Rancang Bangun Sistem Log Server Berbasis Syslog dan Cassandra untuk Monitoring Pengelolaan Jaringan di ITS

Thiar Hasbiya Ditanaya, Royyana Muslim Ijtihade, dan Muchammad Husni
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: roy@if.its.ac.id

***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 system 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.**

***Kata Kunci*— Syslog, Cassandra, Scalability, NodeJS.**

I. PENDAHULUAN

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 system administrator harus memeriksa pesan *log* semua perangkat komputernya.

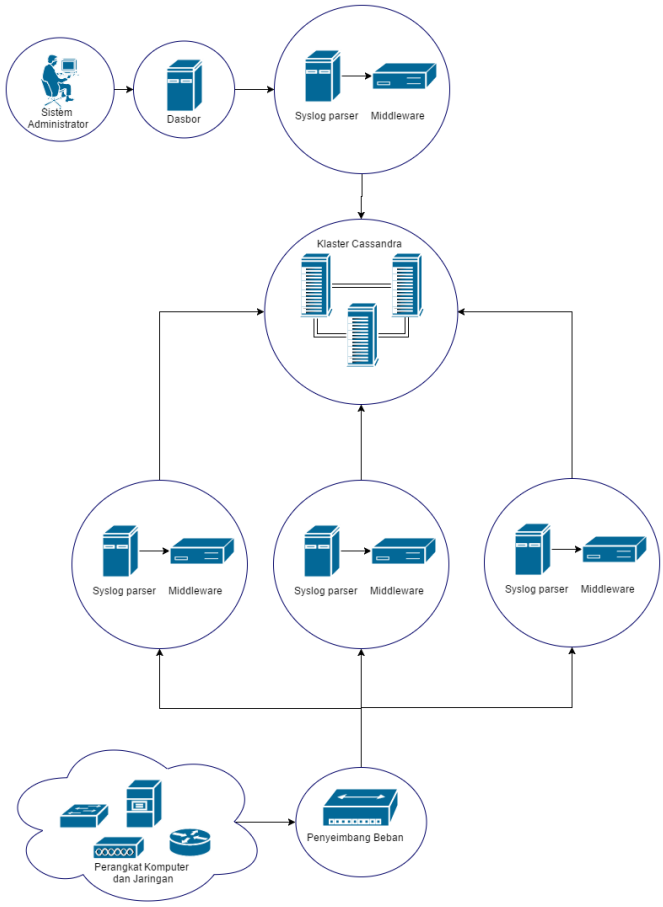
Syslog server adalah sebuah server yang menyimpan data syslog berbagai macam perangkat komputer dan jaringan secara terpusat. Syslog server harus memiliki ketersediaan tinggi untuk melayani penyimpanan syslog setiap perangkat komputer dan jaringan.

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. Makalah ini membahas tentang penyimpanan realtime yang memiliki ketersediaan tinggi pada syslog server yang dapat melayani setiap permintaan pencatatan log secara realtime dan menampilkan kembali log tersebut kepada sistem administrator secara mudah dan cepat.

II. PERANCANGAN

A. Desain Arsitektur Sistem

Sistem *log* server mendukung skalabilitas dengan membagi arsitektur menjadi komponen-komponen kecil yang masing-masing dapat dikembangkan tersendiri. Komponen-komponen yang digunakan adalah penyeimbang beban, penyimpanan data, *middleware*, dasbor dan parser. Komponen-komponen tersebut dipisahkan dan di *bundle* menggunakan Docker kontainer.



Gambar. 1. Arsitektur Sistem

B. *Syslog*

Syslog adalah protokol yang digunakan untuk mengirimkan pesan *logging* pada komputer dan perangkat jaringan. *Syslog* yang dikirim sesuai format RFC3164 dan RFC5424. *Syslog* digunakan pada perangkat komputer dengan sistem operasi linux dan windows dan pada perangkat jaringan *router* dan akses poin [1].

C. *Penyeimbang Beban*

Penyeimbang beban membagi beban yang akan diterima sistem secara bergantian menggunakan metode *round-robin*. Penyeimbang beban menyembunyikan komponen-komponen sistem yang lain. Perangkat komputer dan jaringan mengirim pesan *syslog* melalui penyeimbang beban yang kemudian akan diteruskan menuju *middleware*.

D. *Middleware*

Middleware adalah penghubung akses menuju penyimpanan data. *Middleware* membatasi akses menggunakan *token. Token* yang digunakan unik untuk setiap penyimpanan data yang ada. *Token* adalah gabungan dari nama penyimpanan data dan nama pengelola yang telah disandikan menjadi satu baris huruf. *Middleware* dibangun menggunakan *Node Js* agar dapat melayani permintaan dengan lebih cepat. *Node Js* menggunakan *asynchronous call* untuk memproses semua baris perintah, sehingga memungkinkan pekerjaan dapat dilakukan secara konkuren. Pekerjaan yang dilakukan secara konkuren dapat meningkatkan performa sistem dalam melayani permintaan [4].

E. *Penyimpanan Data*

Penyimpanan data adalah klaster Cassandra yang digunakan untuk menyimpan *syslog* dari setiap perangkat. Jumlah *node* Cassandra yang digunakan adalah 3 buah. Klaster yang digunakan memiliki konsistensi level *quorum* dan *replication factor* 3. Setiap *node* Cassandra yang digunakan memiliki memori sebesar 1 GB. *Node* yang tidak aktif atau mati akan otomatis dikeluarkan dari klaster hingga *node* dapat aktif kembali [2] [3].

F. *Syslog Parser*

Syslog parser menerjemahkan pesan *syslog* dalam format RFC3164 dan RFC5424 menjadi kata kunci dan nilai. Kata kunci dan nilai kemudian akan disimpan ke penyimpanan data Cassandra. *Syslog parser* berjalan pada protokol TCP dan UDP.

G. *Dasbor*

Sistem administrator berinteraksi dengan sistem melalui dasbor. Dasbor memiliki fungsi pengelolaan penyimpanan data, grafik kejadian syslog dan pembuatan fungsi baru.

H. *Grafik kejadian Syslog*

Grafik kejadian syslog adalah salah satu fitur pada aplikasi dasbor. Grafik kejadian *syslog* menampilkan kejadian *syslog* berdasarkan waktu kejadian. Grafik menampilkan kejadian *syslog* dengan pengelompokan berdasarkan *severity*. Grafik kejadian *syslog* digunakan untuk keperluan analisis kinerja perangkat melalui pesan *syslog* yang terjadi.

III. UJI COBA DAN EVALUASI

Pengujian dilakukan dengan beberapa komputer pekerja untuk melihat perbandingan performa sistem dengan 1,2 dan 3 komputer pekerja. Pengujian yang dilakukan adalah penggunaan CPU dan kehilanan data atau data *lost* pada sistem dengan 1,2 dan 3 komputer pekerja. Pengujian dilakukan pada jaringan teknik informatika ITS menggunakan 6 komputer penguji dengan konkurensi masing-masing penguji adalah 100,200,300,400 dan 500.

A. *Penggunaan CPU*

Penggunaan CPU diukur pada setiap uji coba untuk sistem dengan 1,2 dan 3 komputer pekerja. Penggunaan CPU di uji dengan mengirimkan pesan syslog menggunakan komputer penguji. Pengiriman pesan menggunakan konkurensi dengan thread 100,200,300,400 dan 500. Thread adalah pekerjaan yang dilakukan secara bersamaan pada satu proses yang sama. Menggunakan banyak thread dapat menguji ketahanan sistem dalam menangani permintaan yang konkuren. Pada setiap ujicoba akan di evaluasi kenaikan penggunaan CPU pada tiap komputer pekerja. Kenaikan penggunaan CPU dapat mempengaruhi performa sistem. Hasil pengujian ditunjukkan

dalam tabel 1,tabel 2 dan tabel 3. Grafik penggunaan CPU ditunjukkan pada Gambar 2.

Jumlah Thread	Jumlah data dikirim	Jumlah data masuk	Penggunaan CPU
100	15000	14911	87,5%
200	30000	24768	94,5%
300	45000	29782	96%
400	60000	27769	100%
500	75000	38932	100%

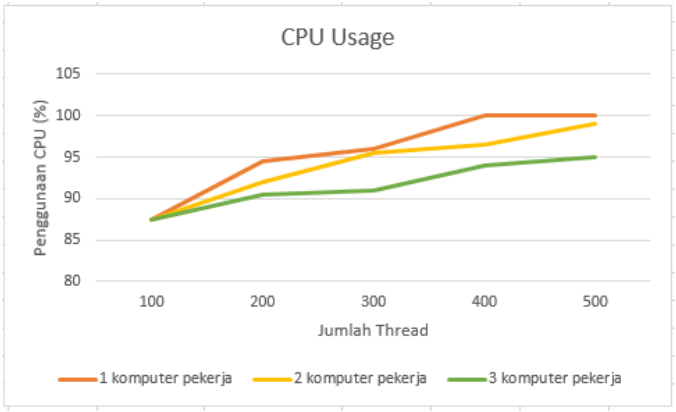
Tabel 1. Hasil Uji Coba 1 komputer pekerja

Jumlah Thread	Jumlah data dikirim	Jumlah data masuk	Penggunaan CPU
100	15000	14911	87,5%
200	30000	24768	92%
300	45000	29782	95,5%
400	60000	27769	96,5%
500	75000	38932	99%

Tabel 2. Hasil Uji Coba 2 komputer pekerja

Jumlah Thread	Jumlah data dikirim	Jumlah data masuk	Penggunaan CPU
100	15000	14911	87,5%
200	30000	24768	90,5%
300	45000	29782	91%
400	60000	27769	94%
500	75000	38932	95%

Tabel 3. Hasil Uji Coba 3 komputer pekerja



Gambar 2. Grafik penggunaan CPU

B. *Data Lost*

Data *lost* terjadi pada pengiriman *syslog* dengan protokol UDP. Penambahan komputer pekerja akan mengurangi data *lost.* Uji coba data *lost* dilakukan dengan 6 komputer penguji menggunakan konkurensi dengan thread 100,200,300,400 dan 500. Pada setiap ujicoba thread akan dievaluasi banyaknya data yang hilang. Prosentase data yang hilang didapatkan dari jumlah data yang dikirimkan oleh komputer penguji dibandingkan dengan jumlah data masuk pada penyimpanan data. Hasil pengujian ditunjukkan dalam tabel 4,tabel 5 dan tabel 6. Grafik data *lost* ditunjukkan pada Gambar 3.

Jumlah Thread	Jumlah data dikirim	Jumlah data masuk	Data *lost*
100	15000	14911	15,95%
200	30000	24768	46,77%
300	45000	29782	81,79%
400	60000	27769	89,56%
500	75000	38932	87,99%

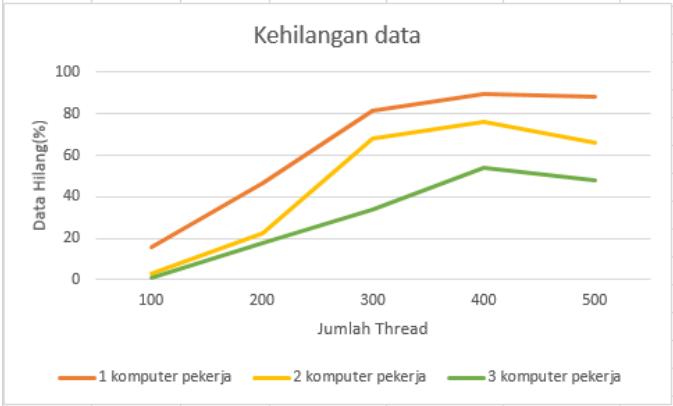
Tabel 4. Hasil Uji Coba 1 komputer pekerja

Jumlah Thread	Jumlah data dikirim	Jumlah data masuk	Data *lost*
100	15000	14911	3,12%
200	30000	24768	22,19%
300	45000	29782	67,80%
400	60000	27769	75,96%
500	75000	38932	66,14%

Tabel 5. Hasil Uji Coba 2 komputer pekerja

Jumlah Thread	Jumlah data dikirim	Jumlah data masuk	Data *lost*
100	15000	14911	0,59%
200	30000	24768	17,44%
300	45000	29782	33,82%
400	60000	27769	53,72%
500	75000	38932	48,09%

Tabel 6. Hasil Uji Coba 3 komputer pekerja



Gambar 3. Grafik penggunaan kehilangan data

IV. KESIMPULAN DAN SARAN

A. *Kesimpulan*

Kesimpulan yang dapat diambil dari uji coba dan evaluasi adalah sebagai berikut:

1. Sistem dapat menyimpan Syslog dari berbagai perangkat secara konkuren. 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 arsitektur sistem, sistem akan memiliki skalabilitas. Sistem dapat dengan mudah diperbesar ketika permintaan penyimpanan data semakin tinggi.

B. Saran

1. Penggunaan algoritma pembagi beban berbasis prioritas dapat menunjang sistem agar dapat melayani lebih banyak perangkat lagi.
2. 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:
   a. SNMP (Simple Network Management Protokol)
   b. RELP (Reliable Event Logging Protocol)
   c. Log4J

DAFTAR PUSTAKA

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

[2] D. CORPORATION, “Introduction to Apache Cassandra,” pp. 1-11, 2013.

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

[4] S. Tilkov dan S. Vinoski, “Node.js: Using JavaScript to Build High-Performance Network Programs,” IEEE Computer Society Issue, 2010, pp. 80-83.

[5] Docker, Inc, “Modern App Architecture for the Enterprise Delivering agility, portability and control with Docker Containers as a Service (CaaS),” Docker, Inc;, [Online]. Available: https://www.docker.com/ sites/default/files/ WP_ModernAppArchitecture_04.12.2016_1.pdf. [Diakses 12 6 2016].

[6] H. Philipp dan W. Ingo, “Four-fold Auto-scaling on a Contemporary,” no. NICTA, 2015.