Analisis Implementasi *Load Balancing* dengan Metode *Source Hash Scheduling* pada *Protocol SSL*

Raden Arief Setyawan

***Abstract*—The Course programming period give the highest load for academic server (SIAM) of UB. More than 65000 student will access the system concurrently. There for, the load balancing mechanism is required to improve system capacity and prevent the access failure**

SIAM using SSL mechanism to provide protection for academik data transaction. SSL do the handshaking process to maintain the connectivity of the web browser. In addition, the client and the server will establish a session recording mechanism to keep the identity of a connection to prevent repeat login. This study tried to implement a source-hash scheduling mechanism on the load balancing system. This mechanism subjected to prevent the termination of a session connection which has been formed.

The results shows that the source hash scheduling has increased the capacity of the system to handle as many as 9.02594 million requests from 65 087 different IP within 1 day. And provide total data throughput of 169 537 010 395 Bytes (169 GB) in a single day get

***Index Terms*—Load Balancing, Source Hash Scheduling**

***Abstract*—Perode KRS memberikan beban puncak bagi server akademik (SIAM) UB. Lebih dari 65000 mahasiswa akan mengakses sistem secara bersamaan. Untuk itu diperlukan mekanisme load balancing untuk meningkatkan kapasitas sistem sehingga mengurangi kegagalan akses.**

SIAM menggunakan mekanisme SSL untuk memberikan proteksi bagi transaksi akademik. SSL melakukan proses handshaking dengan web browser untuk memelihara suatu koneksi. Selain itu client dan server akan membentuk mekanisme pencatatan sesi untuk menjaga identitas sebuah koneksi untuk mencegah login berulang. Penelitian ini mencoba menerapkan mekanisme source hash scheduling pada sistem load balancing. Penerapan source hash ditujukan untuk mencegah terjadinya pemutusan koneksi sesi yang telah terbentuk.

Hasil pengujian menunjukkan bahwa source hash scheduling telah meningkatkan kapasistas sistem sehingga dapat menangani sebanyak 9.025.940 request dari 65.087 IP berbeda dalam 1 hari. Serta memberikan total data troughput sebesar 169.537.010.395 Byte (169 GB) dalam satu hari.

***Kata Kunci* —Load Balancing, Source Hash Scheduling**

Raden Arief Setyawan, dosen Teknik Elektro, Universitas Brawijaya, Malang, Indonesia (Telp.0341-554166; e-mail: rarief@ub.ac.id).

I. PENDAHULUAN

Peningkatan jumlah mahasiswa di Universitas Brawijaya menambah beban pada sistem informasi akademik mahasiswa. Dengan total mahasiswa sejumlah 65000, menyebabkan akses KRS mencapai titik kritis. Pada saat proses pemrograman kuliah, seluruh mahasiswa berlomba untuk memperoleh kelas dengan waktu yang favorit. Hal ini menyebabkan sebagian besar mahasiswa berusaha untuk mengakses pada detik pertama saat waktu pemilihan dibuka. Peningkatan akses secara drastis telah menyebabkan berbagai kendala seperti: kelambatan akses, kegagalan koneksi, server yang tidak responsif dan lain sebagainya.

Penelitian sebelumnya telah mencoba menyelesaikan permasalahan ini dengan menggunakan mekanisme load balancing [1]. Proses ini telah berhasil meningkatkan kualitas layanan untuk sejumlah 45000 mahasiswa. Namun penelitian tersebut belum menggunakan standar enkripsi SSL, sehingga keamanan akses ke sistem informasi belum dijamin. Penelitian ini mencoba menguji performansi implementasi load balancing SSL menggunakan metode source Hash pada Sistem Informasi Akademik Mahasiswa UB.

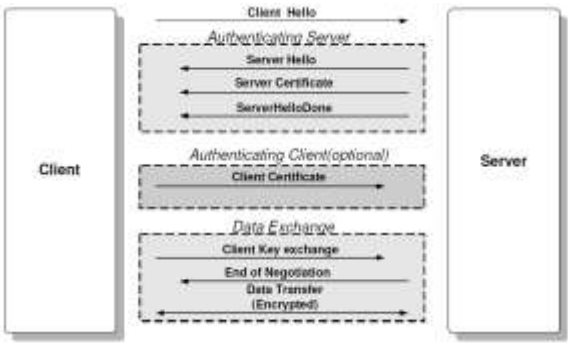
SSL Merupakan salah satu metode enkripsi standar pada website untuk mengamankan pengiriman data yang bersifat privat/rahasia seperti username dan password. Authentifikasi akses menjadi sangat penting saat suatu sistem informasi merupakan layanan privat bagi seorang user. Kebocoran informasi username dan password dapat menyebabkan berbagai permasalahan seperti ketidak otentikan data, pencurian data, perubahan data dan lain sebagainya. Proses enkripsi dan dekripsi data pada protokol SSL terjadi pada end to end system, yaitu pada server dan client. Untuk itu diperlukan kunci khusus untuk komunikasi agar sebuah sesi pada webserver dibuka untuk user tersentu. Setelah sebuah sesi terbuka, seluruh proses akan dianggap dilakukan oleh seorang user hingga sesi tersebut ditutup. Penutupan sebuah sesi dapat terjadi secara disengaja, melalui proses logout, atau tanpa sengaja karena terputusnya koneksi. Mekanisme load balancing menggunakan metode round robin [1] standar tidak memperdulikan sesi koneksi. Jika sebuah client telah terhubung ke server tertentu, dan tiba-tiba oleh load balancer dipindahkan ke server yang lain akan menyebabkan terputusnya sesi yang telah terbentuk.

Sehingga menyebabkan seorang user harus melakukan autentifikasi/login ulang karena identitasnya tidak dikenali oleh server yang baru. Oleh karena itu dalam penelitian ini mekanisme Source Hash Scheduling dipilih untuk menjaga agar perangkat load balancing akan menjaga koneksi dengan memperhatikan IP address client.

II. TEORI

A. SSL

SSL merupakan salah satu protokol paling populer untuk menjamin kerahasiaan data dengan melakukan enkripsi data secara antara client dan server. SSL beroperasi diantara layer HTTP dan layer Transmission Control Protocol (TCP). Terutama bagi web server/data center dengan aplikasi e-commerce menggunakan SSL untuk menjamin keamanan data [2]. Proses enkripsi data ini memberikan dua layanan yaitu proses enkripsi daya dan pesan singkat. Enkripsi data dilakukan melalui algoritma kunci simetris, seperti Triple-DES atau RC4. Pesan yang dikirimkan mengandung informasi tentang integritas data yang diuji melalui kunci Message Authentication Code (MAC). Fungsi Hash yang digunakan antara lain Secure Hash Algorithm Version l.0 (SHA1) or Message Digest 5 (MD5) digunakan untuk perhitungan MAC. Sebuah client yang menginisasi koneksi ke sebuah server dengan mengirimkan pesan "Hello" yang berisi informasi seperti Session ID, angka acak, cipher, dan informasi lainnya. Proses komunikasi SSL ditunjukkan dalam Gambar 1.

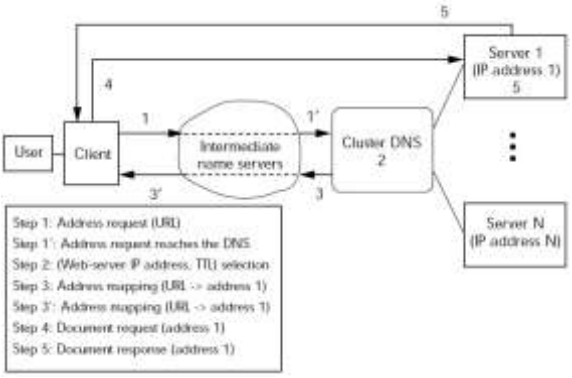


Gambar 1 Proses Handshaking SSL

Setelah menerima pesan "Hello" dari client, server akan mengirimkan pesan "Hello" balasan yang berisi informasi sertifikat dan informasi lainnya. Dengan menggunakan sertifikat dari server, client menyelesaikan proses authentikasi dengan server. Jika diminta, client diharuskan untuk mengirimkan sertifikat tersebut ke server untuk proses verifikasi. Setelah proses autentikasi, client membuat kunci sesi untuk proses enkripsi dan dekripsi suatu data. Sesi di identifikasi oleh session ID yang diketahui oleh client dan server. Selama sesi berlangsung, server menyimpan sementara informasi sesi dari client [3]. Jika client meminta koneksi baru pada sesi yang sama, maka server menggunakan kembali informasi sesi yang telah disimpan untuk membangkitkan sepasang kunci untuk sesi baru guna mengurangi waktu yang diperlukan untuk melakukan inisialisasi koneksi SSL.

B. Load Balancing

Load balancing merupakan salah satu mekanisme untuk membagi beban komputasi ke beberapa server. Load balancing bertujuan untuk mengoptimalkan sumber daya, memaksimal-kan troughput, meminimkan waktu respon, dan menghindari pembebanan berlebihan di satu sumber daya. Menggunakan beberapa sumber daya komputasi juga dapat mengurangi kemungkinan tidak berfungsinya suatu layanan karena setiap sumber daya dapat saling menggantikan (redundant).



Gambar 2 Proses mekanisme Balancing dengan basis DNS [4]

Sistem Load Balancing terdiri dari Virtual server dan real server. Virtual server sekaligus berfungsi sebagai perangkat yang membagi beban ke server sebenarnya. Virtual Server akan dituju oleh akses client. Saat sebuat permintaan dari client diterima oleh virtual server, akan diteruskan ke server sesungguhnya berdasarkan data beban masing-masing server. Secara umum server yang dituju adalah server dengan beban paling sedikit. Mekanisme pemilihan server inilah yang terdiri dari beberapa mekanisme seperti [5]:

a. least connection: mengutamakan real server dengan koneksi paling sedikit
b. round robin: menempatkan semua real-Server pada antrian yang melingkar dan mengalokasikan koneksi bergantian untuk setiap turn
c. destination hash: Menggunakan data tabel statis dari alamat IP tujuan untuk mengalokasikan koneksi
d. source hash scheduling: Menggunakan data tabel statis dari alamat IP asal (client) untuk mengalokasikan koneksi

Mekanisme source hash scheduling akan mengelola sebuah data yang berisi alamat IP client yang mengakses sebuah web. IP client akan dipetakan ke sebuah server tertentu dalam waktu tertentu. Selama ada transaksi antara server dan client, maka Virtual server akan menjaga koneksi tersebut dan mencegah untuk memindahkan ke server yang lain.

C. Web Server Session

Sesi pada webserver digunakan sebagai identitas bahwa sebuah koneksi telah terbangun. Hal ini berfungsi untuk menjaga koneksi identitas pengguna.

Terdapat dua mekanisme menjaga agar sebuah sesi tetap terjaga, yaitu server side session, serta client side session. Server side session, berarti bahwa server yang bertugas untuk menjaga agar identitas dari client terjamin. Server Server-side sessions mudah di implementasikan dan cukup efisien, namun implementasi server side session sulit ditangani pada mekanisme load balancing atau high availability. Mengingat setiap client yang sama tidak selalu di layani oleh server yang sama. Perbedaan ini menyebabkan hilangnya sesi koneksi antara client dan web server. Koneksi ini menyebabkan seorang user yang telah login kehilangan sesi loginnya, dan diminta untuk login kembali. Untuk itu diperlukan mekanisme khusus pada proses load balancing agar server mengetahui status sebuah sesi yang di inisialisasi oleh client. Sehingga akses dari sebuah client yang telah terbentuk koneksinya tidak dialihkan ke server yang lain. Atau dengan mekanisme tertentu, informasi sesi yang telah terbentuk dan disimpan oleh suatu server dapat diakses oleh server yang lain sehingga koneksi tersebut tidak perlu terputus dan identitas sebuah sesi dapat terjaga dengan baik.

III. METODOLOGI

Dalam menganalisa teori tersebut dilakukan pengujian secara bertahap. Tahap pertama proses pengujian dilakukan stress test terhadap satu server SIAM menggunakan tools JMeter. Peningkatan latency dan respon time akan dijadikan referensi kapasitas sebuah server saat menerima beban. Hasil pengujian akan dibandingkan dengan penelitian sebelumnya [1].

Tahap berikutnya adalah melakukan pengujian di level production dengan menganalisis data akses saat seluruh mahasiswa UB melakukan proses KRS. Analisis data dilakukan dengan menggunankan data log akses yang dihasilkan oleh setiap server SIAM. Data log SIAM ditunjukkan dalam Gambar 3, dimana h adalah ip server, t adalah waktu, r adalah ip pengirim, s adalah fungsi yang dipanggil

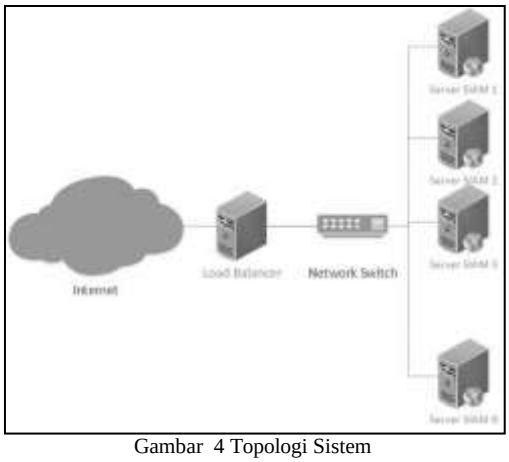

```
{"h":"175.45.184.231","t":"2014-02-01
06:48:15","r":"112.215.44.105","s":"CALL
AKADEMIK.ADDMHSKRS('145090407111020',2014,'0','0','D','MAM423
1','2011','94','04','09','01','1','145090407111020','2014','0
9','01','04','94')","e":"00000","et":"0.004479"}



{"h":"175.45.184.231","t":"2014-02-01
06:50:47","r":"112.215.44.105","s":"CALL
```


Gambar 3 Log SIAM

Data ini mencatat identitas mahasiswa yang mengakses, IP address yang digunakan, waktu akses, proses yang dilakukan oleh mahasiswa (insert, update atau delete) serta parameter yang dikirimkan oleh server SIAM. Data log ini dibandingkan dengan data log web server SIAM. Data log web server siam mencatat beberapa parameter antara lain waktu akses, url yang diakses, nama object, HTTP code, serta ukuran data yang dikirimkan oleh server. Kedua data ini dikorelasikan hingga memperoleh data akses yang dianalisis lebih lanjut.



Gambar 4 Topologi Sistem

Sistem load balancing dikonfigurasi sesuai dengan topologi yang ditunjukkan dalam Gambar 4. Proses ini menggunakan 6 Web Server SIAM dengan aplikasi yang identik. Selanjutnya dilakukan proses stress test untuk memperoleh profil sistem.

Spesifikasi perangkat keras tiap server seragam, dimana masing-masing server memiliki memory sebesar 32 GB dengan processor yang identik. Sebagai load balancing controller digunakan sistem opensource Piranha dengan konfigurasi seperti ditunjukkan dalam Gambar2.

Gambar 5 Konfigurasi sistem

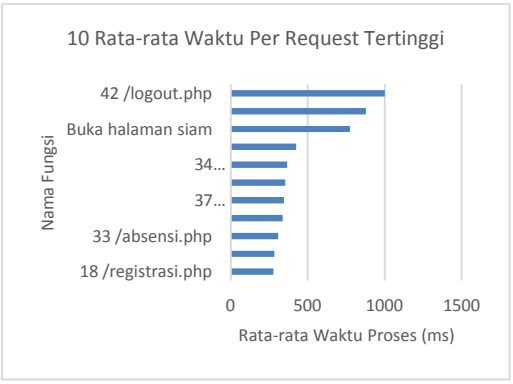
Sebagai load balancing controller digunakan sistem opensource Piranha dengan konfigurasi seperti ditunjukkan dalam Gambar 5.Sedangkan mekanisme scheduling load balancing yang digunakan adalah Soure Hash Scheduling.

IV. PENGUJIAN DAN ANALISIS

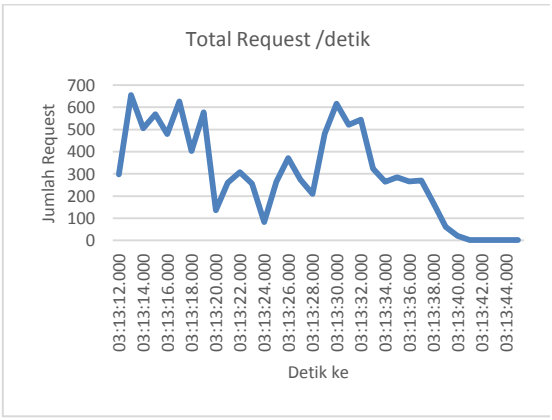
A. Simulasi Pengujian

Pengujian tahap pertama dilakukan dengan proses stress test menggunakan aplikasi JMeter. Parameter pengujian dilakukan dengan mensimulasikan proses akses ke server SIAM serta proses login ke sistem. Tahapan yang dilakukan adalah: Load halaman siam, login, akses krs, akses presensi, logout. Gambar 6 menunjukkan proses pengujian menggunakan Jmeter, sedangkan Gambar 7 menunjukkan hasil pengujian web SIAM.

Gambar 6 Pengujian menggunakan Jmeter



Gambar 7 Rata-rata request situs SIAM



Gambar 8 Stress test server SIAM untuk 300 user

Selanjutnya dilakukan pengujian dengan melui proses stress test. Mengingat mekanisme balancing yang digunakan adalah source hash, maka simulai pengujian dilakukan dengan menggunakan IP yang berbeda. Sehingga diperoleh hasil yang ditunjukkan dalam Gambar 8.

B. Pengujian di Zona Production

Setelah memperoleh data hasil pengujian secara simulasi menggunakan stress test tools, selanjutnya dilakukan pengujian di tingkat production. Pengujian, dilakukan pada saat proses KRS berlangsung. Dari data akses diperoleh bahwa puncak akses terjadi pada pukul 00:11 sebanyak 3115 source IP.

Setelah mengetahui jumlah IP Source dilakukan analisis data yang terjadi pada pukul 00:11. Data saat itu menunjukkan bahwa jumlah IP yang mengakses seluruh server adalah sebanyak 3115 dengan distribusi yang ditunjukkan dalam Gambar 5. Dari gambar ditunjukkan bahwa mekanisme source Hash Scheduling telah berhasi mendistribusikan akses berdasarkan IP Source ke 6 unit real server. Distribusi tersebut cukup merata dengan jumlah akses yang ditunjukkan dalam Gambar 6.

TABEL 1.
DATA AKSES TERTINGGI

Waktu	Jumlah Source IP
00:11	3115
00:10	3100
00:12	3093
00:14	3085
00:07	3081

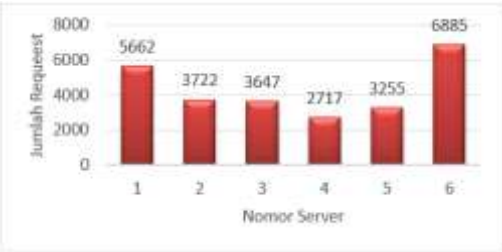
Gambar 9 Distribusi IP tiap server

TABEL 2.
PERSENTASE BEBAN PER SERVER

Server No	1	2	3
% Beban	16.62%	16.50%	16.11%
Server No	4	5	6
% Beban	15.63%	16.40%	18.74%

Jika ditihitung rata-rata tiap server diperoleh persentase beban yang ditunjukkan dalam tabel 2. Tabel tersebut menunjukkan bahwa tiap server memperoleh beban antara 15% hingga 18% dari total beban yang diterima oleh keseluruhan sistem.

Sedangkan jumlah request untuk setiap server pada saat tersebut ditunjukkan dalam Gambar 10. Untuk distribusi beban request terjadi perbedaan yang cukup signifikan antara server dengan beban permintaan terendah dan tertinggi.



Gambar 10 Distribusi permintaan tiap server

Tabel 3 menunjukkan bahwa rata-rata proses tiap request ditiap server SIAM dibawah 50ms. Namun pada kondisi beberapa kondisi mencapai 2,3 detik. Namun rata-rata waktu pemrosesan data sudah menurun sangat signifikan dibandingkan penelitian sebelumnya

Dari data keseluruhan dalam 1 hari diperoleh total data yang ditunjukkan dalam Gambar 3. Dari pengujian diperoleh informasi bahwa total request yang ditangani tiap server dalam satu hari, tertinggi adalah sebesar

1.967.429 permintaan pada server 2. Sedangkan jumlah IP tertinggi ditangani oleh Server 1 sebanyak 11.376 IP.

TABEL 3.
WAKTU PEMROSESAN SUATU PERMINTAAN

Server	Min (s)	Max (s)	Avg (s)
1	0.000216	2.310648	0.003029536
2	0.000218	0.685262	0.005230009
3	0.000216	0.871484	0.003978180
4	0.000234	0.467452	0.004448561
5	0.000234	0.699279	0.004173751
6	0.000242	0.301105	0.004666922

Total data yang ditangani oleh seluruh server adalah sebanyak 9.025.940 request dari 65.087 IP yang berbeda.

Gambar 11 Grafik Jumlah Request dan Jumlah IP Source, dihitung selama 1 hari

V. KESIMPULAN

Berdasarkan hasil pengujian baik di tingkat simulasi maupun production diperoleh bahwa mekanisme source Hash Scheduling telah dapat menjaga sesi suatu koneksi sehingga koneksi yang telah terbentuk tidak terputus meskipun mekanisme load balancing terjadi. Untuk pengujian di production diperoleh bahwa pada puncak akses terdapat 3115 IP yang sedang mengakses dengan distribusi cukup merata dengan tiap server menerima beban antara 15% hingga 18% dari keseluruhan sistem. Namun untuk beban request terjadi selisih permintaan yang signifikan antara server dengan request minimum dan server maksimum. Untuk itu itu perlu dikembangkan kembali mekanisme kombinasi antara pembagian berdasarkan IP dan berdasarkan request agar pembagian beban server lebih merata.

Secara keseluruhan, implementasi source hash scheduling pada mekanisme load balancing di KRS UB telah berhasil menangani sebanyak 9.025.940 request dari 65.087 IP berbeda dalam 1 hari. Serta memberikan total data troughput sebesar 169.537.010.395 Byte (169 GB).

REFERENCES

[1] R. Arief Setyawan, Adharul Muttaqin, Angger Abdul Razak, Lastono Risman. Analisis Mekaniseme Multi Server Load Balancing pada Server SIAKAD Universitas Brawijaya. Jurnal EECCIS Vol. 8, No. 1, halaman 93 – 98, Juni 2014.

[2] V.M.Suresh, D.Karthikeswaran, V.M.Sudha, D.Murali Chandraseker. Web Server Load Balancing Using Ssl Back-End Forwarding Method, IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012), March 2012

[3] Jin-Ha Kim, Gyu Sang Choi, and Chita R. Das. An SSL Back-End Forwarding Scheme in Cluster-Based Web Servers. IEEE Transactions On Parallel And Distributed Systems, VOL. 18, NO. 7, July 2007.

[4] Valeria Cardellini, Michele Colajanni, Philip s. Yu. Dynamic Load Balancing On Web-Server Systems. IEEE INTERNET COMPUTING, June 1999

[5] C. Kopparapu, Load Balancing Server, Firewall, and Caches, Canada: Wiley, 2001.G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.