubah koneksi pada berkas `config/database/driver/mysql/mysqlORMdriver.js`. Ubah `host,user,password dan database` sesuai dengan kode sumber B.16
- Autentikasi dilakukan menggunakan model `Auth` yang telah didefinisikan sebelumnya. Model Auth menggunakan 2 faktor autentikasi nama basis data dan token yang didapatkan dari Syslog parser. Proses autentikasi sesuai dengan kode program B.17. Jika autentikasi telah berhasil

maka kode selanjutnya dapat dijalankan di dalam cakupan logika `if(data)` pada kode program B.17

- Untuk berinteraksi dengan klaster Cassandra dimulai dengan membuka koneksi terlebih dahulu. Koneksi dapat dibuka melalui salah satu dari 3 server Cassandra dalam satu klaster, dengan menggunakan pustaka `cassandra-driver` untuk Node JS. Keuntungan menggunakan pustaka `cassandra-driver` adalah semua server akan dihubungi secara otomatis berdasarkan respon tercepat. Jika salah satu server mati maka driver secara otomatis akan membuka koneksi dengan server yang lain. Koneksi dibuka dengan menggunakan kode sumber B.4. Objek `client` pada kode sumber B.4 dapat digunakan untuk menjalankan perintah-perintah CQL(Cassandra Query Language) berulang kali.
- Rute-rute *HTTP REST API* yang diatur oleh pengendali parser ditunjukkan pada tabel 4.3

Tabel 4.3: Rute REST API pada pengendali parser

No	Rute	Metode	Aksi
1	/parser/:token/:db	GET	Mendapatkan data dari penyimpanan data sesuai dengan token dan nama penyimpanan data
2	/parser/:token/:db	POST	Memasukkan data kedalam penyimpanan data sesuai dengan token dan nama penyimpanan data

4.4.2.1 Implementasi Metode Pengambilan Data

Pengambilan data dilakukan melalui rute `/parser/:token/:db` dengan metode GET. Data akan diambil melalui pengendali parser. Pengendali parser akan membuka koneksi ke klaster Cassandra untuk mengambil data sesuai nama `db`. Pengambilan data hanya akan dilakukan pada permintaan yang telah terautentikasi. Proses pengambilan data sesuai dengan *pseudocode* 4.2

```
1  Terima permintaan pengambilan data
2  Proses autentikasi
3  IF autentikasi berhasil
4      while data belum terambil semua
5          data <- ambil data dari cassandra
6          IF data = 5000
7              Kirim data ke klien
8              data <- 0
9
```

Gambar 4.2: Pseudocode proses pengambilan data

Langkah-langkah proses pengambilan data sebagai berikut:

- Pengambilan data di inisialisasi oleh klien melalui permintaan ke rute `/parser/:token/:db` menggunakan metode GET.
- Pengendali parser akan melakukan autentikasi menggunakan variabel `token` dan `db`. Autentikasi menggunakan basis data Mysql. Proses autentikasi tertera pada kode sumber B.17
- Ketika autentikasi berhasil, pengendali parser akan mengambil data dari Cassandra. Sedangkan ketika autentikasi gagal, pengendali parser akan mengembalikan pesan kegagalan kepada klien
- Data diambil secara bertahap, dan ditampung pada variabel `data`. Ketika jumlah variabel `data` telah mencapai 5000, maka variabel `data` akan dikirimkan ke klien menggunakan Socket.IO. Setelah data terkirim, variabel

`data` akan di kosongkan. Pengosongan variabel `data` bertujuan untuk mengurangi beban pada middleware. Batasan 5000 ditentukan karena 5000 data adalah batas maksimal Cassandra dalam melakukan sekali pengambilan data. Jika jumlah data tidak mencapai 5000 sedangkan semua data telah terambil. Maka data akan langsung dikirimkan ke klien.

- Proses pengiriman akan diulangi hingga seluruh data berhasil terambil.

4.4.2.2 Implementasi Metode Penulisan Data

Pengambilan data dilakukan melalui rute `/parser/:token/:db` dengan metode POST. Data yang dikirimkan sudah dalam bentuk JSON dan akan di proses oleh pengendali parser. Proses penulisan data sesuai dengan *pseudocode* 4.3

```
1  data <- syslog dari komputer dan perangkat jaringan
2  data <- data yang telah diterjemahkan ke dalam bentuk json
3  Kirim data ke middleware
4  Proses autentikasi
5  IF autentikasi berhasil
6      data -> disimpan ke dalam Cassandra
7      IF penyimpanan gagal
8          Kembalikan pesan kegagalan
```

Gambar 4.3: Pseudocode proses penulisan data

Langkah-langkah proses penulisan data sebagai berikut:

- Pesan Syslog dari perangkat komputer dan jaringan akan di terjemahkan oleh Syslog parser sesuai kode sumber 4.1. Setelah data berhasil di format dalam bentuk JSON, data akan dikirim ke middleware melalui rute `/parser/:token/:db` menggunakan metode POST.
- Pengendali parser akan melakukan autentikasi menggunakan variabel `token` dan `db`. Autentikasi

menggunakan basis data Mysql. Proses autentikasi tertera pada kode sumber B.17

- Ketika autentikasi berhasil, pengendali parser akan menulis data ke Cassandra. Sedangkan ketika autentikasi gagal, pengendali parser akan mengembalikan pesan kegagalan kepada klien.
- Setelah data berhasil ditulis, pengendali parser akan mengembalikan pesan berhasil. Sedangkan ketika terjadi kegagalan, pengendali parser akan mengembalikan pesan kegagalan.

4.4.3 Implementasi pengendali storage

Pengembangan sistem lebih lanjut menuntut sistem dapat mengakomodasi jenis-jenis *log* lainnya. Penyimpanan data baru dibuat untuk proses pengembangan pencatatan *log* dengan protokol lain. Pengendali *storage* digunakan untuk mengakomodasi pembuatan penyimpanan data baru. Pengendali *storage* memiliki rute-rute *HTTP REST API* yang ditunjukkan oleh tabel 4.4

Tabel 4.4: Rute REST API pada pengendali storage

No	Rute	Metode	Aksi
1	/storage/	POST	Membuat penyimpanan data baru.
2	/storage/:id	PUT	Memperbarui penyimpanan data dengan id tertentu.
3	/storage/	GET	Mendapatkan daftar penyimpanan data

Tabel 4.4: Rute REST API pada pengendali storage

No	Rute	Metode	Aksi
4	/storage/:id	GET	Mendapatkan keterangan kolom dan tipe penyimpanan data dengan id tertentu.
5	/storage/	GET	Mendapatkan daftar penyimpanan data

4.4.4 Implementasi pengendali function

Pengolahan data Syslog dapat menggunakan fungsi baru yang didefinisikan oleh pengguna. Fungsi yang di dukung hanya fungsi dalam bahasa javascripts. Pengendali *function* mengakomodasi pembuatan *Query Profile* dan kode yang akan dijalankan dalam lingkungan yang terisolasi. *Query Profile* adalah data yang akan diambil untuk diolah.*query profile* terdiri dari beberapa penyimpanan data *log* yang sama atupun berbeda dengan waktu tertentu relatif dari waktu saat ini. Sedangkan kode yang akan dieksekusi adalah kode dalam bahasa javacripts . Pengendali *function* memiliki rute-rute *HTTP REST API* yang ditunjukkan oleh tabel 4.5.

Tabel 4.5: Rute REST API pada pengendali function

No	Rute	Metode	Aksi
1	/function/plugin	GET	Mendapatkan daftar kode sumber yang telah dibuat.
2	/function/plugin/:name	GET	Mendapatkan isi dari kode sumber sesuai dengan nama kode sumber.

Tabel 4.5: Rute REST API pada pengendali function

No	Rute	Metode	Aksi
3	/function/query	GET	Mendapatkan daftar *Query Profile*.
4	/function/query/:name	GET	Mendapatkan keterangan *Query Profile* dengan nama tertentu.
5	/function/	PUT	Menyimpan *Query Profile* dan kode sumber sekaligus menjalankannya di dalam lingkungan yang terisolasi

4.5 Implementasi Penyimpanan Data

Penyimpanan data dibangun menggunakan basis data Cassandra versi 3.0 yang di *deploy* pada mesin virtual menggunakan kontainer Docker. Kontainer Docker yang digunakan adalah Docker versi 1.11.1. Klaster penyimpanan data yang akan dibangun terdiri dari 3 buah *node* menggunakan server virtual Proxmox. Server yang dibangun memiliki alamat IP masing-masing `10.151.36.96`,`10.151.36.97`,`10.151.36.98`. Di setiap server akan terpasang Cassandra yang saling mendistribusikan data dalam cluster tersebut.

4.5.1 Menjalankan Klaster Cassandra pada mesin Docker

Cassandra akan di pasang pada 3 buah server dengan alamat IP masing-masing adalah `10.151.36.96`,`10.151.36.97` dan `10.151.36.98`. Untuk memulai pemasangan Cassandra,

terlebih dahulu lakukan proses instalasi Docker seperti pada A.3. Setelah semua server terpasang mesin Docker , konfigurasi pada masing-masing server sebagai berikut :

- Pada server pertama dengan alamat IP `10.151.36.96` jalankan kontainer dengan perintah berikut `docker run --name cassa1 -v /home/ubuntu/cassa_dir/:/var/lib/cassandra -d -e CASSANDRA_BROADCAST_ADDRESS=10.151.36.96 -p 7000:7000 -p 9042:9042 cassandra:latest`, perintah tersebut akan menjalankan kontainer dengan nama cassa1 dan *images* Cassandra versi terbaru. Perintah `-v` menandakan direktori `/var/lib/cassandra` pada kontainer akan disimpan secara permanen pada penyimpanan server pertama yang terletak pada `/home/ubuntu/cassa_dir`, karena ketika tidak menggunakan perintah `-v` semua perubahann yang terjadi pada kontainer akan hilang ketika kontainer di hentikan. Perintah `-e CASSANDRA_BROADCAST_ADRESS=10.151.36.96` digunakan untuk membuka alamat *broadcast* pada IP `10.151.36.96` dan *port* 7000, alamat *broadcast* ini digunakan untuk melakukan distribusi data pada klaster cassandra. Perintah -p digunakan untuk membuka *port* virtual dalam kontainer agar dapat diakses dari luar dengan cara memetakan *port* dalam kontainer ke *port* pada *host* server.
- Pada server kedua jalankan kontainer Cassandra dengan mengarahkan *seeds* menuju server pertama. Jalankan perintah berikut `docker run --name cassa2 -v /home/ubuntu/cassa_dir/:/var/lib/cassandra -d -e CASSANDRA_BROADCAST_ADDRESS=10.151.36.97 -p`

`7000:7000 -p 9042:9042 -e CASSANDRA_SEEDS=10.151.36.96 cassandra:latest` , perintah `CASSANDRA_SEEDS=10.151.36.96` bertujuan untuk mengarahkan *seeds* ke server pertama. *Seeds* dibutuhkan oleh cassandra ketika pertama berjalan untuk memberitahu Cassandra agar tergabung kedalam klaster yang sama.

- Pada server ketiga jalankan perintah yang sama dengan server kedua yaitu sebagai berikut `docker run --name cassa3 -v /home/ubuntu/cassa_dir/:/var/lib/cassandra -d -e CASSANDRA_BROADCAST_ADDRESS=10.151.36.98 -p 7000:7000 -p 9042:9042 -e CASSANDRA_SEEDS=10.151.36.96 cassandra:latest`
- Untuk melihat apakah klaster Cassandra yang di bangun telah terkoneksi dengan baik perintah `docker exec -i -t cassa1 sh -c 'nodetool status'` dijalankan pada server pertama.

4.5.2 Skema penyimpanan data Syslog pada Cassandra

Penyimapan data Syslog menggunakan skema dengan tabel bernama logs. Deskripsi tabel logs seperti pada tabel 4.6.

Tabel 4.6: Tabel logs pada skema syslog

No	Kolom	Tipe	Keterangan
1	id	uuid	Sebagai primary key pada tabel, nilainya berupa nilai unik yang dibangkitkan ketika akan melakukan penambahan data.

Tabel 4.6: Tabel logs pada skema syslog

No	Kolom	Tipe	Keterangan
2	date	timestamp	Menyimpan waktu ketika data disimpan.
3	content	text	Laporan dari pesan syslog.
4	facility	int	Tipe program yang melakukan pelaporan Syslog.
5	hostname	text	Nama komputer yang mengirikan pesan.
7	ip	text	Alamat IP dari komputer yang mengirimkan pesan.
6	priority	int	Kode pada saat pesan Syslog dikirim dalam format rfc3164 atau rfc5424. Priority terdiri dari *severity* dan facility yang telah di sandi.
7	*severity*	int	Tingkat kepentingan dari pesan yang dikirim.
8	tag	text	Nama dari program yang melakukan pelaporan Syslog.
9	timestamp	timestamp	Waktu pada komputer pengirim ketika mencatat Syslog.

4.6 Implementasi Dasbor

Dasbor adalah aplikasi yang di gunakan sistem administrator untuk mengelola sistem. Dasbor diimplementasikan menggunakan Node JS dan kerangka kerja express js yang sama

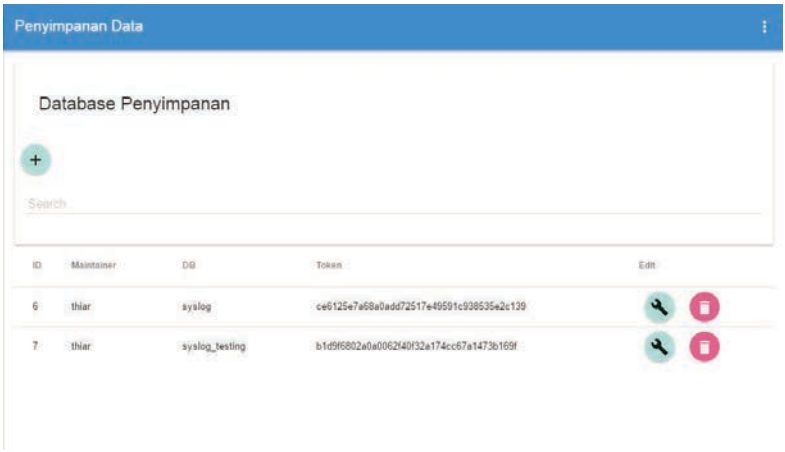
dengan Middleware, untuk proses pemasangan dapat mengacu pada lampiran A.1.1. Dasbor memiliki menu-menu sebagai berikut:

- Kelola Penyimpanan
- Kelola Definisi Penyimpanan
- Lihat Log Data
- Grafik Kejadian Syslog
- Fungsi

Masing-masing menu selanjutnya akan dijelaskan secara rinci.

4.6.1 Kelola Penyimpanan

Kelola penyimpanan adalah menu yang digunakan untuk mengelola penyimpanan data. Fitur yang didukung antara lain adalah membuat penyimpanan data baru dan memperbarui penyimpanan data yang telah ada sebelumnya. Antar muka Kelola penyimpanan ditunjukkan pada gambar 4.4.



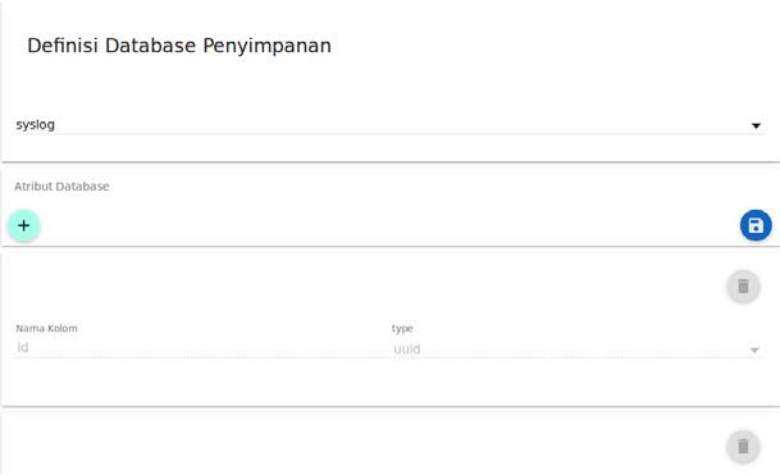
Gambar 4.4: Antar muka menu kelola penyimpanan

DB adalah nama penyimpanan data dan Token adalah

kumpulan karakter unik yang digunakan untuk proses autentikasi. Token digunakan sebagai faktor autentikasi bersama dengan nama penyimpanan data. Seluruh sumber daya pada middleware hanya bisa diakses jika menyertakan nama penyimpanan data dan token yang valid.

4.6.2 Kelola Definisi Penyimpanan

Kelola definisi penyimpanan adalah menu yang dapat di akses setelah membuat penyimpanan data di menu kelola penyimpanan. Menu ini bertujuan untuk mendefinisikan kolom yang diperlukan. Kolom-kolom yang didefinisikan harus sama dengan variabel-variabel yang dikirim oleh parser. Kolom awal yang harus ada adalah `id` dan `date`. Kedua `id` digunakan sebagai kolom pembeda antar baris sedangkan kolom `date` digunakan sebagai pencarian berdasarkan waktu. Antar muka kelola definisi penyimpanan ditunjukkan pada gambar 4.5.



Gambar 4.5: Antar muka menu kelola definisi penyimpanan

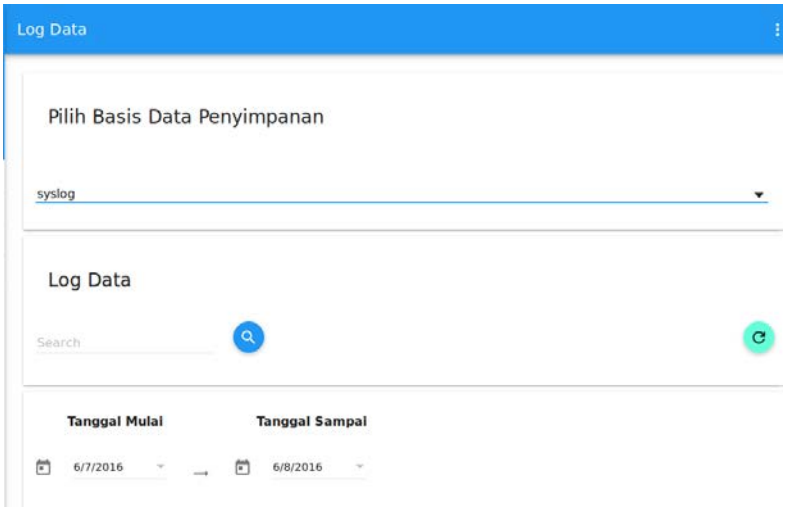
Untuk menambahkan kolom baru tekan tombol '+'. Kemudian masukkan nama kolom dan tipe kolom seperti pada gambar 4.6. Tipe kolom adalah jenis tipe data yang didukung oleh Cassandra. Jenis-jenis yang didukung antara lain :

- uuid
  Biasanya uuid digunakan sebagai primary key pada Cassandra.
- timestamp
  Biasanya timestamp digunakan pada penyimpanan data temporal. Penyimpanan data temporal membutuhkan waktu untuk setiap data yang disimpan.
- text
  Text adalah tipe data untuk menyimpan kalimat/*string*
- int
  Int adalah tipe data untuk menyimpan bilangan bulat

Gambar 4.6: Menambah kolom baru pada menu kelola definisi penyimpanan

4.6.3 Lihat Log Data

Hasil *log* yang telah tersimpan dapat diakses melalui menu lihat *log* data. Menu lihat *log* data menampilkan *log* berdasarkan penyimpanan data yang dipilih. Penyimpanan data yang dapat dipilh adalah penyimpanan data yang telah dibuat sebelumnya menggunakan menu kelola penyimpanan. Antar muka menu lihat *log* data ditunjukkan pada gambar 4.7.



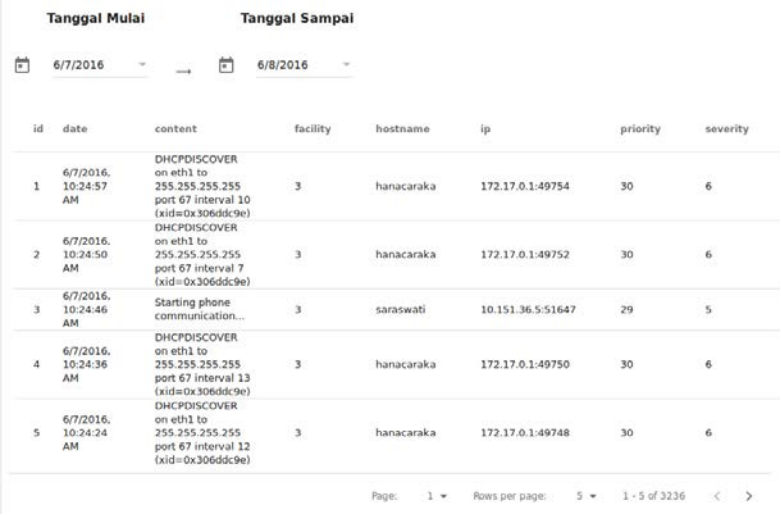
Gambar 4.7: Antar muka menu lihat *log* data

Menu lihat *log* data memiliki fitur seleksi antara lain:

- Seleksi berdasarkan tanggal
  Fitur ini digunakan untuk mendapatkan *log* dari tanggal tertentu. Setelah memilih penyimpanan data, tanggal mulai dan tanggal sampai harus di tentukan. Nilai awal dari tanggal awal adalah waktu kemarin sedangkan tanggal sampai memiliki nilai awal waktu hari ini. Tanggal sampai adalah tanggal saat ini pada pukul 23.59.
- Seleksi berdasarkan kata kunci
  Seleksi akan dilakukan menggunakan kata kunci yang tertulis di kolom *search*. Jika kata kunci ditemukan pada kolom di sebuah baris, maka baris tersebut akan dimasukkan ke dalam susunan. Proses pencarian akan berulang hingga baris terakhir. Setelah proses pencarian berakhir, susunan yang menyimpan hasil seleksi akan ditampilkan.

Hasil *log* akan diurutkan berdasarkan waktu penyimpanan secara

menurun. Setelah memilih fitur seleksi, hasil *log* data akan muncul seperti pada gambar 4.8.



Gambar 4.8: Hasil *log* ditampilkan dalam tabel

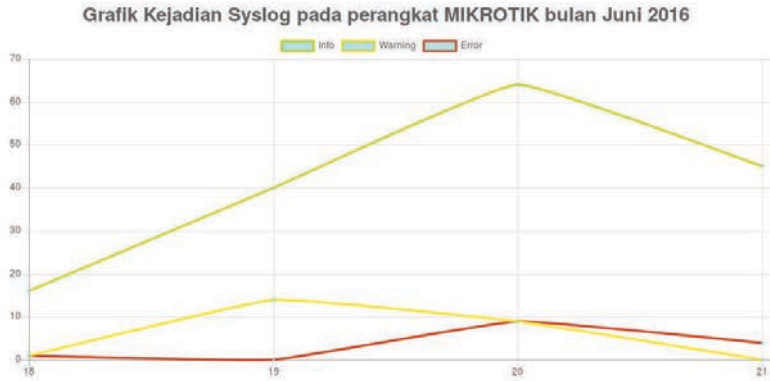
4.6.4 Grafik Kejadian Syslog

Proses pengolahan data membutuhkan penyajian data dalam bentuk grafik. Menu Grafik Kejadian Syslog akan menyajikan data kejadian Syslog tiap perangkat kedalam bentuk grafik. Grafik yang disajikan berupa data jumlah kejadian Syslog yang terjadi pada kurun waktu satu bulan. Grafik mengkategorikan seri data berdasarkan *severity* Syslog tiap perangkat. Daftar *severity* yang dikelompokkan antara lain:

- 1 Alert
- 2 Critical
- 3 Error
- 4 Warning
- 5 Notice

- 6 Informational

Antar muka menu Grafik Kejadian Syslog tertera pada gambar 4.9.

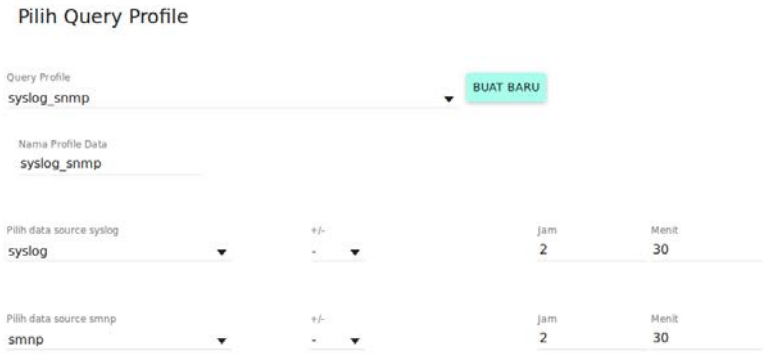


Gambar 4.9: Grafik Kejadian Syslog

4.6.5 Fungsi

Pemrosesan data *log* terkadang membutuhkan fungsi khusus yang dibuat oleh pengembang. Menu fungsi memiliki fitur menjalankan fungsi baru untuk pengolahan data log. Fungsi yang dapat dijalankan adalah fungsi yang dibuat dalam bahasa javascript. Fungsi hanya dapat mengolah data *log* yang diambil sebelumnya sesuai definisi Query Profile. Query Profile adalah definisi sumber data *log* yang akan digunakan dalam fungsi. Query Profile bisa terdiri dari beberapa penyimpanan data yang sama atau berbeda. Query Profile menyediakan data *log* berdasarkan waktu relatif dari waktu saat ini. Query Profile akan diterjemahkan menjadi CQL(Cassandra Query Language) dan dijalankan untuk mendapatkan data log. Query Profile akan disimpan di server dalam bentuk berkas dengan format JSON, sehingga memungkinkan untuk memperbarui atau menggunakan

kembali Query Profile yang telah dibuat sebelumnya. Antar muka pembuatan Query Profile ditunjukkan pada gambar 4.10.



Gambar 4.10: Contoh Query Profile *log* Syslog dan Snmp

Setelah menentukan Query Profile, kode bisa dibuat dan dijalankan. Antar muka pembuatan kode ditunjukkan dengan gambar 4.11. Query Profile dan Kode yang telah dibuat akan dikirim ke Middleware untuk dievaluasi. Pengiriman Query Profile dan Kode menggunakan rute `/function/` dengan metode `PUT`. Kode yang dijalankan tidak dapat mengakses sumber daya lain selain data *log* dari Query Profile. Pembatasan hak akses dilakukan menggunakan metode *sandbox*. Metode *sandbox* menjalankan kode program yang tidak terpercaya dalam lingkungan yang terisolasi dari program utama atau sistem operasi. Pembatasan hak akses ini bertujuan untuk melindungi program utama dari kode-kode berbahaya. Kode-kode berbahaya dapat merusak sistem dengan berbagai cara, antara lain :

- Menambah kode berbahaya ke dalam program utama.
- Mengambil data dari server tempat program utama dijalankan.

- Menghapus data dari server tempat program utama dijalankan.
- Mengubah kode program yang valid.
- Mematikan server tempat program utama dijalankan.

Penerapan *sandbox* pada Node JS menggunakan pustaka `vm`. Pustaka `vm` akan menjalankan kode dalam konteks baru. Walaupun kode dijalankan dalam konteks baru, tetapi kode masih berjalan dalam program yang sama sehingga kode program yang dapat berjalan adalah kode program yang ditulis dalam bahasa javascript. Menjalankan kode dalam program yang sama dapat mempercepat proses evaluasi.

Gambar 4.11: Pembuatan kode

BAB 5

PENGUJIAN DAN EVALUASI

5.1 Lingkungan Uji Coba

Lingkungan pengujian menggunakan komponen-komponen yang terdiri dari : satu server untuk Syslog parser dan Middleware, 3 server virtual Cassandra, 6 komputer penguji performa dan 12 Syslog Agen. Semua server virtual menggunakan virtualisasi proxmox. Pengujian dilakukan di Laboratorium Arsitektur dan Jaringan Komputer Jurusan Teknik Informatika ITS.

Spesifikasi untuk setiap komponen yang digunakan adalah sebagai berikut:

- Syslog Parser dan Middleware:
  - Perangkat Keras:
    * Processor Intel(R) Core(TM) i3 CPU 550 @ 3.20GHz
    * RAM 3692 MB
    * Ethernet interface Express Gigabit Ethernet Controller speed=100Mbit/s
  - Perangkat Lunak:
    * Sistem operasi Ubuntu 14.04.2 LTS
    * Docker 1.10.2
    * Golang 1.5.4
    * Node JS 5.9.1
    * OpenSSH
- Penyeimbang beban :
  - Processor Intel(R) Core(TM)2 Duo CPU E7200 @ 2.53GHz
  - RAM 1986 MB
  - Ethernet interface Express Gigabit Ethernet Controller speed=100Mbit/s
- Perangkat Lunak:
  - Sistem operasi Ubuntu 14.04 LTS

  - Docker 1.10.2
- Syslog Parser dan Middleware sebagai pekerja:
  - Pekerja 1:
    * Perangkat Keras:
      · Processor Intel(R) Core(TM)2 Duo CPU E7200 @ 2.53GHz
      · RAM 1986 MB
      · Ethernet interface Express Gigabit Ethernet Controller speed=100Mbit/s
    * Perangkat Lunak:
      · Sistem operasi Ubuntu 14.04 LTS
      · Docker 1.10.2
  - Pekerja 2:
    * Perangkat Keras:
      · Processor Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz
      · RAM 1986 MB
      · Ethernet interface Express Gigabit Ethernet Controller speed=100Mbit/s
    * Perangkat Lunak:
      · Sistem operasi Ubuntu 14.04 LTS
      · Docker 1.10.2
  - Pekerja 3:
    * Perangkat Keras:
      · Processor Intel(R) Core(TM)2 Duo CPU E7200 @ 2.53GHz
      · RAM 981 MB
      · Ethernet interface Express Gigabit Ethernet Controller speed=100Mbit/s
    * Perangkat Lunak:
      · Sistem operasi Ubuntu 14.04 LTS
      · Docker 1.10.2
- Komputer Host Proxmox

  - Perangkat Keras:
    * Processor 4 x Intel(R) Xeon(R) CPU E3-1220 v2 @ 3.10GHz
    * Ram 7510 MB
  - Perangkat Lunak:
    * Proxmox Virtual Environtment Version 4.1-15
- Komputer virtual proxmox 3 buah Cassandra
  - Perangkat Lunak:
    * Docker 1.10.2
    * Cassandra 3.5
    * RAM 1497 MB
- Komputer Penguji Fungsionalitas:
  - Perangkat Keras:
    * Processor Intel(R) Core(TM) i3-3240 @ 3.40GHz
    * RAM 4096 MB
  - Perangkat Lunak:
    * Sistem operasi windows 8.0
    * Peramban Mozilla Firefox 47.0
- Komputer Penguji Performa 6 buah:
  - 3 Buah Komputer
    * Perangkat Keras:
      · Processor Intel(R) Core(TM) i3-3240 @ 3.40GHz
      · RAM 4096 MB
    * Perangkat Lunak:
      · Sistem operasi ubuntu 14.04
      · sendip v2.5
  - 3 Buah Mini Komputer
    * Perangkat Keras:
      · Processor Intel(R) Celeron(R) CPU N2820 @ 2.13GHz
      · RAM 4096 MB
    * Perangkat Lunak:

        - Sistem operasi Linux Mint 17 Qiana
        - sendip v2.5
- Syslog Agen:
  - Komputer fisik 5 buah server praktikum jaringan komputer
    - Perangkat Lunak:
      - Sistem operasi Debian GNU/Linux 8.3 (jessie)
      - Rsyslogd versi 8.4.2
  - Komputer fisik 3 buah komputer workstation Laboratorium Arsitektur dan Jaringan Komputer
    - Perangkat Lunak:
      - Sistem operasi windows 8 dan windows 8.1
      - Datagram SyslogAgent versi 3.6
  - Komputer fisik 1 buah Active Directory server
    - Perangkat Lunak:
      - Sistem operasi windows server 2012
      - Datagram SyslogAgent versi 3.6
  - Komputer fisik 2 buah Server Riset AJK
    - Perangkat Lunak:
      - Sistem operasi Ubuntu 14.04.4 LTS
      - Rsyslogd versi 7.4.4
  - Miktrotik 1 buah routerboard RB951 Series
    - Perangkat Lunak:
      - RouterOS v6.15

Untuk akses ke masing-masing komponen, dibutuh pembagian alamat IP sesuai yaitu :

- Syslog Parser memiliki IP 10.151.38.181
- Komputer virtual proxmox 3 buah Cassandra
  - server pertama memiliki alamat IP 10.151.36.96
  - server kedua memiliki alamat IP 10.151.36.97
  - server ketiga memiliki alamat IP 10.151.36.98
- Midleware memiliki IP 10.151.38.181

- Komputer penguji fungsionalitas memiliki alamat IP 10.151.36.24
- Penyeimbang beban memiliki alamat IP 10.151.36.15

5.2 Skenario Uji Coba

Uji coba akan dilakukan untuk mengetahui keberhasilan sistem yang telah dibangun. Skenario pengujian dibedakan menjadi 2 bagian yaitu :

- **Uji Fungsionalitas**.
  Pengujian ini didasarkan pada fungsionalitas yang disajikan sistem. Uji coba yang akan dilakukan adalah uji menyimpan data dari Syslog agen dan uji fungsionalitas aplikasi dasbor. Uji coba dilakukan untuk mengetahui kemampuan sistem menangani pesan syslog dari berbagai perangkat yang beragam dan keberhasilan aplikasi dasbor menjalankan fungsi-fungsinya.
- **Uji Performa**.
  Pengujian ini untuk menguji ketahanan sistem terhadap sejumlah permintaan *log* yang masuk. Uji coba yang akan dilakukan adalah uji coba skalabilitas. Uji coba skalabilitas diperlukan untuk menunjukkan performa sistem semakin membaik seiring dilakukan perluasan pada sistem.

5.2.1 Skenario Uji Fungsionalitas

Uji fungsionalitas dibagi menjadi 2, yaitu uji menyimpan data dari Syslog Agen dan uji fungsionalitas menu aplikasi Dasbor.

5.2.1.1 Uji menyimpan data dari Syslog agen

Pengujian Syslog Agen dilakukan dengan mengarahkan layanan syslog pada setiap perangkat menuju ke Syslog parser. Perangkat harus terkoneksi dengan jaringan ITS dan telah

terpasang Syslog agen. Alamat IP tujuan adalah 10.151.38.181 dengan nomor port 9000. Penyimpanan data oleh Syslog agen dilakukan secara *realtime* tergantung dari masing-masing perangkat. Setiap kejadian yang terjadi pada tiap perangkat akan langsung dikirimkan ke sistem dan tercatat sesuai dengan waktu pengiriman. Pesan Syslog yang dikirimkan adalah pesan dengan *severity* sebagai berikut:

- 1 Alert
  Pemberitahuan atas masalah yang harus segera di tangani.
- 2 Critical
  Pemberitahuan ketika sistem berada dalam keadaan kritikal.
- 3 Error
  Pemberitahuan ketika terjadi *error*.
- 4 Warning
  Mengindikasikan *error* akan terjadi, seperti *filesystem* hanya tersisa 2 GB.
- 5 Notice
  Pemberitahuan terkait kejadian yang tidak terlalu penting.
- 6 Informational
  Informasi terkait apa yang terjadi pada sistem, seperti aplikasi berhasil di jalankan atau di matikan.

Daftar uji fungsionalitas menyimpan data dari Syslog Agen dijelaskan pada tabel 5.1

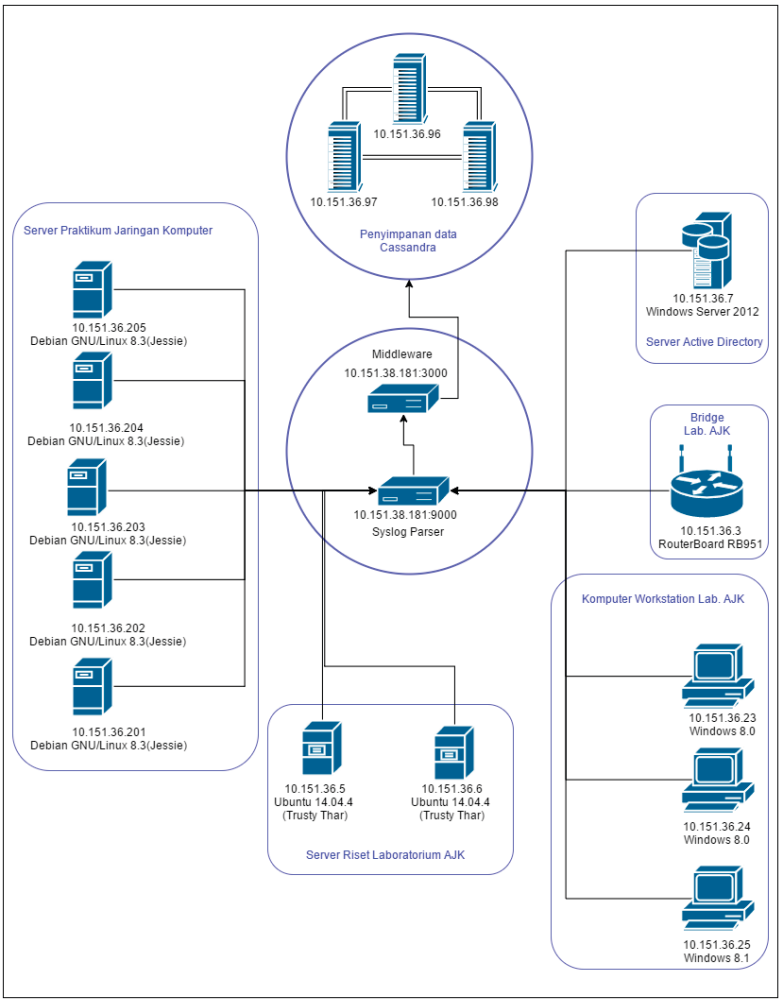
Tabel 5.1: Skenario Uji Coba Syslog Agen

No	Syslog Agen	Uji Coba	Hasil Harapan
1	Ubuntu 14.04	Mengirim pesan syslog yang telah diformat ke Syslog parser	Pesan Syslog berhasil diterjemahkan dan tersimpan pada Cassandra.

Tabel 5.1: Skenario uji Coba Syslog Agen

No	Syslog Agen	Uji Coba	Hasil Harapan
2	Debian 8	Mengirim pesan syslog yang telah diformat ke Syslog parser	Pesan Syslog berhasil diterjemahkan dan tersimpan pada Cassandra.
3	Windows server 2012	Mengirim pesan syslog yang telah diformat ke Syslog parser	Pesan Syslog berhasil diterjemahkan dan tersimpan pada Cassandra.
4	Windows 8	Mengirim pesan syslog yang telah diformat ke Syslog parser	Pesan Syslog berhasil diterjemahkan dan tersimpan pada Cassandra.
5	RouterOS v6.15	Mengirim pesan syslog yang telah diformat ke Syslog parser	Pesan Syslog berhasil diterjemahkan dan tersimpan pada Cassandra.

Arsitektur uji menyimpan data dari Syslog agen tertera pada gambar 5.1. Sistem akan di uji menggunakan perangkat-perangkat Syslog agen yang telah di definisikan pada Bab 5.1. Syslog agen yang digunakan tersebar pada subnet 10.151.38.0/24 dan 10.151.36.0/24. Perangkat-perangkat yang digunakan beragam untuk mendapatkan hasil yang lebih baik. Semakin beragam perangkat yang digunakan dapat menjadi tolok ukur kehandalan sistem dalam menangani pencatatan syslog dari berbagai perangkat. Alamat IP tiap perangkat dijelaskan pada gambar 5.1.



Gambar 5.1: Arsitektur Pengujian Syslog Agen

5.2.1.2 Uji Fungsionalitas Aplikasi Dasbor

Aplikasi Dasbor digunakan untuk mengelola data syslog. Aplikasi Dasbor memiliki 5 menu yaitu Kelola Penyimpanan, Kelola Definisi Penyimpanan, Lihat Log Data, Fungsi dan Grafik Kejadian Syslog. Rancangan pengujian dan hasil yang diharapkan ditunjukkan dengan tabel 5.2

Tabel 5.2: Skenario Uji Fungsionalitas Aplikasi Dasbor

No	Menu	Uji Coba	Hasil Harapan
1	Kelola Penyimpanan	Membuat penyimpanan data baru	penyimpanan data baru berhasil dibuat di cassandra dan mysql, token berhasil dibuat.
		Memperbarui penyimpanan data yang sudah ada	penyimpanan data pada cassandra dan mysql berhasil diperbarui, token berhasil diperbarui.
2	Kelola Definisi Penyimpanan	Menambahkan kolom pada penyimpanan data	Kolom pada cassandra dan mysql berhasil ditambahkan .
		Mengganti tipe data kolom pada penyimpanan data	Kolom pada pada cassandra dan mysql berhasil diperbarui.

Tabel 5.2: Skenario Uji Fungsionalitas Aplikasi Dasbor

No	Menu	Uji Coba	Hasil Harapan
		Menghapus kolom pada penyimpanan data	Kolom pada pada cassandra dan mysql berhasil dihapus.
3	Lihat Log Data	Melihat *log* sesuai penyimpanan data yang dipilih dan jarak waktu pencarian	data *log* akan ditampilkan sesuai penyimpanan data yang dipilih dan jarak waktu yang ditentukan minimal satu hari, Data *log* disajikan dalam bentuk tabel.
		Menampilkan data *log* yang mengandung kata tertentu	Tabel yang ditampilkan hanya data *log* yang mengandung kata tersebut
4	Fungsi	Membuat Query Profile dengan satu sumber data	Query Profile dapat disimpan dan dijalankan.

Tabel 5.2: Skenario Uji Fungsionalitas Aplikasi Dasbor

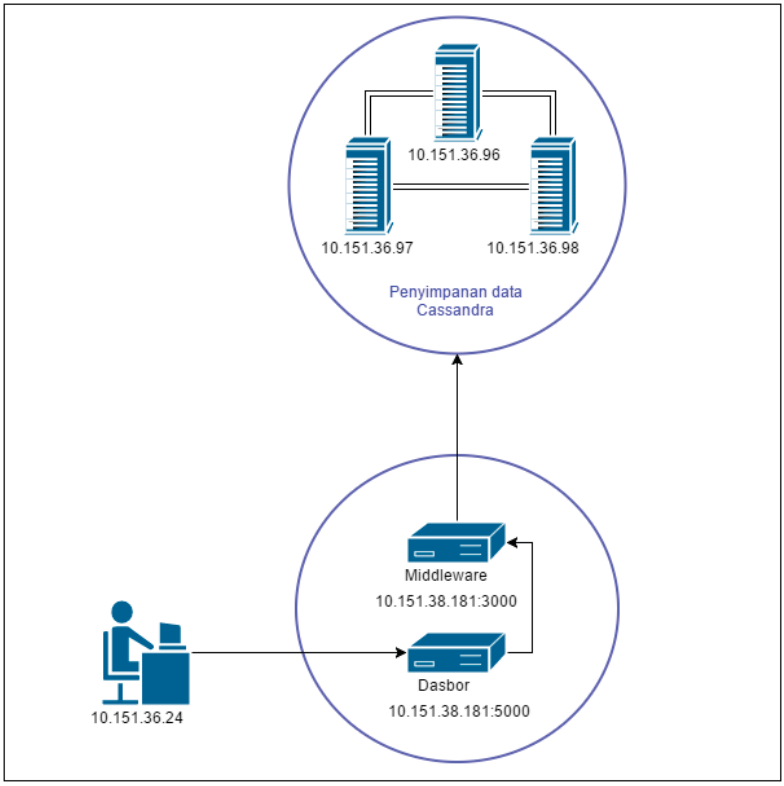
No	Menu	Uji Coba	Hasil Harapan
		Membuat kode sumber baru untuk keperluan pengolahan *log* data. Kode sumber yang digunakan B.19. Kode sumber B.19 bertujuan menampilkan *log* data berdasarkan severity dan hostname tertentu. Hostname yang digunakan sebagai contoh adalah `hanacaraka` sedangkan *severity* yang digunakan sebagai contoh adalah *severity* level 6.	kode dapat mengakses dan mengolah data yang telah diambil oleh Query Profile.

Tabel 5.2: Skenario Uji Fungsionalitas Aplikasi Dasbor

No	Menu	Uji Coba	Hasil Harapan
		Mencegah kode sumber berbahaya untuk dijalankan. Kode sumber yang digunakan B.20. Kode sumber B.20 akan melakukan serangan *file inclusion* terhadap sistem. *file inclusion* akan dilakukan dengan menambahkan berkas baru pada direktori sistem.	kode sumber berbahaya tidak dapat berjalan
5	Grafik Kejadian Syslog	Menampilkan jumlah kejadian Syslog ke dalam grafik.	Data kejadian Syslog dapat disajikan dalam bentuk grafik.
		Menampilkan data kejadian berdasarkan bulan dan tahun ke dalam grafik.	Grafik yang ditampilkan berdasarkan bulan dan tahun yang telah dipilih.
		Mengelompokkan data berdasarkan *severity* ke dalam grafik.	Grafik dapat mengelompokkan data berdasarkan *severity*.

Pengujian dilakukan menggunakan komputer penguji dengan

alamat IP 10.151.36.24 dan menggunakan peramban mozilla firefox. Arsitektur pengujian fungsionalitas aplikasi dasbor tertera pada gambar 5.2



Gambar 5.2: Arsitektur Pengujian Aplikasi Dasbor

5.2.2 Skenario Uji Performa dan Skalabilitas

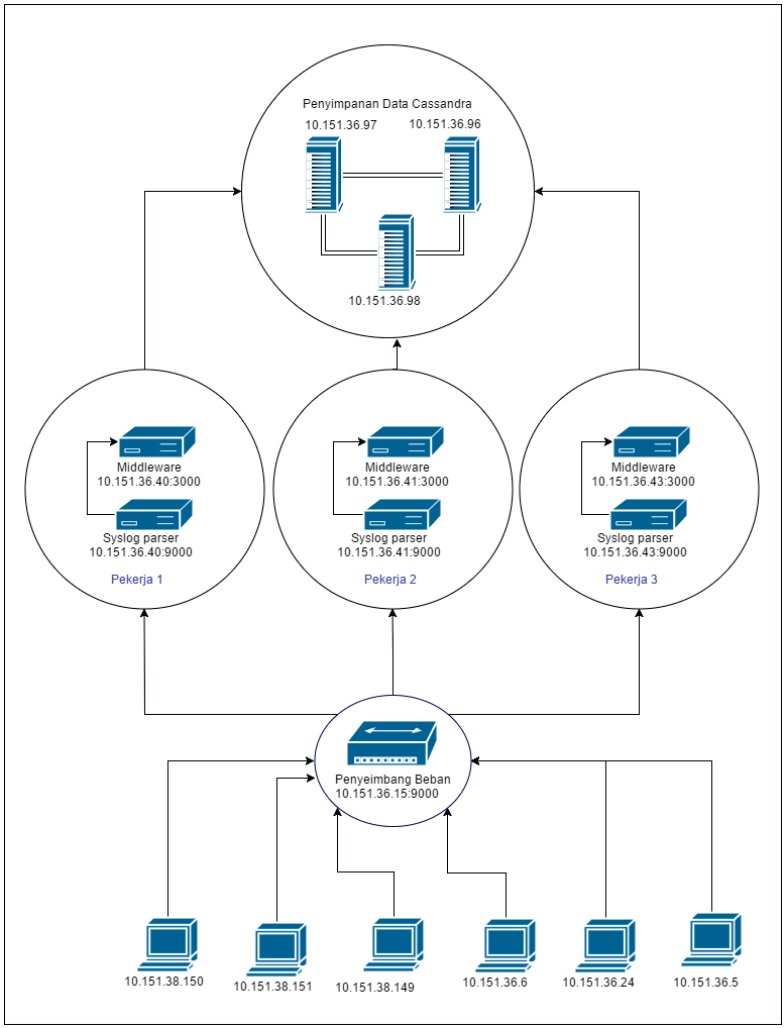
Semakin banyak syslog agen yang ditangangani, sistem harus dapat di skala untuk menerima permintaan penyimpanana yang lebih banyak. Uji coba performa di lakukan untuk menguji performa sistem yang di skala secara bertahap. Pengujian

dilakukan dengan metode *stress test* pada sistem. Metode *stress test* dilakukan untuk mengetahui batas akhir kemampuan sistem dalam menangani permintaan. Metode *stress test* dilakukan dengan cara memberikan permintaan dalam jumlah besar kepada sistem. Kemudian akan dilakukan evaluasi tentang ketahanan sistem dalam menerima permintaan yang banyak. Uji coba skalabilitas dilakukan dengan mengirimkan permintaan penyimpanan dalam protokol UDP menggunakan aplikasi `sendip`. Permintaan akan dikirimkan menggunakan 6 komputer. Permintaan yang dikirimkan tiap komputer penguji adalah sebanyak 25 permintaan pada masing-masing *thread*. Pengujian akan dilakukan secara bertahap menggunakan jumlah *thread* yang berbeda. *Thread* yang digunakan pada pengujian adalah 100,200,300,400 dan 500. Sistem akan di evaluasi secara betahap sejumlah komputer pekerja yang digunakan. Komputer pekerja yang digunakan pada uji coba skalabilitas sebanyak 1,2 dan 3 buah. Uji coba skalabilitas akan menguji performa sistem antara lain :

- Ketahanan sistem menangani permintaan dalam jumlah besar pada setiap ujicoba. Uji coba dilakukan untuk mengetahui apakah sistem tetap berjalan atau akan mengalami *crash*.
- Prosentase kehilangan data ketika menghadapi banyak permintaan secara bersamaan pada setiap ujicoba. Ujicoba dilakukan untuk mengetahui prosentase kehilangan data seiring bertambahnya permintaan yang datang ke sistem.
- Prosentase penggunaan CPU pada setiap ujicoba. Ujicoba dilakukan untuk mengetahui prosentase penggunaan CPU seiring bertambahnya permintaan yang datang ke sistem.

Uji coba dilakukan untuk membuktikan dengan bertambahnya jumlah pekerja akan mempengaruhi performa sistem. Perbandingan akan dilakukan pada uji coba dengan menggunakan 1 pekerja, 2 pekerja dan 3 pekerja. Kemudian

akan di evaluasi pengaruh jumlah pekerja terhadap performa sistem. Arsitektur pengujian tertera pada gambar 5.3



Gambar 5.3: Arsitektur Pengujian Performa dan Skalabilitas

5.3 Hasil Uji Coba dan Evaluasi

Berikut dijelaskan hasil uji coba dan evaluasi berdasarkan skenario yang sudah dijelaskan pada bab 5.2.

5.3.1 Uji Fungsionalitas

Berikut dijelaskan hasil pengujian fungsionalitas pada sistem yang sudah dibangun.

5.3.1.1 Menyimpan data dari Syslog Agen

Dilakukan pengujian untuk setiap agen. Setiap agen yang telah ditentukan pada tabel 5.1 akan mengirimkan pesan Syslog secara bersamaan sesuai keadaan yang terjadi pada masing-masing agen. Hasil pengujian seperti tertera pada tabel 5.7.

Tabel 5.3: Hasil Uji Coba Syslog Agen

No	Syslog Agen	Uji Coba	Hasil
1	Ubuntu 14.04	Mengirim pesan syslog yang telah diformat ke Syslog parser	OK.
2	Debian 8	Mengirim pesan syslog yang telah diformat ke Syslog parser	OK.
3	Windows server 2012	Mengirim pesan syslog yang telah diformat ke Syslog parser	OK.

Tabel 5.3: Hasil Uji Coba Syslog Agen

No	Syslog Agen	Uji Coba	Hasil
4	Windows 8	Mengirim pesan syslog yang telah diformat ke Syslog parser	OK.
5	RouterOS v6.15	Mengirim pesan syslog yang telah diformat ke Syslog parser	OK.

Sesuai dengan skenario ujicoba yang diberikan pada Tabel 5.1, hasil ujicoba menunjukkan semua agen berhasil ditangani.

5.3.1.2 Uji Fungsionalitas Aplikasi Dasbor

Sesuai dengan skenario pengujian yang akan dilakukan pada aplikasi dasbor. Pengujian dilakukan dengan menguji setiap menu pada aplikasi dasbor. Hasil uji coba tertera pada tabel 5.4.

Tabel 5.4: Hsil Uji Fungsionalitas Aplikasi Dasbor

No	Menu	Uji Coba	Hasil
1	Kelola Penyimpanan	Membuat penyimpanan data baru	Penyimpanan data dan token berhasil dibuat.
		Memperbarui penyimpanan data yang sudah ada	Penyimpanan data dan token berhasil diperbarui.
2	Kelola Definisi Penyimpanan	Menambahkan kolom pada penyimpanan data	Kolom berhasil ditambahkan.

Tabel 5.4: Hsil Uji Fungsionalitas Aplikasi Dasbor

No	Menu	Uji Coba	Hasil
		Mengganti tipe data kolom pada penyimpanan data	Kolom berhasil diperbarui.
		Menghapus kolom pada penyimpanan data	Kolom berhasil dihapus.
3	Lihat Log Data	Melihat *log* sesuai penyimpanan data yang dipilih dan jarak waktu pencarian	Data berhasil ditampilkan.
		Menampilkan data *log* yang mengandung kata tertentu	Data pada tabel berhasil di seleksi.
4	Fungsi	Membuat Query Profile dengan satu sumber data	Query Profile tersimpan dan dapat dijalankan.
		Membuat kode sumber baru untuk keperluan pengolahan *log* data. Kode sumber yang digunakan B.19	Kode berhasil dijalankan.
		Mencegah kode sumber berbahaya untuk dijalankan. Kode sumber yang digunakan B.20	Kode tidak berjalan.

Tabel 5.4: Hsil Uji Fungsionalitas Aplikasi Dasbor

No	Menu	Uji Coba	Hasil
5	Grafik Kejadian Syslog	Menampilkan jumlah kejadian Syslog ke dalam grafik.	Data kejadian Syslog berhasil disajikan dalam grafik.
		Menampilkan data kejadian berdasarkan bulan dan tahun ke dalam grafik.	Grafik berhasil ditampilkan berdasarkan bulan dan tahun yang telah dipilih.
		Mengelompokkan data berdasarkan *severity* ke dalam grafik.	Grafik dapat mengelompokkan data berdasarkan *severity*.

Semua menu pada aplikasi dasbor berjalan sebagaimana mestinya. Aplikasi dasbor tidak dapat diserang dengan serangan *file inclusion* karena kode sumber dijalankan dalam lingkungan yang terisolasi.

5.3.2 Uji Performa dan Skalabilitas

Seperti yang sudah dijelaskan pada bab 5.2 pengujian performa dan skalabilitas dilakukan menggunakan 6 komputer penguji. Pengujian dilakukan secara bertahap dengan menggunakan jumlah *thread* yang berbeda pada tiap komputer penguji. Pada masing-masing *thread* akan mengirimkan 25 permintaan. Pengujian dilakukan pada sistem dengan jumlah pekerja 1,2 dan 3. Evaluasi akan dilakukan terhadap prosentase jumlah data yang hilang dan penggunaan CPU pada tiap pengujian. Jumlah data yang dikirim adalah akumulasi data yang

dikirimkan tiap thread pada 6 komputer penguji. Jumlah data yang dikirimkan dapat di hitung dengan perhitungan `25 x jumlah thread x 6`. Prosentase jumlah data yang hilang di dapatkan dengan membandingkan data yang masuk dan data yang dikirimkan. Prosentase jumlah data yang hilang di dapatkan dengan perhitungan `data masuk / jumlah data dikirim x 100%`. Perbandingan akan dilakuan terhadap sistem dengan jumlah pekerja 1,2 dan 3. Poin perbandingan yang digunakan adalah prosentase jumlah data yang hilang dan penggunaan CPU pada tiap pengujian. Hasil pengujian sebagai berikut :

- Pengujian dengan 1 pekerja:

Tabel 5.5: Hasil Uji Coba menggunakan 1 pekerja

Jumlah Thread	Jumlah Data Dikirim	Jumlah Data Masuk	Prosentase Kehilangan Data	Penggunaan CPU
100	15000	12607	15,95%	87,5%
200	30000	15969	46,77%	94,5%
300	45000	8195	81,79%	96%
400	60000	6265	89,56%	100%
500	75000	9004	87,99%	100%

- Pengujian dengan 2 pekerja:

Tabel 5.6: Hasil Uji Coba menggunakan 2 pekerja

Jumlah Thread	Jumlah Data Dikirim	Jumlah Data Masuk	Prosentase Kehilangan Data	Penggunaan CPU
100	15000	14532	3,12%	87,5%
200	30000	23344	22,19%	92%
300	45000	14488	67,80%	95,5%

Tabel 5.6: Hasil Uji Coba menggunakan 2 pekerja

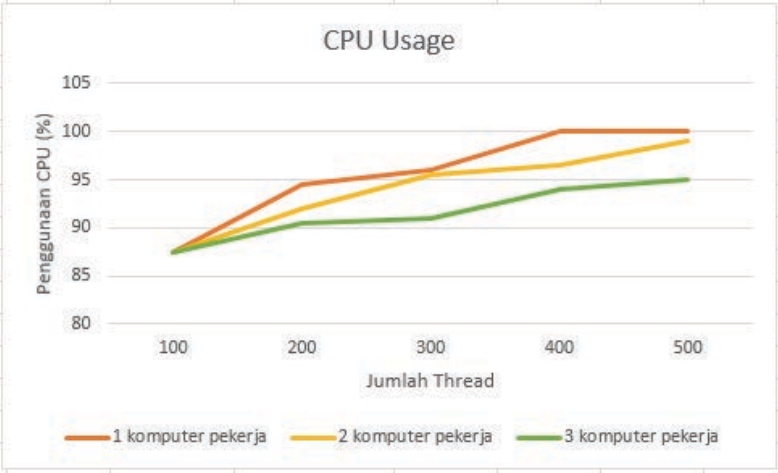
Jumlah Thread	Jumlah Data Dikirim	Jumlah Data Masuk	Prosentase Kehilangan Data	Penggunaan CPU
400	60000	14425	75,96%	96,5%
500	75000	25399	66,14%	99%

- Pengujian dengan 3 pekerja:

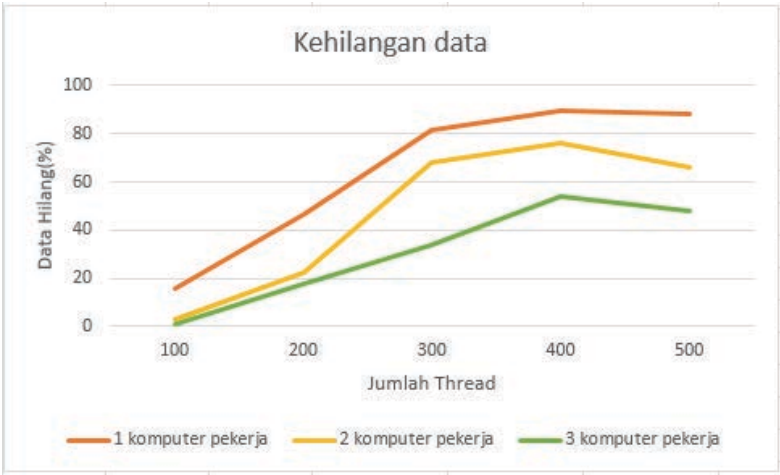
Tabel 5.7: Hasil Uji Coba menggunakan 3 pekerja

Jumlah Thread	Jumlah Data Dikirim	Jumlah Data Masuk	Prosentase Kehilangan Data	Penggunaan CPU
100	15000	14911	0,59%	87,5%
200	30000	24768	17,44%	90,5%
300	45000	29782	33,82%	91%
400	60000	27769	53,72%	94%
500	75000	38932	48,09%	95%

Dari hasil uji coba dapat dilihat, semakin banyak pekerja yang digunakan maka sistem dapat menangani permintaan dengan lebih baik. Penggunaan CPU berkurang seiring dengan betambahnya jumlah pekerja yang digunakan. Dari hasil percobaan terbukti dengan menambahkan jumlah komputer pekerja dapat mengurangi baik penggunaan CPU maupun kehilangan data. Hal ini terjadi karena beban di distribusikan secara merata kepada setiap komputer pekerja. Dengan menurunnya penggunaan CPU akan mengurangi prosentase kehilangan data. Semakin menurunnya data yang hilang, sistem akan menjadi lebih handal dalam menangani permintaan. Berikut grafik hasil uji coba performa dan skalabilitas.



Gambar 5.4: Grafik Penggunaan CPU pada Uji Coba Performa dan Skalabilitas



Gambar 5.5: Grafik Kehilangan Data pada Uji Coba Performa dan Skalabilitas

LAMPIRAN A

INSTALASI PERANGKAT LUNAK

A.1 Instalasi Node JS

Ada banyak cara untuk menginstall NodeJS ke dalam komputer, salah satunya melalui kode sumber yang dapat diunduh pada halaman `https://nodejs.org/en/download/`. Setelah diunduh berkas perlu diekstrak untuk melanjutkan instalasi. Di dalam folder hasil ekstraksi, terdapat berkas `README.md` yang berisi langkah untuk menginstall NodeJS ke komputer. Semua langkah dilakukan melalui *command line* dan langkah installasinya adalah sebagai berikut

- Pindah ke folder dimana Node JS di ekstrak dengan perintah `cd {$pathdownload}/node-v4.2.3/`.
- Di dalam folder jalankan perintah berikut secara berurutan `./configure`, `make`, `make install`.
- Jika terdapat pesan *error* mengenai beberapa pustaka yang belum terinstall sebelumnya, jalankan perintah `sudo apt-get install -f`.
- Semua perintah dilakukan pada hak akses *root*.

Pemasangan Node JS juga dapat dilakukan dengan menjalankan perintah `sudo apt-get install nodejs`. Agar mempermudah pemasangan pustaka-pustaka berbasis Node JS maka NPM(*Node Package Manager*) juga harus terpasang pada komputer. Untuk memasang paket NPM jalankan perintah `sudo apt-get install npm`.

A.1.1 Pemasangan kerangka kerja Expressjs

Setelah Node JS dan NPM berhasil dipasang, maka kerangka kerja Expressjs juga harus terpasang. Untuk memasang kerangka kerja Expressjs cukup menjalankan perintah `sudo npm install express --g`. Setelah kerangka kerja Expressjs berhasil terpasang, unduh kerangka kerja Expressjs yang telah dimodifikasi agar mendukung MVC(*Model View Controller*).

Untuk mengunduh template Express-MVC dapat dilakukan dengan menjalankan perintah `git clone https://github.com/thiar/ExpressMVC-Framework.git`. Setelah template baru berhasil di unduh, modifikasi berkas `package.json` menjadi A.1

```
{
  "name": "middleware",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.13.2",
    "bookshelf": "^0.9.1",
    "cassandra-driver": "^3.0.1",
    "cookie-parser": "~1.3.5",
    "debug": "~2.2.0",
    "express": "~4.13.1",
    "express-session": "^1.7.6",
    "express-socket.io-session": "^1.3.1",
    "hbs": "~3.1.0",
    "helmet": "^0.10.0",
    "knex": "^0.9.0",
    "moment": "~2.x.x",
    "morgan": "~1.6.1",
    "mysql": "~2.x.x",
    "sandbox": "^0.8.6",
    "serve-favicon": "~2.3.0",
    "sha1": "^1.1.1",
    "socket.io": "~1.x.x"
  },
  "description": "1. Templating engine using handlebars 2.
      Css preprocessors using Sass 3. Aplication route using
       Express 4. Model database using mysql driver",
  "main": "app.js",
  "devDependencies": {},
  "author": "",
  "license": "ISC"
```

```
33 }
```

Kode Sumber A.1: Isi dari berkas package.json

Setelah memodifikasi berkas `package.json` jalankan perintah `npm install` untuk mengunduh semua pustaka yang diperlukan. Setelah pemasangan selesai, konfirmasi keberhasilan pemasangan dengan menjalankan perintah `nodejs bin/www`.

A.2 Instalasi paket GO Lang

Dalam proses pengembangan, dibutuhkan paket kompilator GO Lang yang terpasang pada server guna mempermudah proses kompilasi kode sumber. Untuk memasang paket GO Lang dengan langkah-langkah sebagai berikut:

- Tambahkan repositori GO Lang pada *apt* dengan menggunakan perintah `sudo add-apt-repository ppa:ubuntu-lxc/lxd-stable`. Kemudian perbarui daftar paket *apt* menggunakan perintah `sudo apt-get update`.
- Untuk melakukan pemasangan kompilator GO Lang, jalankan perintah `sudo apt-get install golang`. Setelah proses pemasangan selesai, konfirmasi keberhasilan pemasangan menggunakan perintah `go version`
- Atur lokasi *GOPATH* untuk memudahkan penambahan pustaka berbasis GO Lang. Jalankan perintah `export GOPATH=$HOME/go_library` untuk mengatur lokasi *GOPATH* yang akan digunakan

A.3 Instalasi Lingkungan Kontainer Docker

Kontainer Docker digunakan sebagai lingkungan penyebaran komponen aplikasi. Mesin Docker yang digunakan adalah Docker versi 1.11.1.

A.3.1 Instalasi Mesin Docker

Tahap instalasi mesin docker adalah sebagai berikut:

- Masuk menggunakan hak akses *root* atau gunakan perintah *sudo*. Gunakan perintah `sudo apt-get update` untuk memperbarui repositori *apt*.
- Pastikan *apt* dapat berjalan menggunakan protokol *https* dengan menggunakan perintah `sudo apt-get install apt-transport-https ca-certificates`.
- Tambahkan kunci *GPG* baru dengan menggunakan perintah `sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80--recv-keys 58118E89F3A912897C070ADBF76221572C52609D` .
- Buka berkas `/etc/apt/sources.list.d/docker.list`, jika berkas tidak ada maka buatlah. Tambahkan baris sesuai pada Kode Sumber A.2.

```
1   deb https://apt.dockerproject.org/repo ubuntu-
        trusty main
```

Kode Sumber A.2: Isi Berkas docker.list

Kemudian simpan dan keluar dari berkas `/etc/apt/sources.list.d/docker.list`

- Perbarui paket **apt** menggunakan perintah `sudo apt-get update`.
- Pasang paket *kernel linux-image-extra* dengan menggunakan perintah `sudo apt-get install linux-image-extra-$(uname -r)`
- Pasang mesin Docker menggunakan perintah `sudo apt-get install docker-engine`. Kemudian jalankan mesin Docker menggunakan perintah `sudo service docker start`. Untuk mengecek mesin Docker sudah terpasang dengan benar jalankan perintah berikut `sudo`

```
docker run hello-world
```

A.3.2 Menjalankan Docker tanpa akses root

Karena menjalankan Docker dengan akses *root* sangat berbahaya, maka Docker harus dijalankan menggunakan *non-root user*. Untuk menjalankan Docker sebagai *non-root user*, jalankan perintah-perintah berikut :

- Buat grup baru untuk menjalankan Docker dengan menjalankan perintah `sudo groupadd docker`
- Tambahkan pengguna yang sedang aktif ke dalam grup Docker menggunakan perintah `sudo gpasswd -a $USER docker`
- Jalankan ulang Docker menggunakan perintah `sudo service docker restart`

A.3.3 Menjalankan Syslog parser di dalam mesin Docker

Ketika pengembangan telah selesai, akan lebih mudah jika aplikasi di paketkan ke dalam Docker untuk automasi proses *deployment*. Untuk membuat kontainer Docker yang telah terisi aplikasi beserta semua kebutuhannya, maka berkas `Dockerfile` harus dibuat. `Dockerfile` yang dibuat ada 2, yaitu `DockerfileUDP` untuk aplikasi yang berjalan pada protokol UDP dan `DockerfileTCP` untuk aplikasi yang berjalan pada protokol TCP. Berikut kode sumber `DockerfileUDP` A.3 dan kode porgram `DockerfileTCP` dan A.4

```
1 FROM golang
2
3 ADD app_udp.go /go/src/app_udp/app_udp.go
4
5 RUN go get github.com/jeromer/syslogparser/rfc3164
6 RUN go get github.com/jeromer/syslogparser/rfc5424
7
8 RUN go install app_udp
```

```
9  ENTRYPOINT /go/bin/app_udp
10
11 EXPOSE 9000/udp
```

Kode Sumber A.3: Isi dari berkas DockerfileUDP

```
1  FROM golang
2
3  ADD app_tcp.go /go/src/app_tcp/app_tcp.go
4
5  RUN go get github.com/jeromer/syslogparser/rfc3164
6  RUN go get github.com/jeromer/syslogparser/rfc5424
7
8  RUN go install app_tcp
9  ENTRYPOINT /go/bin/app_tcp
10
11 EXPOSE 9000/tcp
```

Kode Sumber A.4: Isi dari berkas DockerfileTCP

Untuk membuat *images* baru berdasarkan berkas `DockerfileTCP` dan `DockerfileUDP` perintah `"docker build -f DockerfileUdp -t app_udp ."` dan `"docker build -f DockerfileTcp -t app_tcp ."` harus dijalankan pada komputer yang telah terpasang mesin Docker. Untuk pemasangan mesin Docker dapat merujuk pada Bab 4.2. Setelah *images* baru terbentuk, buat dan jalankan kontainer yang berisi aplikasi Syslog parser dalam UDP dan Syslog parser dalam TCP menggunakan perintah `docker run -p 9000:9000/tcp --name tcp_syslog --restart=always -d app_tcp` dan `docker run -p 9000:9000/udp --name udp_syslog --restart=always -d app_udp`. Konfirmasi bahwa kontainer telah berjalan dengan menjalankan perintah `docker ps`. Perintah tersebut akan menampilkan semua kontainer yang sedang berjalan. Untuk mendapatkan keluaran dari aplikasi yang sedang berjalan di dalam Docker, dapat digunakan perintah `docker logs -f udp_syslog` dan `docker logs -f tcp_syslog`.

A.3.4 Menjalankan Cassandra di dalam mesin Docker

Membangun kontainer untuk menjalankan Cassandra terlebih dahulu harus mengunduh *images* Cassandra dengan menjalankan perintah `docker pull cassandra`. Untuk memastikan *images* Cassandra telah terinstall dengan benar, jalankan perintah berikut `docker run --name cassa -d cassandra:latest`.
Perintah tersebut akan menjalankan kontainer dengan nama cassa dan *images* Cassandra versi terbaru. Untuk menghentikan kontainer jalankan perintah `docker stop cassa`.

LAMPIRAN B

KODE SUMBER

B.1 Kode Sumber Model

B.1.1 Model Auth

Model Auth digunakan untuk melakukan autentikasi terhadap permintaan yang masuk ke dalam sistem. Kode sumber tertera pada B.1

```
1  'use strict';
2  var Bookshelf = require('../config/database/driver/mysql/
       mysqlORMdriver');
3  require('./ColumnMapping');
4  var Auth = Bookshelf.Model.extend({
5    tableName: 'auth',
6    column_mapping : function (){
7      return this.hasMany('ColumnMapping');
8    }
9  });
10
11 module.exports = Bookshelf.model('Auth', Auth);
```

Kode Sumber B.1: Kode sumber Model Auth

B.1.2 Model CassandraType

Model CasandraType digunakan untuk mendefinisikan tipe kolom yang dibutuhkan oleh Cassandra. Kode sumber CassandraType tertera pada B.2

```
1  'use strict';
2  var Bookshelf = require('../config/database/driver/mysql/
       mysqlORMdriver');
3  require('./ColumnMapping');
4  var CassandraType = Bookshelf.Model.extend({
5    tableName: 'cassandra_type',
6    column_mapping : function (){
7      return this.hasMany('ColumnMapping');
8    }
9  });
```

```
10
11 module.exports = Bookshelf.model('CassandraType',
       CassandraType);
```

Kode Sumber B.2: Kode sumber Model CassandraType

B.1.3 Model ColumnMapping

Model ColumnMapping digunakan untuk melakukan pemetaan pada tiap kolom yang didefinisikan di cassandra. Kode sumber ColumnMapping tertera pada B.3

```
1  'use strict';
2  var Bookshelf = require('../config/database/driver/mysql/
       mysqlORMdriver');
3  require('./Auth');
4  require('./DataMapping');
5  require('./CassandraType');
6  var ColumnMapping = Bookshelf.Model.extend({
7    tableName: 'column_mapping',
8    auth : function (){
9      return this.belongsTo('Auth','auth_id');
10   },
11   data_mapping : function (){
12     return this.hasMany('DataMapping');
13   },
14   cassandra_type: function (){
15     return this.belongsTo('CassandraType','column_type');
16   }
17 });
18
19 module.exports = Bookshelf.model('ColumnMapping',
       ColumnMapping);
```

Kode Sumber B.3: Kode sumber Model ColumnMapping

B.2 Kode Sumber Pengendali

```
1 var cassandra = require('cassandra-driver');
2 var contactPoints = ['10.151.36.96','10.151.36.97'
3 ,'10.151.36.98'];
4 var client = new cassandra.Client({ contactPoints:
      contactPoints, keyspace: db});
```

Kode Sumber B.4: Autentikasi pada Middleware

B.3 Kode Sumber Syslog Parser

Daftar kode sumber yang digunakan untuk membangun Syslog parser. Untuk menerjemahkan format Syslog dibutuhkan pustaka `syslogparser` yang dikembangkan oleh Jeromer. Untuk mengunduh pustaka syslogparser dapat menggunakan perintah `go get "github.com/jeromer/syslogparser/"`

B.3.1 Impor pustaka pada Golang

Beberapa pustaka harus di impor untuk mendukung berjalannya Syslog parser. Pustaka-pustaka yang diperlukan antara lain adalah pustaka net,os,syslogparser dan bytes.

```
1  package main
2
3  import (
4    "fmt"
5      "net"
6      "os"
7      "time"
8      "github.com/jeromer/syslogparser/rfc3164"
9      "github.com/jeromer/syslogparser/rfc5424"
10     "strings"
11     "encoding/json"
12     "net/http"
13     "io/ioutil"
14     "bytes"
15 )
```

Kode Sumber B.5: Impor pustaka dalam bahasa GO

B.3.2 Struktur Data Syslog

Pesan Syslog yang telah diterjemahkan akan di simpan dalam struktur data. Struktur data ini yang nantinya akan di konversi ke dalam format json untuk dikirim ke middleware.

```
type Message struct {
    Ip string `json:"ip"`
    Timestamp time.Time `json:"timestamp"`
    Hostname string `json:"hostname"`
    Content string `json:"content"`
    Facility int `json:"facility"`
    Tag string `json:"tag"`
    Priority int `json:"priority"`
    Severity int `json:"severity"`
}
```

Kode Sumber B.6: Pembuatan Struktur Data

B.3.3 Proses Penerjemahan Syslog

Proses penerjemahan Syslog dilakukan dengan cara melewatkan pesan Syslog ke pustaka syslogparser. Kemudian akan di ubah menjadi struktur data setelah melewati proses penerjemahan. Proses penerjemahan dapat terjadi sebanyak 2 kali karena pesan Syslog yang diterima diformat menggunakan rfc5424 atau rfc3164.

```

/*Penerjemahan pertama*/
p := rfc3164.NewParser(buff)
err = p.Parse()


if err != nil {
```

```
  /*penerjemahan kedua jika penerjemahan pertama gagal*/
    p := rfc5424.NewParser(buff)
    err = p.Parse()
    if err != nil {
      fmt.Println("Error parsing:", err.Error())
    }
}


ip := addr.String()
var timestamp time.Time
var hostname string
var content string
var facility int
var tag string
var priority int
var severity int
for k, v:= range p.Dump() {
    if k=="timestamp" {
      timestamp = v.(time.Time)
    }else if k=="hostname" {
      hostname = v.(string)
    }else if k=="content" {
      content = v.(string)
    }else if k=="facility"{
      facility = v.(int)
    }else if k=="tag"{
      tag = v.(string)
    }else if k=="priority" {
      priority = v.(int)
    }else if k=="severity" {
      severity = v.(int)
    }
}
```

Kode Sumber B.7: Penerjemahan pesan Syslog menjadi variabel dan nilai

B.3.4 Pembuatan Layanan UDP

Layanan UDP dijalankan menggunakan port 9000. Berikut definisi konstanta yang akan digunakan untuk membuat layanan UDP.

```
const (
    CONN_HOST = "0.0.0.0"
    CONN_PORT = "9000"
    CONN_TYPE = "udp4"
    CONN_TOKEN = "ce6125e7a68a0add"
    CONN_TARGET = "10.151.36.99"
    CONN_TARGET_PORT = "3000"
)
```

Kode Sumber B.8: Pembuatan variabel konstan pada protokol UDP

Penjelasan kode tersebut adalah sebagai berikut :

- CONN_HOST
  `CONN_HOST` adalah alamat IP dari komputer server tempat layanan berjalan.
- CONN_PORT
  `CONN_PORT` adalah nomor *port* dari komputer server tempat layanan berjalan.
- CONN_TYPE
  `CONN_TYPE` adalah tipe koneksi yang akan dibuat, karena kita akan membuat koneksi dalam protokol udp pada IP versi 4 maka digunakan `udp4`
- CONN_TOKEN
  Untuk melakukan penyimpanan data pada klaster Cassandra, Middleware memerlukan `TOKEN` yang valid agar tidak semua orang dapat melakukan penyimpanan data. `CONN_TOKEN` adalah `TOKEN` yang valid untuk proses autentikasi oleh Middleware.
- CONN_TARGET
  `CONN_Target` adalah alamat IP dari Middleware.

- CONN_TARGET_PORT `CONN_TARGET_PORT` adalah nomor *port* dari Middleware.

B.3.5 Membuka Layanan UDP

Layanan UDP dibuka dengan membuka koneksi port yang telah di definisikan sebelumnya.

```
1 ServerAddr,err := net.ResolveUDPAddr(CONN_TYPE,CONN_HOST + "
      :" + CONN_PORT)
2 CheckError(err)
3
4 ServerConn, err := net.ListenUDP("udp", ServerAddr)
5 CheckError(err)
6 defer ServerConn.Close()
7 fmt.Println("Listening on " + CONN_HOST + ":" + CONN_PORT)
```

Kode Sumber B.9: Membuka koneksi UDP

B.3.6 Menerima Pesan Melalui Layanan UDP

Pesan yang akan diterima dimasukkan ke dalam variabel `buf`. Variabel `buf` kemudian akan di olah menggunakan fungsi penerjemahan pada kode sumber B.7.

```
1 n,addr,err := ServerConn.ReadFromUDP(buf)
2 b := string(buf[0:n])
3 s := strings.Split(b,"\n")
```

Kode Sumber B.10: Menerima pesan melalui protokol UDP

B.3.7 Membuat Layanan TCP

Pembuatan layanan TCP sesuai dengan layanan UDP. Perbedaannya terdapat fungsi `accept()` yang digunakan untuk menerima koneksi dan memulai pertukaran data.

```
l, err := net.Listen(CONN_TYPE, CONN_HOST+":"+CONN_PORT)
if err != nil {
    fmt.Println("Error listening:", err.Error())
    os.Exit(1)
}

defer l.Close()
fmt.Println("Listening on " + CONN_HOST + ":" + CONN_PORT)
for {
  conn, err := l.Accept()
  if err != nil {
    fmt.Println("Error accepting: ", err.Error())
    os.Exit(1)
  }

  go handleRequest(conn)
}
```

Kode Sumber B.11: Membuka koneksi TCP

B.3.8 Menerima Pesan Melalui Layanan TCP

Layanan TCP dijalankan menggunakan port 9000. Berikut definisi konstanta yang akan digunakan untuk membuat layanan TCP.

```
const (
CONN_HOST = "0.0.0.0"
CONN_PORT = "9000"
CONN_TYPE = "tcp"
CONN_TOKEN = "ce6125e7a68a0add72517e49591c938535e2c139"
CONN_TARGET = "10.151.38.181"
CONN_TARGET_PORT = "3000"
)
```

Kode Sumber B.12: Pembuatan variabel konstan pada protokol TCP

Setelah koneksi diterima melalui fungsi `accept()`. Pesan yang akan diterima dimasukkan ke dalam variabel `buf`. Variabel

`buf` kemudian akan di olah menggunakan fungsi penerjemahan pada kode sumber B.7

```
n, err := conn.Read(buf)
if err != nil {
  fmt.Println("Error reading:", err.Error())
}
b := string(buf[0:n])
s := strings.Split(b,"\n")
```

Kode Sumber B.13: Menerima pesan melalui protokol TCP

B.3.9 Mengubah Data Syslog Kedalam Format JSON

Setelah penerjemahan berhasil dan struktur data terbentuk. Struktur data akan di format kedalam bentuk JSON untuk dikirimkan ke middleware.

```

m := new(Message)
m.Ip=ip
m.Timestamp=timestamp
m.Hostname=hostname
m.Content=content
m.Facility=facility
m.Tag=tag
m.Priority=priority
m.Severity=severity
data, err := json.Marshal(m)
```

Kode Sumber B.14: Mengubah data hasil terjemahan ke dalam format JSON

B.3.10 Pengiriman Data JSON ke Middleware

Setelah daata JSON terbentuk, pengiriman dilakukan. Pengiriman dilakukan dengan menggnakan metode POST. Pengiriman dilakukan dengan mengakses rute `/parser/:token/:db` pada middleware.

```
if err == nil {
    url := "http://"+CONN_TARGET+":"+CONN_TARGET_PORT+"/
        parser/"+CONN_TOKEN+"/syslog"
    // fmt.Println(string(data[:]))
    req, err := http.NewRequest("POST", url, bytes.NewBuffer
        (data))
    req.Header.Set("Content-Type", "application/json")

    client := &http.Client{}
    resp, err := client.Do(req)
    if err != nil {
        fmt.Println(err)
    } else {
        fmt.Println("response Status:", resp.Status)
        fmt.Println("response Headers:", resp.Header)
        body, _ := ioutil.ReadAll(resp.Body)
        fmt.Println("response Body:", string(body))
        defer resp.Body.Close()
    }

}
```

Kode Sumber B.15: Mengirim data JSON ke Middleware

B.4 Kode Sumber Middleware

B.4.1 Koneksi Basis Data Mysql

```
'use strict';
var knex = require('knex')({
client: 'mysql',
connection: {
host     : 'localhost',
user     : 'root',
password : 'kucinglucu',
database : 'middleware',
charset  : 'utf8'
}
});
```

```
12
13 var bookshelf = require('bookshelf')(knex);
14 bookshelf.plugin('registry');
15 module.exports=bookshelf;
```

Kode Sumber B.16: Isi dari berkas mysqlORMdriver.js

B.4.2 Proses Autentikasi

```
1 Auth.query({where:where}).fetch().then(function(data){
2 if(data){
3 /*Kode setelah autentikasi akan berjalan */
4 }
5 else res.json({status:'error',message:'nama database dan
      token tidak cocok'})
6 }).catch(function(err){
7 console.log(err)
8 })
```

Kode Sumber B.17: Autentikasi pada Middleware

B.5 Penyeimbang Beban

```
1  import sys
2  import signal
3  import logging
4  from socket import *
5
6  LB_NODES = ['10.151.36.40','10.151.36.41']
7  LB_PORT = 9000
8
9  affinityTable = {}
10 _next = 0
11
12
13 def get_next_node(srcIP):
14 global affinityTable, _next
15 if srcIP not in affinityTable:
```

```
16 _next = (_next + 1) % len(LB_NODES)
17 affinityTable[srcIP] = _next
18 print _next
19 _next = (_next + 1) % len(LB_NODES)
20 return LB_NODES[_next]
21
22
23 def main():
24 def signal_handler(signal, frame):
25 logging.info('Interrupted, shutting down')
26 s1.close()
27 s2.close()
28
29 signal.signal(signal.SIGINT, signal_handler)
30
31 logging.basicConfig(format='%(asctime)s [%(levelname)s] - %(
       message)s',
32 datefmt='%m/%d/%Y %I:%M:%S %p',
33 filename='udplb.log', level=logging.INFO)
```

Kode Sumber B.18: Kode Sumber Penyeimbang beban

B.6 Kode Sumber Pengujian Menu Fungsi

```
1 /*data adalah variabel yang menyimpan \emph{log} untuk di
      proses*/
2 for(i=0;i<data[0].length;i++)
3 {
4 if(data[0][i].severity==6 && data[0][i].hostname=='
      hanacaraka')console.log(data[0][i].hostname + " " + data
      [0][i].content)
5 }
```

Kode Sumber B.19: Kode sumber mencari data dengan severity dan hostname tertentu

```
1 /*Kode sumber akan menambahkan berkas baru bernama malware
      pada direktori plugin yang berada di server Middleware*/
2 var fs = require('fs');
```

```
3  fs.writeFile("plugin/malware", "Kode Berbahaya!!", function(
       err) {
4  if(err) {
5  return console.log(err);
6  }
7  console.log("The file was saved!");
8  });
9  fs.readdir('plugin', function (err, files) {
10 if (!err)
11 console.log(files);
12 else
13 throw err;
14         });
```

Kode Sumber B.20: Kode sumber file inclusion

Keluaran kode sumber B.19 tertera pada B.21. Sedangkan keluaran dari kode berbahaya B.20 tertera pada B.22

```
1  hanacaraka DHCPDISCOVER on eth1 to 255.255.255.255 port 67
       interval 12 (xid=0x26ac0d33)
2  hanacaraka pam_unix(samba:session): session closed for user
       nobody
3  hanacaraka No working leases in persistent database -
       sleeping.
```

Kode Sumber B.21: Keluaran Kode sumber mencari data dengan severity dan hostname tertentu

```
1  ReferenceError: require is not defined
```

Kode Sumber B.22: Keluaran kode sumber file inclusion

BAB 6

PENUTUP

Bab ini membahas kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hubungannya dengan hasil uji coba dan evaluasi yang telah dilakukan. Selain itu, terdapat beberapa saran yang bisa dijadikan acuan untuk melakukan pengembangan dan penelitian lebih lanjut.

6.1 Kesimpulan

Dari proses perancangan, implementasi dan pengujian terhadap sistem, dapat diambil beberapa kesimpulan berikut:

1. Sistem dapat menyimpan Syslog dari berbagai perangkat secara bersamaan. Sistem operasi yang digunakan perangkat tidak berpengaruh terhadapap keberhasilan pencatatan Syslog. Perangkat yang mendukung protokol Syslog dapat mengirimkan pesan Syslog untuk disimpan oleh sistem. Data Syslog yang disimpan dapat ditampilkan kembali dalam bentuk grafik untuk proses analisis data syslog.
2. Dengan dukungan klaster Cassandra dan Docker sebgai media penyebaran, sistem akan memiliki skalabilitas. Sistem dapat dengan mudah diperbesar ketika permintaan penyimpanan data semakin tinggi.

6.2 Saran

Berikut beberapa saran yang diberikan untuk pengembangan lebih lanjut:

- Sistem sudah mendukung skalabilitas. Pengembang bisa dengan mudah membuat penyimpanan data baru dan *parser* yang sesuai. Sistem dapat dikembangkan untuk mencatat *log* yang menggunakan protokol-protokol lain

selain syslog. Beberapa protokol yang sesuai untuk membantu pengelolaan jaringan antara lain:

  - SNMP (Simple Network Management Protokol)
  - RELP (Reliable Event Logging Protocol)
  - Log4J

- Untuk menangani permintaan yang banyak dan bersamaan, komputer server yang digunakan untuk menjalankan sistem harus diperbanyak. Penggunaan algoritma pembagi beban juga dapat menunjang sistem agar dapat melayani lebih banyak perangkat lagi.

DAFTAR PUSTAKA

[1] M. Schütte, **Syslog Protocols**, Potsdam: universität potsdam, 2008.

[2] Joyent, Inc, **About Node.js**, [Online], `https://nodejs.org/en/about/`, diakses tanggal 01 April 2016

[3] S. Tilkov and S. Vinoski, **Node.js: Using JavaScript to Build High-Performance Network Programs**, IEEE Computer Society Issue, vol. 6, pp. 80-83, 2010

[4] A. Chebotko, A. Kashlev and S. Lu, **"A Big Data Modeling Methodology for Apache Cassandra,"** IEEE International Congress on Big Data, vol. I, pp. 1-8, 2015.

[5] D. CORPORATION, **"Introduction to Apache Cassandra,"** pp. 1-11, 2013.

[6] Jeffrey D. Walker, Steven C. Chapra, **A client-side web application for interactive environmental simulation modeling**, Environmental Modelling & Software, vol. 55, pp. 49-60, 2014

[7] Google, **Angular JS**, [Online], `https://angularjs.org/`, diakses tanggal 07 April 2016

[8] Philipp Hoenisch, Ingo Weber, **Four-fold Auto-scaling on a Contemporary Deployment Platform using Docker Containers**,NICTA, 2015

[9] Docker, Inc, **Modern App Architecture for the Enterprise Delivering agility, portability and control with Docker Containers as a Service (CaaS)**, `https://www.docker.com/sites/default/files/WP_ModernAppArchitecture_04.12.2016_1.pdf`, diakses 12 Mei 2016

BIODATA PENULIS

Thiar Hasbiya Ditanaya, Akrab dipanggil Thiar lahir di Muara Bulian pada tanggal 14 maret 1994. Penulis adalah anak pertama dari dua bersaudara. Penulis menempuh pendidikan formal di SDN Wates III Mojokerto, SMPN 2 Mojokerto, SMAN 1 Sooko Mojokerto dan S1 Teknik Informatika FTIF ITS. Penulis adalah pengembang aplikasi berbasis web. Sebagai seorang pengembang, penulis juga harus selalu berkembang. Beberapa aplikasi berbasis web yang pernah di kembangkan penulis adalah kerangka kerja Express-MVC, web penerimaan peserta didik baru sidoarjo dan web kenaikan pangkat online dinas pendidikan surabaya. Penulis adalah asisten Lab pada Laboratorium Arsitektur dan Jaringan Komputer. Selama menjadi asisten Lab penulis beberapa kali menjadi asisten dosen dan praktikum pada mata kuliah sistem operasi, jaringan komputer, keamanan informasi jaringan dan komputasi awan. Penulis senang mempelajari hal baru terkait teknologi informasi. Beberapa topik yang sedang di dalami adalah topik terkait Aplikasi Berbasis Web, Komputasi Awan, dan Teknologi Pada Jaringan. Penulis dapat dihubungi melalui surat elektronik `thiar.hasbiya@gmail.com`.