Pengujian Distribusi Beban Web dengan Algoritma *Least Connection* dan *Weighted Least Connection*

Nongki Angsar[1]

***Abstract*— The increased of web traffic and the development of the network bandwidth is relatively faster than the development of microprocessor technology led to today's server platforms one point no longer adequate to meet the needs of system scalability web server. Multiple server platforms is the answer. One solution that has been known is a cluster-based web server system. This research would be testing a web workload distribution on a cluster-based web server system by means of generating HTTP workloads statically (with fast HTTP requests per second is fixed) and dynamic (with HTTP requests per second rapidly changing or ride on a regular basis) from the client to a web server system pool. Followed by analysis of traffic data packets. And then compare the results of the testing of the web workload distribution among the Least Connection algorithm (without weights) and Weighted Least Connection algorithm (with weights).**

***Intisari*— 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 pengujian distribusi beban kerja web pada sistem server web berbasis *cluster* dengan cara menghasilkan beban kerja HTTP secara statis (dengan pesat permintaan HTTP per detik yang tetap) dan secara dinamis (dengan pesat permintaan HTTP per detik yang berubah atau naik secara teratur) dari client ke *pool* sistem server web. Dilanjutkan dengan analisa lalu-lintas paket data. Dan kemudian membandingkan hasil pengujian distribusi beban kerja web tersebut antara algoritma *Least Connection* (tanpa bobot) dan algoritma *Weighted Least Connection* (dengan bobot).**

***Keywords*— Pengujian Distribusi, Server Web, Klaster.**

I. 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.

II. SISTEM SERVER WEB BERBASIS CLUSTER

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.

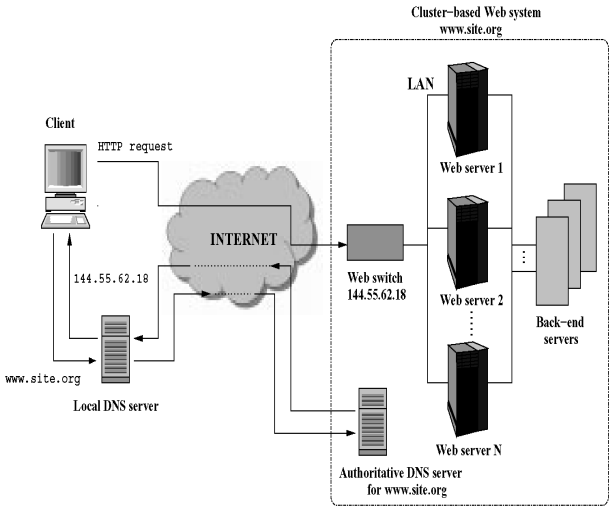
Salah satu sistem server web jamak yang populer dan banyak dipakai adalah sistem server web berbasis *cluster*. Sebuah sistem server web berbasis cluster adalah sekumpulan server web heterogen yang bekerja di bawah koordinasi penyeimbang beban untuk melayani permintaan HTTP dari klien. *Cluster* server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

1) *Penyeimbang beban* : adalah piranti digital yang sengaja ditempatkan pada lapis ke-7 atau ke-4 ISO/OSI untuk membagi beban kerja antar server web.

[1]*Politeknik Negeri Kupang, Jl. Adi Sucipto Ten Fui, Kampus Undana Baru, Kupang, NTT., angsar.nongki@gmail.com*

2) *Server Pool* : adalah *cluster* server-server yang mengerjakan layanan sesungguhnya, seperti: web, ftp, mail.

3) *Back-end Server* : adalah bagian belakang sistem yang menyimpan data dan isi layanan terkait server, seperti database dan NFS.



Gbr. 1 Arsitektur sistem server web berbasis cluster [6]

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

A. 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).

B. 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 *Least Connection* (tanpa bobot), lalu dibandingkan dengan algoritma *Weighted Least Connection* (dengan bobot).

C. 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

$$w = \alpha P_p + (1-\alpha) P_m \qquad (1)$$

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

$$\alpha = \frac{N_d}{(N_d + N_s)} \qquad (2)$$

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

III. METODOLOGI

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

A. Alat dan Bahan

Spesifikasi alat yang digunakan dalam penelitian ini adalah:

1) *Penyeimbang Beban*: Intel® Core 2 Duo CPU E4500 2,20 GHz x 2, DDR2 SDRAM Visipro® 2 GB, HD Seagate® Barracuda® SATA 160 GB 7200 rpm x 1, NIC PCI Express 10/100 Mbps (on board), NIC Realtek RTL8139 Family PCI Fast Ethernet, Linux 3.6.10-4

2) *Real-server 1*: Intel® Core 2 Duo CPU E4500 2,20 GHz x 2, DDR2 SDRAM Visipro® 2 GB, HD Seagate® Barracuda® SATA 160 GB 7200 rpm x 1, NIC PCI Express 10/100 Mbps (on board), Windows 7, Apache 2.2.25.

3) *Real-server 2*: Intel® Pentium 4 CPU 2,40 GHz x 1, DDR2 SDRAM Visipro® 256 MB, HD Seagate® Barracuda® SATA 40 GB 7200 rpm x 1, NIC Realtek RTL8139 Family PCI Fast Ethernet, Windows XP Professional SP2, 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 statis dan dinamis.

B. Jalannya Penelitian

1) Mengkonfigurasi perangkat keras

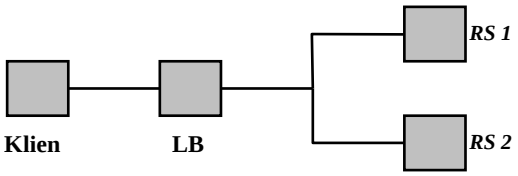
2) Mengkonfigurasi perangkat lunak.

3) Melakukan pengujian distribusi beban kerja web statis pada sistem server web berbasis *cluster*. Pada akhir pengujian dilakukan pengambilan data.

4) Melakukan pengujian distribusi beban kerja web dinamis pada sistem server web berbasis *cluster*. Pada akhir pengujian dilakukan pengambilan data.

C. Perancangan Sistem

Sistem yang dirancang dalam penelitian ini adalah:



Gbr. 2 Jaringan sistem server web berbasis cluster

D. Cara Analisis

Sistem server web yang dibuat dalam penelitian ini kemudian divalidasi dan dievaluasi menurut tiga parameter pengujian, yaitu: jumlah pesat balasan HTTP, waktu tanggapan, dan *troughput*. Ketiga parameter pengujian tersebut diuji dan dibandingkan untuk masing-masing algoritma yang dipakai, baik algoritma *Least Connection* maupun algoritma *Weighted Least Connection*.

Cara pengujian dilakukan dengan menghasilkan pesat permintaan HTTP dari klien (baik statis maupun dinamis), lalu mencatat berapa jumlah pesat balasan HTTP, waktu tanggapan, dan *troughput* dari penyeimbang beban yang mengatur permintaan HTTP ke kedua *real-server*. Data-data tersebut lalu ditampilkan dalam grafik.

Perbandingan ketiga parameter dilakukan dengan melihat grafik data-data yang dihasilkan untuk masing-masing algoritma. Jadi akan ada dua grafik yang masing-masing berisi jumlah pesat permintaan HTTP, jumlah pesat balasan HTTP, waktu tanggapan, dan *troughput*. Kedua grafik inilah yang akan dibandingkan untuk melihat mana algoritma yang lebih baik dalam mendistribusikan beban kerja web ke sistem server web berbasis *cluster*.

IV. 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 membuktikan mana algoritma yang lebih baik dalam mendistribusikan permintaan HTTP ke kedua *real-server*. Untuk mengujinya, dibuat permintaan HTTP dari sisi klien untuk memproduksi beban, baik secara statis dengan pesat koneksi TCP tunggal maupun secara dinamis dengan pesat koneksi TCP jamak.

A. Hasil Pengujian Beban Statis

Pada pengujian ini, pesat permintaan HTTP yang dihasilkan sebesar 8.000 permintaan HTTP per detik, dan didistribusikan ke kedua server web dalam *cluster* (*pool*) dengan algoritma *Least Connection* (LC) dan algoritma *Weighted Least Connection* (WLC). Angka 8.000 permintaan HTTP per detik ini diperoleh dengan metode *Trial and Error* dan akan berbeda untuk konfigurasi *hardware* yang berbeda. Dasar penggunaan angka 8.000 ini karena pada angka 8.000 permintaan HTTP ini, pesat balasan HTTP dari server mulai beranjak stabil.

Tujuan pengujian distribusi beban kerja web ini untuk mengukur jumlah pesat balasan HTTP, waktu tanggapan, dan *troughput* menurut cara masing-masing algoritma, di saat pesat permintaan HTTP tetap.

Untuk pengujian beban statis ini, hasilnya adalah:

TABEL I
HASIL PENGUJIAN BEBAN STATIS

Parameter	WLC	LC
Pesat permintaan HTTP (requests/s)	7006,7	6954,2
Pesat balasan HTTP (replies/s)	7177,8	6986,9
Waktu tanggapan (ms)	39,5	37,3
Throughput (KBps)	2478,8	2460,1

Jika yang dibandingkan adalah parameter pesat permintaan HTTP, pesat balasan HTTP, dan *throughput* maka algoritma WLC lebih baik dibandingkan algoritma LC. Hanya, untuk parameter waktu tanggapan, algoritma LC lebih baik (cepat) daripada algoritma WLC.

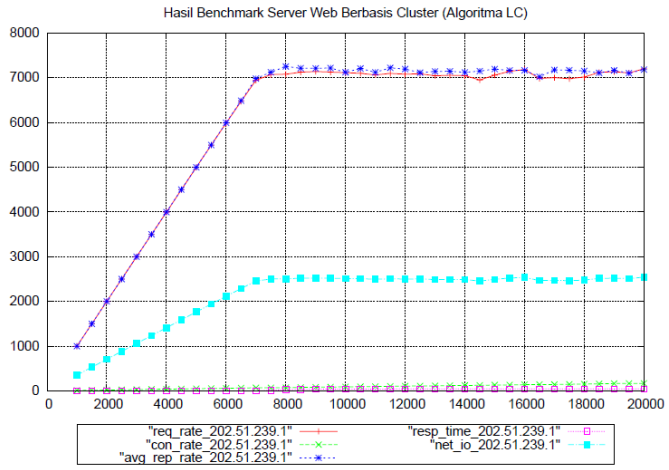
B. Hasil Pengujian Beban Dinamis

Pada pengujian ini, pesat permintaan HTTP yang dihasilkan sebesar 1.000 permintaan HTTP per detik secara bertahap dan kontinyu dinaikkan 500 permintaan HTTP per detik hingga mencapai 20.000 permintaan HTTP per detik, dan didistribusikan ke kedua server web dalam *cluster* (*pool*) dengan algoritma *Least Connection* dan algoritma *Weighted Least Connection*. Angka 1.000, 500, dan 20.000 permintaan HTTP per detik diperoleh secara empiris dari percobaan yang dilakukan berulang kali. Dan rentang 1.000 hingga 20.000 permintaan HTTP per detik adalah rentang yang terbaik untuk ditampilkan dalam grafik.

Apabila diambil rentang 1.000 hingga 7.000 permintaan HTTP per detik maka grafik yang ditampilkan akan berupa garis linier. Sedangkan apabila diambil rentang 7.000 hingga 20.000 permintaan HTTP per detik maka grafik yang ditampilkan akan berupa garis mendatar. Dengan kata lain, mengambil rentang 1.000-7.000 saja atau 7.000-20.000 saja, tidak akan menampilkan keseluruhan grafik, baik transien maupun *steady-state*-nya. Sedangkan angka 500 permintaan HTTP per detik merupakan langkah (*step*) yang moderat, tidak terlalu kasar dan tidak terlalu halus untuk tampilan grafik, sekaligus tidak terlalu cepat dan tidak terlalu lama untuk durasi waktu pengujian.

Tujuan pengujian distribusi beban kerja web ini untuk mengukur jumlah pesat balasan HTTP, waktu tanggapan, dan *throughput* menurut cara masing-masing algoritma, di saat pesat permintaan HTTP berubah (naik secara teratur).

Berikut ini Gbr.3 adalah grafik hasil pengujian beban dinamis untuk algoritma *Least Connection* (LC).



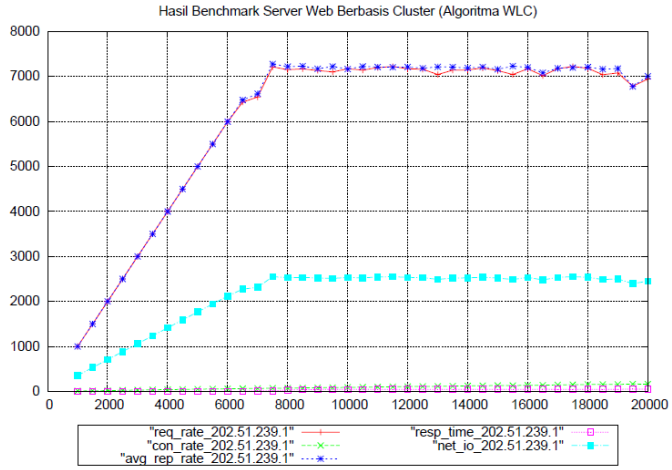
Gbr. 3 Grafik pesat permintaan HTTP (permintaan/detik), pesat balasan HTTP (balasan/detik), waktu tanggapan (milidetik), dan throughput (KBps) pada server web dengan algoritma Least Connection (LC)

Pada saat diberi pesat koneksi TCP sebesar 10 hingga 70 koneksi TCP/detik (1 koneksi TCP berisi 100 permintaan HTTP), seluruh pesat permintaan HTTP yang dihasilkan oleh klien dapat diterima dengan baik oleh server, dan sebaliknya rata-rata pesat balasan HTTP yang dihasilkan server dapat diterima dengan baik oleh klien. Tampak dalam grafik berupa dua garis yang berimpit, dari 999,9 hingga 6940,8 permintaan HTTP/detik dan 999,9 hingga 6980,4 rata-rata balasan HTTP/detik. Dengan lain kata, kinerja dan performa server web masih prima dengan galat nyaris nol (nyaris tanpa kesalahan).

Setelah melewati pesat koneksi di atas 70 koneksi TCP/detik maka pesat permintaan HTTP dan rata-rata pesat balasan HTTP pada server mulai mencapai titik jenuhnya, yaitu di sekitar aras 7100 permintaan atau balasan HTTP/detik. Semakin pesat koneksi TCP ditingkatkan maka pesat permintaan HTTP dan pesat balasan HTTP hanya akan berkisar tidak jauh dari garis tersebut (relatif stabil).

Pada kondisi ini, server menjadi semakin sibuk dan waktu respons menjadi semakin lama, sehingga koneksi yang ditolak bertambah dan permintaan HTTP yang melewati batas waktu semakin banyak. Waktu tanggapan pada pesat koneksi 70 koneksi TCP/detik masih kecil yaitu 3,1 milidetik, namun pada pesat koneksi di atas 70 koneksi TCP/detik, waktu tanggapan menjadi semakin besar yaitu 14,6 milidetik dan diakhir iterasi ke-39 menjadi 44,1 milidetik. Sedangkan *througput* awalnya pada pesat koneksi 10 koneksi TCP/detik adalah 353,5 KBps, mulai meningkat seiring bertambahnya pesat koneksi TCP hingga mencapai titik jenuhnya di sekitar nilai 2500 KBps.

Selanjutnya, grafik hasil pengujian beban dinamis untuk algoritma *Weighted Least Connection* (WLC) tampak pada Gbr. 4. Pada saat diberi pesat koneksi TCP sebesar 10 hingga 75 koneksi TCP/detik (bandingkan dengan data algoritma *Least Connection* sebelumnya 70 koneksi TCP/detik), seluruh pesat permintaan HTTP yang dihasilkan oleh klien dapat diterima dengan baik oleh server, dan sebaliknya rata-rata pesat balasan HTTP yang dihasilkan server dapat diterima dengan baik oleh klien.



Gbr. 4 Grafik pesat permintaan HTTP (permintaan/detik), pesat balasan HTTP (balasan/detik), waktu tanggapan (milidetik), dan troughput (KBps) pada server web dengan algoritma Weighted Least Connection (WLC)

Tampak dalam grafik berupa dua garis yang berimpit, dari 1000 hingga 7211,3 permintaan HTTP/detik (bandingkan dengan data algoritma *Least Connection* sebelumnya, yaitu dari 999,9 hingga 6940,8 permintaan HTTP/detik) dan dari 1000 hingga 7274,4 balasan HTTP/detik (bandingkan dengan data algoritma *Least Connection* sebelumnya, yaitu dari 999,9 hingga 6980,4 balasan HTTP/detik). Dengan lain kata, kinerja dan performa server web masih prima dengan galat nyaris nol (nyaris tanpa kesalahan).

Setelah melewati pesat koneksi di atas 75 koneksi TCP/detik maka pesat permintaan HTTP dan rata-rata pesat balasan HTTP pada server mulai mencapai titik jenuhnya, yaitu di sekitar aras 7175 permintaan atau balasan HTTP/detik (bandingkan dengan data algoritma *Least Connection* sebelumnya, yaitu 7100 permintaan atau balasan HTTP/detik).

Semakin pesat koneksi TCP ditingkatkan maka pesat permintaan HTTP dan pesat balasan HTTP hanya akan berkisar tidak jauh dari garis tersebut (relatif stabil). Pada kondisi ini, server menjadi semakin sibuk dan waktu respons menjadi semakin lama, sehingga koneksi yang ditolak bertambah dan permintaan HTTP yang melewati batas waktu semakin banyak.

Waktu tanggapan pada pesat koneksi 65 koneksi TCP/detik masih kecil yaitu 2,6 milidetik, namun pada pesat koneksi di atas 70 koneksi TCP/detik, waktu tanggapan menjadi semakin besar yaitu 15,7 milidetik (bandingkan dengan data algoritma *Least Connection* yaitu 14,6 milidetik) dan diakhir iterasi ke-39 menjadi 49,2 milidetik (bandingkan dengan data algoritma *Least Connection* yaitu 44,1 milidetik). Sedangkan *througput* awalnya pada pesat koneksi 10 koneksi TCP/detik adalah 353,5 KBps, mulai meningkat seiring bertambahnya pesat koneksi TCP hingga mencapai titik

ISSN 2301 - 4156

jenuhnya di sekitar nilai 2500 KBps (bandingkan dengan data algoritma *Least Connection* sebelumnya yaitu 2500 KBps).

V. KESIMPULAN

Beberapa kesimpulan yang bisa diambil dari penelitian ini adalah.

1) Untuk hasil pengujian beban statis, jika yang dibandingkan adalah parameter pesat permintaan HTTP, pesat balasan HTTP, dan *throughput* maka algoritma WLC lebih baik dibandingkan algoritma LC. Hanya, untuk parameter waktu tanggapan, algoritma LC lebih baik (cepat) daripada algoritma WLC.
2) Untuk hasil pengujian beban dinamis, algoritma WLC lebih baik dalam mendistribusikan permintaan HTTP dan balasan HTTP, namun waktu tanggapannya lebih lambat jika dibandingkan dengan algoritma LC. Sedangkan *throughput* algoritma WLC dan algoritma LC sama besar.

REFERENSI

[1] Roger L. Freeman. *Telecommunication Transmission Handbook, 4th edition*. Canada: John Wiley & Sons, Inc., 1998.
[2] William Stallings. *Data and Computer Communication, 6th edition*. Upper Saddle River, New Jersey: Prentice-Hall, 2000.
[3] H. Kaplan, B. Noseworthy. *The Ethernet Evolution: 10 to 10,000 Mbps*. Atlanta: Networld Interop, 2000.
[4] J. Gray, P. Shenoy. *Rules of Thumb in Data Engineering*. In IEEE 16th International Conference on Data Engineering. San Diego, California: IEEE, 2000.
[5] _____. *IA-32 Intel® Architecture Software Developer's Manual Vol. 1: Basic Architecture, Order Number 24547-012*. Illionis: Intel Corporation, 2003.
[6] Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, Philip S. Yu. *The State of the Art in Locally Distributed Web-server Systems*. IBM Research Report, 2001.
[7] G. Gilder. *The Coming Creativity Boom*. October 23rd, 2008. *http://www.forbes.com/forbes/2008/1110/036.html*
[8] N. G. Shivaratri, P. Krueger, M. Singhal. *Load Distributing for Locally Distributed Systems*. IEEE Computer, 1992.