Uji Performa Fitur Penyeimbang Beban pada Aplikasi Reverse Proxy

Ibnu Febry Kurniawan[1], Cahya Ningsih Fitri[2]

[1,2] Teknik Informatika, Universitas Negeri Surabaya
[1]ibnufebry@unesa.ac.id
[2]cahyafitri@mhs.unesa.ac.id

***Abstrak*— Teknologi penyeimbang beban (*load balancer*) bertindak sebagai regulator alokasi pemrosesan akses dari klien ke sistem layanan. Dengan adanya teknologi ini, dimungkinkan *server* dalam sistem klaster menerima beban secara bergantian. Squid lebih dikenal sebagai aplikasi *proxy*, namun aplikasi ini mempunyai fitur penyeimbang beban yang jarang digunakan. Studi ini menguji performa penyeimbang beban pada klaster *web server* dalam kondisi jaringan virtual dengan kakas uji HTTPerf. Hasil pengujian memperoleh rata-rata waktu respon, waktu *reply*, dan tingkat paket drop yang signifikan lebih baik daripada topologi jaringan tanpa Squid.**

***Kata Kunci*— penyeimbang beban, *reverse proxy*, *web server*.**

I. PENDAHULUAN

Pertumbuhan data dan informasi yang diakses melalui internet tumbuh dengan cepatnya. Menurut data yang dirilis oleh Komunitas Internet Internasional [1] pada tahun 2015 terdapat sekitar 3 triliun pengguna internet global. Lebih jauh lagi, terdapat 30 dari 100 warga Asia Pasifik yang mengakses internet. Jumlah yang masif ini belum lagi ditunjang dengan pertumbuhan ukuran laman yang ada.

Peningkatan ukuran laman *web* seringkali tidak diikuti dengan pemutakhiran kemampuan layanan *server*, baik dari segi spesifikasi maupun konfigurasi. Beberapa insiden terjadi seperti kegagalan layanan *e-commerce*, sulitnya akses ke situs registrasi pekerjaan, maupun *web* layanan pemerintahan. Insiden-insiden tersebut terjadi dikarenakan tingginya jumlah permintaan akses secara simultan namun tanpa diikuti dengan kesiapan sistem yang berjalan. Sistem layanan, baik dari spesifikasi maupun konfigurasi *server*, mungkin tidak dirancang untuk melayani jumlah pengguna dalam jumlah besar secara simultan

Salah satu upaya untuk meningkatkan kualitas layanan *web* adalah penggunaan aplikasi *server* penyeimbang beban (*load balancer*). Teknologi ini memungkinkan adanya sebuah regulator beban di dalam suatu sistem klaster jaringan. Regulator akan mengatur jumlah permintaan akses ke sistem layanan secara merata ke klaster *server* yang dimiliki. Dengan demikian, *server* akan memiliki waktu yang lebih banyak untuk melayani tiap permintaan. Tiap permintaan yang masuk dialokasikan secara terkoordinasi ke tiap anggota klaster. Beberapa aplikasi yang umum digunakan untuk keperluan ini adalah Linux Virtualization Server, HAProxy, Network Load Balancer (NLB), dan Squid.

Pada studi [2]–[5] dilakukan usaha peningkatan layanan dengan aplikasi LVS, dan NLB. Namun, belum ditemukan adanya implementasi penyeimbang beban menggunakan Squid. Squid lebih umum dikenal dengan fitur *proxy*, namun dengan aktivasi *reverse proxy*, opsi penyeimbang beban secara hierarki dapat digunakan. Eksistensi fitur ini menarik penulis untuk melihat performa *server* saat menerima jumlah akses yang tinggi.

II. STUDI LITERATUR

Studi analisis kinerja *server* telah diawali beberapa literatur terdahulu. Artikel-artikel tersebut mengamati berbagai metrik performa *server* baik dalam lingkungan virtual maupun sesungguhnya. Namun, pada umumnya literatur melakukan uji coba pada lingkungan virtual dikarenakan faktor kemudahan dan fleksibilitas skenario dan topologi. Penulis artikel [6] melakukan pengamatan kinerja *web server* pada lingkungan virtualisasi *open source* KVM dan VirtualBox dengan kakas bantuan httperf [7] dan siege. Sedikit berbeda dengan artikel ini, [8] juga melakukan pengukuran performa beberapa *web* dan FTP *server* virtual yang berjalan di atas VMWare vSphere dengan kakas bantu pylot, dan curl-loader.

Pengujian yang dilakukan pada [6], [8] mengamati kualitas layanan aplikasi dari sisi waktu jeda, respon aplikasi, dan tingkat konkurensi. Proses eksperimen yang dilakukan pada berlangsung secara sekuensial bergiliran pada tiap *server* virtual dan dibandingkan dengan performa *server* fisik. Hasil akhir dari pengujian kedua artikel ini menyatakan performa *server* virtual menurun seiring dengan meningkatnya beban pekerjaan yang diberika, namun teknologi virtual dapat digunakan untuk keperluan peningkatan sekuritas, efisientsi utilisasi sumber daya, dan minimalisasi penyimpanan serta biaya.

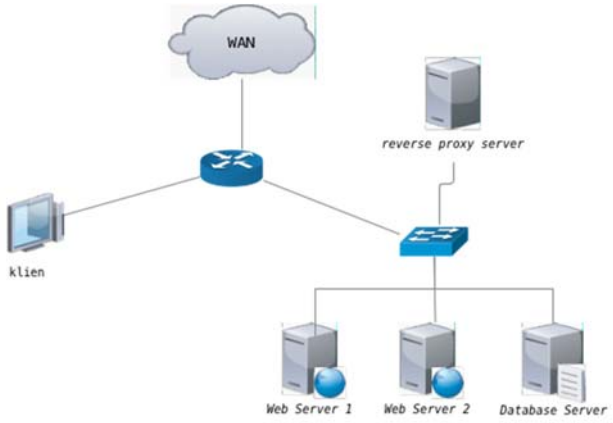
Selain penggunaan teknologi virtualisasi, kinerja layanan aplikasi sistem dioptimalisasi dengan teknologi penyeimbang beban (*load balancer*) dalam [3]–[5], [9] Menurut [4], usaha penyeimbang beban dapat ditempuh dengan 4 cara, yakni melalui (1) penggunaan fitur perangkat keras, (2) penggunaan sistem operasi, (3) teknologi *round robin* DNS, (4) algoritma penyeimbang beban berbasis Linux untuk klaster *server*. Teknologi ke-4 memiliki fitur kemampuan komputasi, skalabilitas, ketersediaan tinggi, serta lebih efektif dalam biaya. Algoritma [9] bekerja menggunakan konsep *feedback* dinamis yang disebut *Dynamic Load Sharing* (DLS) yang membagi wilayah kerja (peta) menggunakan grid, dan tiap grid akan ditangani oleh sebuah agen.

Penulis pada [3], [5] melakukan pengamatan konfigurasi penyeimbang beban dengan kakas bantu LVS dan NLB untuk melihat performa *server* virtual. Penggunaan penyeimbang beban ditujukan untuk meningkatkan ketersediaan layanan, dan kualitas layanan aplikasi.

III. METDOLOGI

A. Topologi Jaringan

Sebuah *server* disiapkan sebagai *reverse proxy* untuk melayani beberapa klien melalui protokol HTTP. *Server* ini selanjutnya akan meneruskan permintaan laman yang diminta ke *web server* yang sudah ditentukan. Pada gambar 1 terlihat bahwa 2 *web server* akan bertindak sebagai anggota klaster, dan konten *web* akan mengambil data dari *database server*. Seluruh *server* dan klien yang digunakan dalam studi ini berada pada lingkungan virtual.



Gbr 1 Topologi Jaringan dengan Squid

Kedua *web server* dapat diakses oleh klien secara internal (dalam jaringan VirtualBox). Laman *web* (www.cahya.net) yang tersedia juga dapat melalui *host* melalui entri DNS. Tiap rekues HTTP ke laman www.cahya.net akan diarahkan oleh *router* menuju *reverse proxy server*, yang selanjutnya akan diteruskan ke klaster *web server* jika terjadi TCP_MISS. Proses pemilihan *web server* tujuan berdasarkan algoritma Round Robin, sehingga kedua *server* bergantian menerima permintaan akses.

B. Spesifikasi Perangkat Lunak dan Keras Perangkat

Spesifikasi perangkat lunak dan keras baik *server* maupun klien dapat dilihat pada tabel I, II, dan III. Rendahnya jumlah RAM pada *web server* ditujukan untuk mengetahui sejauh mana kemampuan *server* menangani jumlah permintaan yang meningkat.

TABEL I
SPESIFIKASI WEB SERVER 1 DAN 2

Sistem Operasi	Debian Wheezy
CPU	Intel(R) Core(TM) i3-2370M CPU @2.4GHz
RAM	128 MB
HDD	8 GB

TABEL II
SPESIFIKASI DATABASE SERVER

Sistem Operasi	Debian Wheezy
Aplikasi Basis Data	MySQL
CPU	Intel(R) Core(TM) i3-2370M CPU @2.4GHz
RAM	800 MB
HDD	14 GB

TABEL III
SPESIFIKASI PC KLIEN

Sistem Operasi	**Windows XP**
CPU	Intel(R) Core(TM) i3-2370M CPU @2.4GHz
RAM	512 MB
HDD	14 GB

C. Skenario Pengujian

Pengujian konfigurasi fitur *reverse proxy* dalam hal penyeimbang beban dilakukan dengan cara melakukan akses melalui protokol HTTP dengan jumlah yang berbedaJumlah permintaan akses divariasi mulai 600, 800, 1000, 1200, dan1400. Keseluruhan permintaan tersebut dikirimkan serentak dalam satu waktu. Sehingga, dalam 1 detik *server* akan menerima seluruh permintaan.

Terdapat 2 jenis topologi yang akan digunakan dalam pengujian, yakni topologi dengan Squid, dan tanpa Squid. Pembedaan topologi ini ditujukan agar studi dapat melihat perbedaan metrik performa antar kedua kondisi.

Metrik performa yang diamati adalah *throughput*, *response time*, dan tingkat utilisasi CPU. Nominal permintaan akses HTTP divariasi untuk melihat sejauh mana penanganan klaster *web* pada spesifikasi *server* yang cukup minimum. Pemberian beban ini dilakukan oleh PC klien dengan menggunakan kakas bantu HTTPerf.

IV. EKSPERIMEN DAN PEMBAHASAN

Berdasarkan rancangan skenario pengujian yang dituliskan pada bab sebelumnya, studi melakukan eksperimen untuk mengamati performa klaster *web*.

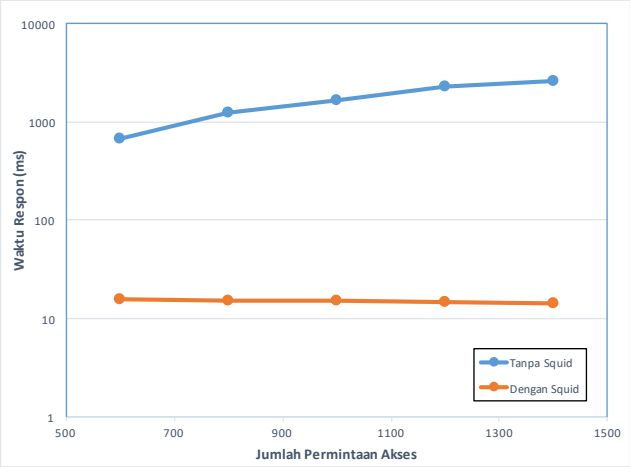
A. Hasil Pengamatan Waktu Respon

Hasil dari eksperimen pada keseluruhan permintaan akses terlihat pada tabel IV yang selanjutnya diilustrasikan pada gambar 2. Terlihat bahwa implementasi Squid pada klaster web memberikan pengurangan waktu respons *server* secara signifikan.

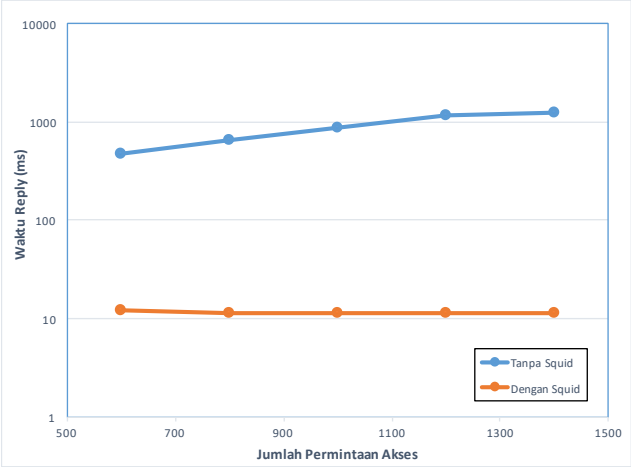
Hal ini dikarenakan Squid dapat melakukan regulasi pembagian beban secara adil dengan algoritma Round-Robin. Waktu I/O yang dibutuhkan oleh klaster *web* dapat melayani jumlah akses yang banyak secara bergantian.

TABEL IV
HUBUNGAN JUMLAH AKSES TERHADAP WAKTU RESPON

Jumlah Akses	Waktu Respons (ms)	
	Dengan Squid	Tanpa Squid
600	15,8	677,2
800	15,1	1241,7
1000	15,1	1653,1
1200	14,8	2293,1
1400	14,5	2637,6



Gbr 2 Hubungan Waktu Respon Server terhadap Jumlah Permintaan Akses



Gbr 3 Hubungan Waktu Reply terhadap Jumlah Permintaan Akses

B. Hasil Pengamatan Waktu *Reply*

Waktu *reply* (balasan) adalah waktu yang dibutuhkan dalam satu komunikasi secara penuh dari proses pengiriman hingga menerima seluruh paket laman *web* yang diminta. Tabel V memperlihatkan kemampuan Squid untuk menangani jumlah permintaan akses dengan baik. Waktu *reply* pada topologi Squid konstan dibandingkan topologi tanpa Squid.

TABEL V
HUBUNGAN JUMLAH AKSES TERHADAP WAKTU REPLY

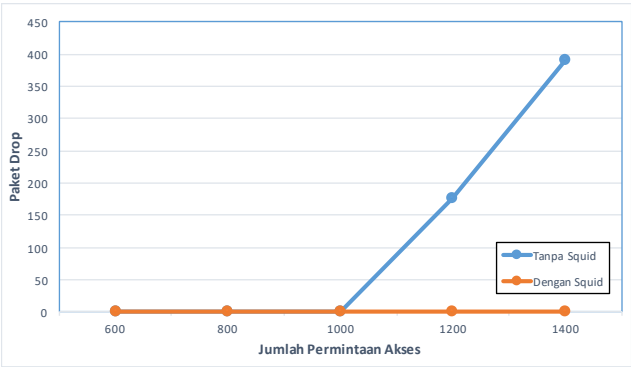
Jumlah Akses	Waktu *Reply* (ms)	
	Dengan Squid	Tanpa Squid
600	12,1	470,6
800	11,5	649,1
1000	11,6	878,1
1200	11,6	1170,5
1400	11,3	1232,2

C. Hasil Pengamatan Jumlah Paket Drop

Paket mengalami *drop* ketika antrian pengiriman terlalu penuh dan mengakibatkan suatu paket harus dibuang (*drop*). Pada tabel VI terlihat hasil eksperimen, dan diilustrasikan pada gambar 4. Gambar tersebut memperlihatkan topologi tanpa Squid gagal menangani jumlah permintaan akses bernilai 1200. Pada saat itu, kegagalan sistem terlihat dengan ada nya 176 paket harus dibuang dari antrian. Sebagai akibatnya, hanya terdapat 1124 permintaan yang mendapatkan pesan balasan dari *server*.

TABEL VI
HUBUNGAN JUMLAH AKSES TERHADAP PAKET *DROP*

Jumlah Akses	Jumlah Paket *Drop*	
	Dengan Squid	Tanpa Squid
600	0	0
800	0	0
1000	0	0
1200	0	176
1400	0	390



Gbr 4 Hubungan Paket Drop dengan Jumlah Permintaan Akses

D. Hasil Pengamatan Utilisasi CPU

Tingkat penggunaan / utilisasi CPU pada kedua topologi dapat dilihat pada tabel VI dan gambar 5. Dapat disimpulkan bahwa tingkat utilisasi rata-rata dari klaster web dengan Squid mirip dengan utilisasi tanpa Squid. Kedua topologi menunjukkan tren peningkatan utiisasi CPU seiring dengan bertambahnya jumlah permintaan akses.

TABEL VII
HUBUNGAN JUMLAH AKSES TERHADAP UTILISASI

Jumlah Akses	Utilisasi CPU (%)	
	Dengan Squid	Tanpa Squid
600	1,5	1,8
800	3,1	1
1000	2,25	2,6
1200	2,6	2,8
1400	4,3	3,8

V. SIMPULAN

Implementasi Squid sebagai *reverse proxy* dengan fitur penyeimbang beban dapat membantu klaster *web* dalam melayani jumlah permintaan akses yang tinggi. Regulator distribusi beban akan membagi *server* dalam klaster yang akan menangani tiap permintaan akses yang datang ke layanan *web*. Hasil-hasil eksperimen menunjukkan dengan tingkat utilisasi CPU yang hamper sama, penggunaan Squid dapat melayani jumlah permintaan akses yang lebih banyak dengan baik. Hal ini terlihat dari waktu balasan, respon, serta tingkat kesalahan (paket *drop*) pada toplogi jaringan Squid yang lebih baik.

REFERENSI

[1] I. Society, "Global Internet Report 2016 by Internet Society," vol. 110, p. xvi, 794 , 2017.

[2] A. Alimuddin and A. Ashari, "Peningkatan Kinerja Siakad Menggunakan Metode Load Balancing dan Fault Tolerance Di Jaringan Kampus Universitas Halu Oleo," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 10, no. 1, p. 11, 2016.

[3] A. Rukun and E. R. P. M. Kom, "Analisa Jumlah Ideal Server dengan Beban Traffic pada Load Balancing Webserver menggunakan NLB Manager," pp. 1–5, 2003.

[4] Z. Xingming and Z. Shaoxin, "One Load Balancing Solution for Mobile Video Surveillance System," in *2012 International Conference on Computer Science and Service System*, 2012, pp. 757–760.

[5] D. Lukitasari and F. Oklilas, "Analisis Perbandingan Load Balancing Web Server Tunggal Dengan Web server Cluster Menggunakan Linux Virtual Server," *Generic*, vol. 5, no. 2, pp. 31–34, 2010.

[6] S.-J. Jung, Y.-M. Bae, and W. Soh, "Web Performance Analysis of Open Source Server Virtualization Techniques," *Int. J. Multimed. Ubiquitous Eng.*, vol. 6, no. 4, 2011.

[7] "httperf(1): HTTP performance measurement tool - Linux man page." [Online]. Available: https://linux.die.net/man/1/httperf. [Accessed: 05-Dec-2017].

[8] A. Jalili, S. Homayoun, and M. Keshtgari, "Performance Analysis of Multiple Virtualized Servers," *Comput. Eng. Appl. J.*, vol. 4, no. 3, pp. 183–188, 2015.

[9] T. Nguyen, B. Duong, and S. Zhou, "A Dynamic Load Sharing Algorithm for Massively Multiplayer Online Games."