PENGUJIAN DINAMIS DARI 1050 RPS HINGGA 2000 RPS PADA SISTEM SERVER WEB BERBASIS *CLUSTER* DENGAN ALGORITMA *NEVER QUEUE*

Nongki Angsar[1] dan Maria D. Badjowawo[2]

[1]Program Studi Teknik Komputer dan Jaringan, Politeknik Negeri Kupang, Jl. Adisucipto, Penfui, Kupang
Email: angsar.nongki@gmail.com
[2] Program Studi Teknik Instalasi Listrik, Politeknik Negeri Kupang, Jl. Adisucipto, Penfui, Kupang
Email: badjowawomaria@yahoo.com

ABSTRAK

Peningkatan lalu-lintas web dan perkembangan bandwidth jaringan yang relatif lebih cepat dari perkembangan teknologi mikroprosesor dewasa ini menyebabkan platform server satu titik tidak lagi memadai untuk memenuhi kebutuhan skalabilitas sistem server web. Platform server jamak adalah jawabannya. Salah satu solusi yang telah dikenal adalah sistem server web berbasis cluster. Dalam penelitian ini akan dilakukan rancang bangun sistem server web berbasis cluster dengan algoritma Never Queue dan dilanjutkan dengan pengujian distribusi beban kerja web pada sistem ini. Pengujian dilakukan dengan cara menghasilkan beban kerja HTTP secara dinamis (dengan pesat permintaan HTTP per detik yang berubah atau naik secara teratur) dari client ke pool sistem server web. Dalam penelitian ini, hasil pengujian secara dinamis dengan pesat permintaan HTTP per detik yang berubah atau naik secara teratur dari 1050 rps hingga 2000 rps menunjukkan bahwa algoritma Never Queue mendistribusikan permintaan HTTP ke pool sistem server web dengan baik. Pesat balasan HTTP mulai dari 994,5 replies/s hingga 995,2 replies/s. Waktu tanggapan mulai dari 24,5 ms hingga 224,6 ms. Throughput mulai dari 2,6608 Mbps hingga 2,8264 Mbps. Pesat koneksi TCP mulai dari 95,1 cps hingga 187,1 cps. Galat mulai dari 0.0563153660498 hingga 8.6150409530901.

Kata kunci: pengujian distribusi, server web, klaster

Author : Nongki Angsar dan Maria D. Badjowawo

1. PENDAHULUAN

Seiring dengan semakin kompleksnya layanan dan aplikasi web dalam berbagai bidang, maka permintaan layanan web dari pengguna semakin meningkat. Contoh layanan dan aplikasi web yang populer adalah layanan dan aplikasi bisnis *(e-business)*, pendidikan *(e-learning)*, berita *(e-news)*, dan lain-lain.

Demikian pula dengan perkembangan infrastruktur jaringan dan komunikasi komputer semakin tahun semakin baik. Penerapan serat optis pada kabel [1], Gigabit Ethernet pada LAN [3], *broadband*-ISDN pada WAN [2], transmisi digital xDSL pada jalur telepon [2], dan modem kabel membuat *bandwidth* jaringan semakin besar. Bahkan sebuah prediksi yang dibuat oleh George Gilder pada tahun 1995 memperkirakan bahwa perkembangan *bandwidth* jaringan akan berlipat tiga kali setiap tahun untuk 25 tahun mendatang [4]. Prediksi ini masih berlaku, khusus untuk serat optis, merujuk pada tulisan yang dibuat pada tahun 2008 [7].

Di satu sisi, perkembangan komputer (jumlah transistor dalam keping mikroprosesor), menurut prediksi pendiri Intel, Gordon Moore pada tahun 1960-an, hanya akan berlipat dua kali setiap 18 bulan [5]. Prediksi ini sudah terbukti bertahun-tahun hingga saat ini dan lazim disebut dengan hukum Moore *(Moore's Law)*. Dengan melihat fakta perkembangan *bandwidth* jaringan yang berlipat lebih dari dua kali perkembangan komputer dan melihat kompleksnya perkembangan layanan dan aplikasi web, maka kemungkinan kemacetan di masa mendatang akan terletak pada sisi server.

Tinjauan Pustaka

Menurut Cardellini et al [6], ada dua upaya yang bisa dilakukan, yaitu upaya *scale-up* (platform server tunggal) dan upaya *scale-out* (platform server jamak). Upaya pertama sudah cukup baik, akan tetapi mempunyai beberapa

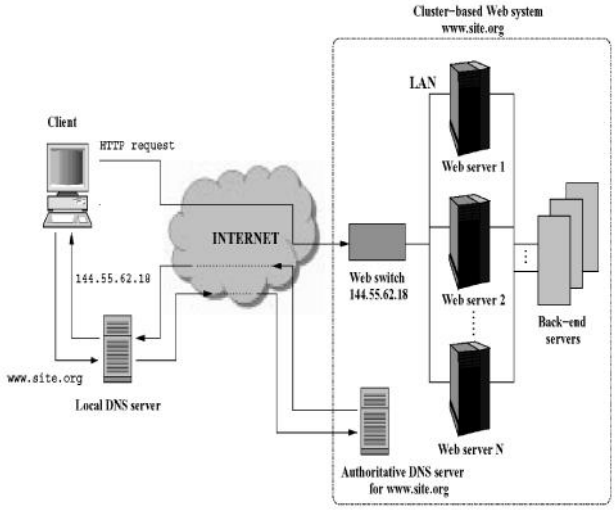
kelemahan. Pertama, membutuhkan biaya yang besar agar dapat selalu mengikuti perkembangan teknologi mutakhir. Kedua, tidak dapat menghilangkan fakta bahwa titik tunggal kegagalan (*Single Point of Failure*, SPOF) justru ada pada server itu sendiri. Ketiga, keberlangsungan dan ketersediaan layanan akan terganggu saat peningkatan skalabilitas server. Keempat, penggantian ke perangkat keras baru menyebabkan perangkat keras lama cenderung tidak terpakai lagi dalam sistem. Sedangkan upaya kedua, sebaliknya, lebih murah dan tidak memiliki SPOF.

Salah satu sistem server web jamak yang populer dan banyak dipakai adalah sistem server web berbasis *cluster.*

Dasar Teori

Sebuah sistem server web berbasis cluster adalah sekumpulan server web heterogen yang bekerja di bawah koordinasi penyeimbang beban untuk melayani permintaan HTTP dari klien. *Cluster* server web tampak dari klien sebagai satu sistem tunggal dengan satu nama dan alamat IP. Sistem ini mempunyai bagian-bagian sebagai berikut [6]:

a. Penyeimbang beban, adalah piranti digital yang sengaja ditempatkan pada lapis ke-7 atau ke-4 ISO/OSI untuk membagi beban kerja antar server web.
b. *Server Pool*, adalah *cluster* server-server yang mengerjakan layanan sesungguhnya, seperti: web, ftp, mail.
c. *Back-end Server*, adalah bagian belakang sistem yang menyimpan data dan isi layanan terkait server, seperti database dan NFS.



Gambar 1 *Arsitektur sistem server web berbasis cluster* [6]

Ada dua fungsi utama penyeimbang beban dalam sistem server web berbasis *cluster*, yaitu fungsi perutean (yang diwujudkan dalam mekanisme perutean) dan fungsi pengiriman (yang diwujudkan dalam algoritma pengiriman).

Mekanisme Perutean

Mekanisme perutean berfungsi untuk mengemas dan mengarahkan permintaan klien ke sebuah titik server web target. Mekanisme perutean yang dipakai dalam makalah ini adalah *Network Address Translation* (NAT).

Algoritma Pengiriman

Algoritma pengiriman berfungsi untuk memilih titik server web yang tepat dalam memberikan tanggapan atas permintaan klien [8]. Algoritma pengiriman yang dipakai dalam makalah ini adalah algoritma *Never Queue*.

Penentuan Bobot

Penentuan bobot dipengaruhi oleh jenis isi web (*web-content*) yang disediakan oleh server web. Apabila isi web bersifat statis (*static web-content*) maka bobot hanya akan dipengaruhi oleh faktor kecepatan media penyimpan, $\mathbf{P_m}$. Apabila isi web bersifat dinamis (*dynamic web-content*) maka bobot hanya akan dipengaruhi oleh faktor kecepatan prosesor, $\mathbf{P_p}$. Jika isi web merupakan gabungan statis dan dinamis, maka rumusnya akan menjadi

$$w = \sim P_p + (1 - \sim) P_m \quad (1)$$

di mana, $\sim$ adalah rasio yang menentukan besar kontribusi $\mathbf{P_m}$ dan $\mathbf{P_p}$ terhadap bobot w

$$\sim = \frac{N_d}{\left(N_d + N_s\right)} \tag{2}$$

dengan N_d dan N_s adalah jumlah statistik akses isi web dinamis dan statis.

2. METODOLOGI

Metodologi yang akan digunakan dalam penelitian ini mencakup alat dan bahan, jalannya penelitian, perancangan sistem dan cara analisis.

Alat dan Bahan

Spesifikasi alat yang digunakan dalam penelitian ini adalah:

1. Penyeimbang Beban: Intel® Celeron® Dual-Core N3060 1,6 GHz x 2, DDR3 SDRAM 2 GB, HD Toshiba® SATA 500 GB x 1, NIC Realtek PCI Fast Ethernet, Linux 4.8.6-300
2. Real-server 1: AMD® A4-1200 APU with Radeon® HD Graphics 1GHz x 2, DDR3 SDRAM 2 GB, HD Seagate® Barracuda® ATA 500 GB x 1, NIC Realtek PCI Fast Ethernet, Windows 8 Pro, Apache 2.2.25.
3. Real-server 2: AMD® Dual Core Processor C-50 1 GHz x 2, DDR3 SDRAM 2GB, HD Hitachi® ATA 320GB x 1, NIC Atheros Family PCI, Windows 7 Ultimate, Apache 2.2.25.
4. Klien: Intel® Celeron® M CPU 430 1,73 GHz, DDR2 SDRAM Visipro® 512 MB, HD Seagate® Barracuda® 60 GB 5400 rpm x 1, NIC Broadcom 440x 10/100 Mbps, Linux 2.6.25-14
5. Switch: SMC® 5-port 10/100Mbps Auto-MDIX Switch - SMC-EZ6505TX (*store-and-forward transmission*)
6. Kabel UTP (Cat 5) 15 meter

Bahan yang diteliti adalah rata-rata jumlah balasan HTTP per detik (pesat balasan HTTP) dari sistem server web berbasis *cluster* apabila jumlah permintaan HTTP per detik (pesat permintaan HTTP) oleh klien bersifat dinamis.

Jalannya Penelitian

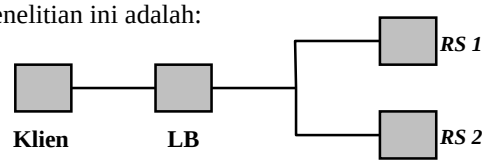
1. Mengkonfigurasi perangkat keras
   a. Memotong kabel UTP cat 5 menjadi 5 bagian dengan panjang masing-masing 3 meter per ruas
   b. Memasang konektor RJ-45 pada masing-masing kedua ujung kabel UTP cat 5 dengan *crimping tool*.
   c. Menghubungkan ujung kabel UTP cat 5 yang sudah dipasangi konektor RJ-45 pada port *layer 2 switch* dan ujung kabel satunya lagi pada port kartu LAN komputer yang bersesuaian, masing-masing pada port kartu LAN komputer penyeimbang beban, port kartu LAN komputer real-server 1, port kartu LAN komputer real-server 2 dan port kartu LAN komputer klien.
2. Mengkonfigurasi perangkat lunak.
   a. Penyeimbang Beban
      - Konfigurasi Kernel dan *Compile* Kernel
      - Konfigurasi antarmuka jaringan dan penopengan (NAT)
      - Konfigurasi perangkat lunak penyeimbang beban, mendefinisikan algoritma pengiriman, pemetaan penyeimbang beban ke real server 1 dan 2 (alamat dan *port*) serta penentuan bobot
   b. Real server
      - Konfigurasi web server Apache dan antarmuka jaringan pada Real server 1
      - Konfigurasi web server Apache dan antarmuka jaringan pada Real server 2
   c. DNS Server
      Konfigurasi DNS server Bind: file named.conf, file localhost.zone, file in-addr.arpa.zone, resolv.conf, dan host.conf.
   d. Klien
      - Konfigurasi antarmuka jaringan
      - Pemasangan perangkat lunak pengujian beban kerja web
3. Melakukan pengujian distribusi beban kerja web secara dinamis dengan pesat permintaan HTTP per detik yang berubah atau naik secara teratur dari 1050 rps hingga 2000 rps pada sistem server web berbasis *cluster* dengan algoritma *Never Queue*.
4. Pada akhir pengujian dilakukan pengambilan data-data berupa:
   - Pesat permintaan HTTP (rps),
   - Pesat balasan HTTP (replies/s),
   - Waktu tanggapan (ms),
   - Throughput (Mbps),
   - Pesat koneksi TCP (cps),

- Galat (galat).

5. Membuat tabel berdasarkan data-data yang disebutkan pada langkah nomor 4.
6. Menyusun laporan penelitian berdasarkan hasil penelitian dan didukung literatur yang sesuai.
7. Membuat kesimpulan

Perancangan Sistem

Sistem yang dirancang dalam penelitian ini adalah:



Gambar 2 *Jaringan sistem server web berbasis cluster*

Cara Analisis

Sistem server web yang dibuat dalam penelitian ini kemudian divalidasi dan dievaluasi menurut enam parameter pengujian, yaitu: jumlah pesat permintaan HTTP, pesat balasan HTTP, waktu tanggapan, *throughput*. , pesat koneksi TCP dan galat. Keenam parameter pengujian tersebut diuji untuk algoritma yang dipakai, yaitu *Never Queue*. Cara pengujian dilakukan dengan menghasilkan pesat permintaan HTTP (rps) dari klien secara dinamis dengan pesat permintaan HTTP per detik yang berubah atau naik secara teratur dari 1050 rps hingga 2000 rps, lalu mencatat berapa jumlah pesat balasan HTTP (replies/s), waktu tanggapan (ms), *troughput* (Mbps), pesat koneksi TCP (cps) dan galat dari penyeimbang beban yang mengatur permintaan HTTP ke kedua *real-server*. Data-data tersebut lalu ditampilkan dalam tabel. Perbandingan keenam parameter dilakukan dengan melihat tabel data-data yang dihasilkan untuk algoritma *Never Queue*. Jadi ada satu tabel yang berisi jumlah pesat permintaan HTTP (rps), jumlah pesat balasan HTTP (replies/s), waktu tanggapan (ms), *troughput* (Mbps), pesat koneksi TCP (cps) dan galat. Tabel inilah yang dipakai untuk melihat karakteristik sistem yang dibuat dengan algoritma *Never Queue* dalam mendistribusikan beban kerja web ke sistem server web berbasis *cluster*.

3. HASIL DAN PEMBAHASAN

Setelah konfigurasi perangkat keras dan konfigurasi perangkat lunak pada sistem server web berbasis *cluster* selesai, maka tahap selanjutnya adalah pengujian distribusi beban kerja web untuk menunjukkan karakteristik algoritma *Never Queue* dalam mendistribusikan permintaan HTTP ke kedua *real-server*. Untuk mengujinya, dibuat permintaan HTTP dari sisi klien untuk memproduksi beban secara dinamis dengan pesat permintaan HTTP per detik yang berubah atau naik secara teratur.

Hasil Pengujian Beban Dinamis dari 1050 rps hingga 2000 rps

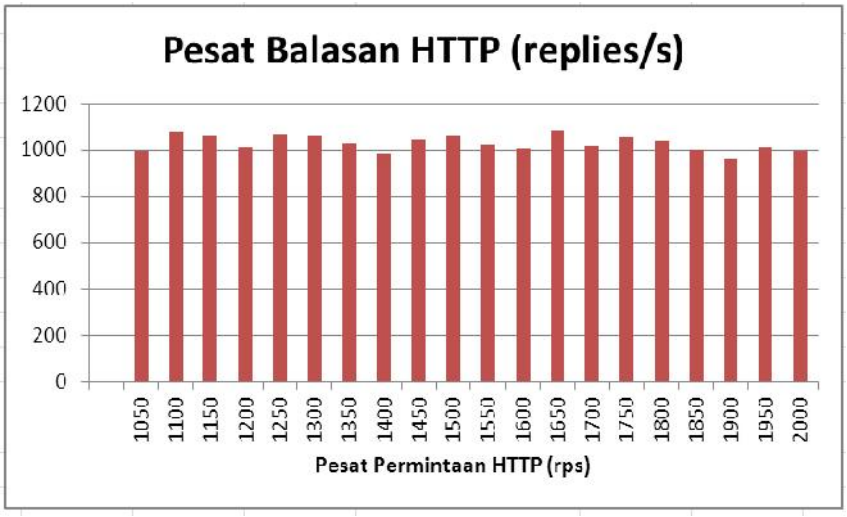
Pada pengujian ini, pesat permintaan HTTP yang dihasilkan sebesar 1050 hingga 2000 permintaan HTTP per detik, dan didistribusikan ke kedua server web dalam *cluster* (*pool*) dengan algoritma *Never Queue*. Angka-angka permintaan HTTP per detik ini diperoleh dengan metode pembebanan permintaan HTTP yang berubah atau naik secara teratur dari 1050 rps hingga 2000 rps dengan langkah 50 rps dan akan berbeda untuk konfigurasi *hardware* yang berbeda. Dasar penggunaan angka-angka ini karena pada angka-angka permintaan HTTP ini, pesat balasan HTTP dari server sudah stabil. Tujuan pengujian distribusi beban kerja web ini untuk mengukur jumlah pesat balasan HTTP, waktu tanggapan, dan *troughput* menurut algoritma *Never Queue*, di saat pesat permintaan HTTP berubah atau naik secara teratur. Hasilnya tampak dalam Tabel 1 berikut ini.

Tabel 1 Hasil pengujian dinamis Algoritma Never Queue

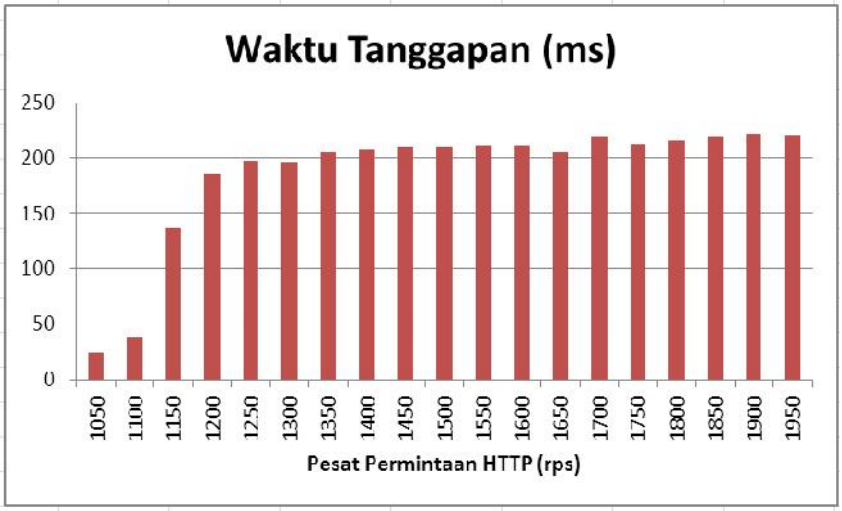
Pesat Permintaan HTTP (rps)	Pesat Balasan HTTP (replies/s)	Waktu Tanggapan (ms)	Through put (Mbps)	Pesat Koneksi TCP (cps)	Galat (galat)
1050	994,5	24,5	2,66	95,1	0,056
1100	1082,2	37,5	3,05	109,5	0,103

1150	1066,3	137,1	2,78	103,2	0,418
1200	1012,9	186,4	2,76	107,6	0,969
1250	1071,2	198,0	2,83	111,1	1,059
1300	1066,6	195,4	3,00	123,5	1,560
1350	1028,5	205,2	2,76	119,1	2,150
1400	987,3	207,3	2,77	124,8	2,661
1450	1044,0	210,2	2,76	128,8	3,099
1500	1058,5	210,0	2,78	132,1	3,386
1550	1023,5	211,9	2,74	135,8	3,958
1600	1006,6	211,8	2,73	138,0	4,192
1650	1083,3	204,9	3,04	154,3	4,269
1700	1022,6	219,9	2,70	148,8	5,513
1750	1054,0	212,8	2,97	163,7	5,484
1800	1041,4	215,7	2,68	152,6	6,005
1850	999,6	220,0	2,70	160,3	6,672
1900	965,0	222,3	2,61	160,2	7,271
1950	1016,7	220,9	2,68	167,6	7,580
2000	995,2	224,6	2,83	187,1	8,615

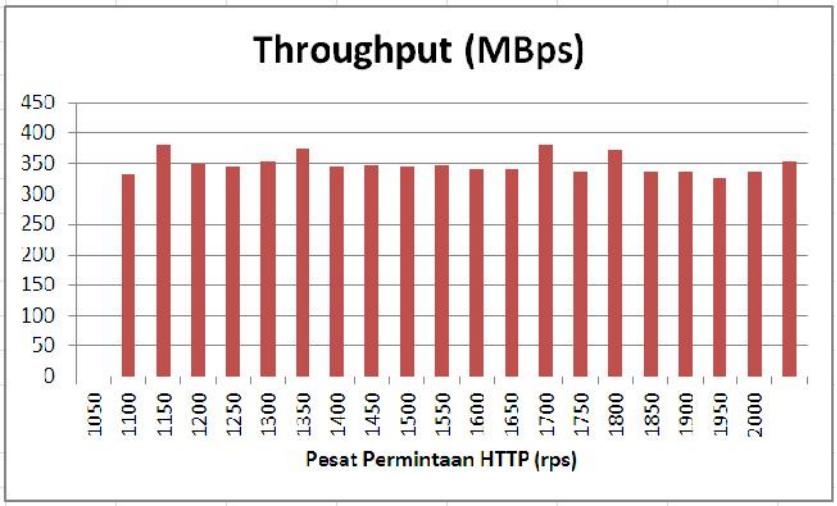
Dari hasil pengujian dinamis dari 1050 rps hingga 2000 rps diperoleh data-data pada Tabel I yang ditampilkan dalam grafik balasan HTTP (pada Gambar 3), waktu tanggapan (pada Gambar 4), throughput (pada Gambar 5), pesat koneksi TCP (pada Gambar 6), dan galat (pada Gambar 7) berikut ini.



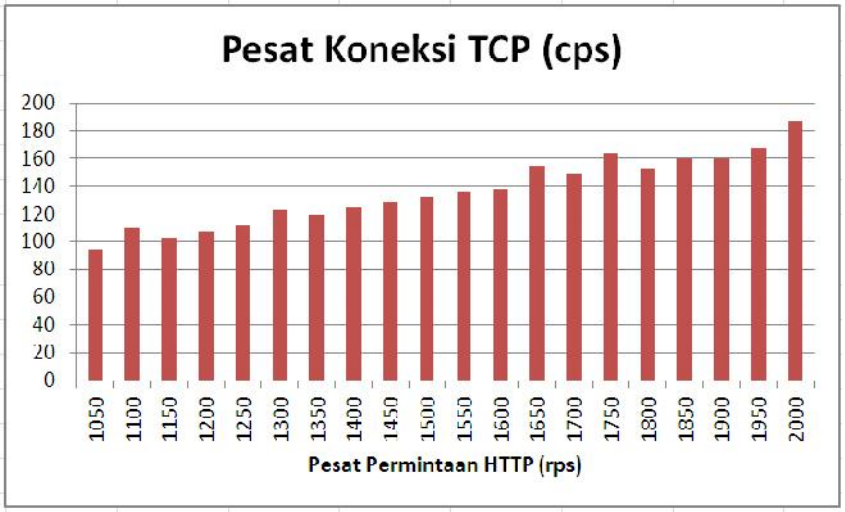
Gambar 3 Grafik Balasan HTTP Cluster dengan algoritma Never Queue



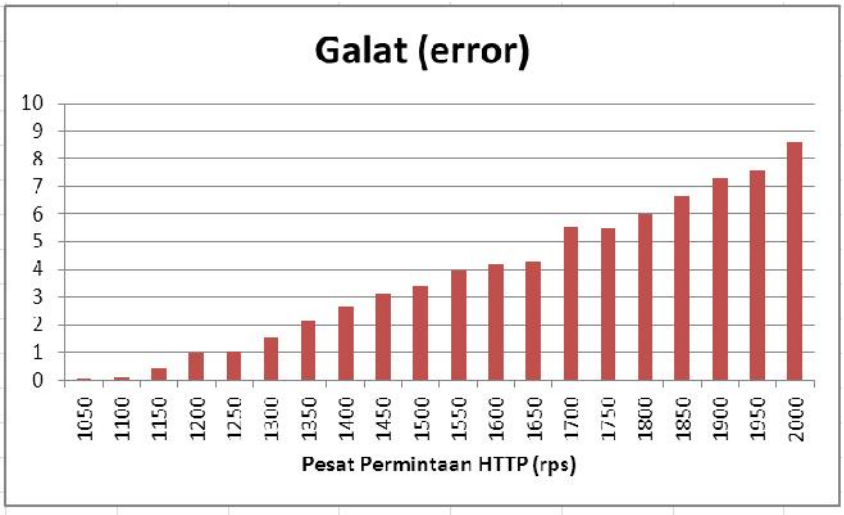
Gambar 4 Grafik Waktu Tanggapan Cluster dengan algoritma Never Queue



Gambar 5 Grafik Throughput Cluster dengan algoritma Never Queue



Gambar 6 Grafik Pesat Koneksi TCP Cluster dengan algoritma Never Queue



Gambar 7 Grafik Galat Cluster dengan algoritma Never Queue

Dalam pengujian beban dinamis dengan pesat permintaan HTTP 1050 rps hingga 2000 rps dengan langkah 50 rps, jika dilihat dalam Tabel 1 serta Gambar 3, Gambar 4, Gambar 5, Gambar 6, dan Gambar 7, maka parameter pesat balasan HTTP, waktu tanggapan, dan throughput cenderung stabil terhadap kenaikan pesat permintaan HTTP. Sedangkan untuk parameter pesat koneksi TCP, dan galat terjadi kenaikan linier juga seiring dengan bertambahnya pesat permintaan HTTP yang dihasilkan. Kestabilan parameter balasan HTTP, waktu tanggapan, dan *throughput*, terhadap kenaikan pesat permintaan HTTP disebabkan karena tingkat kejenuhan pelayanan server telah mencapai batasnya. Sedangkan pesat koneksi TCP dan galat naik secara linier terhadap kenaikan pesat permintaan HTTP ini disebabkan oleh kenaikan jumlah paket yang bertumbukan atau drop dalam perjalanan seiring dengan bertambahnya pesat permintaan HTTP dari klien ke server web.

4. KESIMPULAN

Kesimpulan yang bisa diambil dari penelitian ini adalah :
Untuk pengujian beban dinamis dengan pesat permintaan HTTP 1050 rps hingga 2000 rps dengan langkah 50 rps, terjadi kejenuhan pada parameter balasan HTTP, waktu tanggapan, dan *throughput*. Sedangkan pesat koneksi TCP, dan galat, terjadi kenaikan secara linier seiring dengan bertambahnya permintaan HTTP. Korelasi kestabilan layanan dengan kenaikan galat adalah saat mencapai titik jenuh, semakin besar permintaan HTTP, semakin besar galat yang terjadi.

DAFTAR PUSTAKA

Roger L. Freeman. (1998). "*Telecommunication Transmission Handbook, 4th edition"*. Canada: John Wiley & Sons, Inc.

H. Kaplan, B. Noseworthy. (2000). "*The Ethernet Evolution: 10 to 10,000 Mbps"*. Atlanta: Networld Interop.

William Stallings. (2000). "*Data and Computer Communication, 6th edition"*. Upper Saddle River, New Jersey: Prentice-Hall.

J. Gray, P. Shenoy. (2000). "*Rules of Thumb in Data Engineering"*. In IEEE 16th International Conference on Data Engineering. San Diego, California: IEEE.

G. Gilder. (2008). "*The Coming Creativity Boom"*. [Online], *http://www.forbes.com/forbes/2008/1110/ 036.html.*

Intel Corporation. (2003). ("IA-32 Intel® Architecture Software Developer's Manual) *Vol. 1: Basic Architecture, Order Number 24547-012"*. Illionis.

Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, Philip S. Yu. (2001). "*The State of the Art in Locally Distributed Web-server Systems"*. IBM Research Report.

N. G. Shivaratri, P. Krueger, M. Singhal. (1992). "*Load Distributing for Locally Distributed Systems"*. IEEE Computer.