# PENGUJIAN DISTRIBUSI BEBAN KERJA WEB SECARA STATIS PADA SISTEM SERVER WEB BERBASIS *CLUSTER* DENGAN ALGORITMA *NEVER QUEUE*

**Nongki Angsar[1], Maria D Badjowawo[2]**

[1,2]Jurusan Teknik Elektro Politeknik Negeri Kupang
Email: [1] angsar.nongki@gmail.com, [2] badjowawomaria@yahoo.com

***ABSTRACT***

*The increase in web traffic and the development of network bandwidth that is relatively faster than the development of microprocessor technology today causes the one point server platform to be no longer sufficient to meet the scalability requirements of web server systems. Multiple server platforms are the answer. One known solution is cluster-based web server systems. In this study, a cluster-based web server system would be designed with the Never Queue algorithm and continued with testing the distribution of web workload on this system. The tests were carried out by generating HTTP workloads statically (with fast HTTP requests per fixed second) and dynamically (rapid HTTP requests per second that change or rise regularly) from the client to the web server system pool. Followed by analyzing data package traffic. In this study, the results of static testing with rapid HTTP requests per second which still showed that the Never Queue algorithm distributed HTTP requests to the web server system pool properly and got HTTP replies that tend to be stable at the HTTP average of 1031.8 replies/s. As for the rapid parameters of TCP connections, response times and errors increased with the rapid increasing HTTP requests generated. The average output was at 2,983 Mbps.*

***Keywords****: Testing distribution, web server, cluster*

## 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].

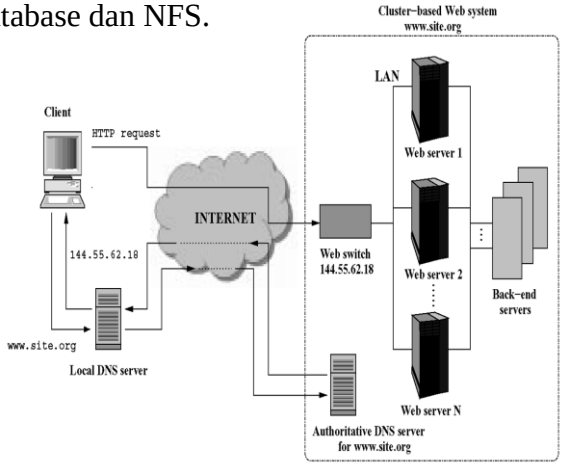
Di satu sisi, perkembangan komputer (jumlah transistor dalam keping mikroprosesor), menurut prediksi pendiri Intel, Gordon Moore pada tahun 1960-an, hanya akan berlipat dua kali setiap 18 bulan [5]. Prediksi ini sudah terbukti bertahun-tahun hingga saat ini dan lazim

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.

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]:

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**

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).

### *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 *Never Queue*.

### *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

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

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

$$\alpha = \frac{N_d}{\left(N_d + N_s\right)}$$

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

## BAHAN DAN METODE PENELITIAN

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® 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

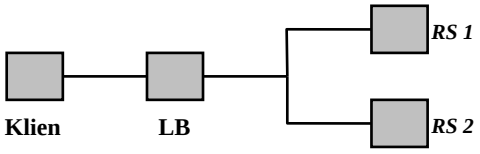
Bahan yang diteliti adalah rata-rata jumlah balasan HTTP per detik (pesat balasan HTTP) dari sistem server web berbasis *cluster* apabila jumlah permintaan HTTP per detik (pesat permintaan HTTP) oleh klien bersifat statis.

*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.

*C. Perancangan Sistem*

Sistem yang dirancang dalam penelitian ini adalah:




Gambar 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 untuk algoritma yang dipakai, yaitu *Never Queue*. 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 algoritma *Never Queue*. Jadi ada satu grafik yang berisi jumlah pesat permintaan HTTP, jumlah pesat balasan HTTP, waktu tanggapan, dan *troughput*. Grafik inilah yang dipakai untuk melihat karakteristik sistem yang dibuat dengan algoritma *Never Queue* dalam mendistribusikan beban kerja web ke sistem server web berbasis *cluster*.

## 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 statis dengan pesat koneksi TCP tunggal.

### *a. Hasil Pengujian Beban Statis*

Pada pengujian ini, pesat permintaan HTTP yang dihasilkan sebesar 1000, 1200, 1400, 1600, 1800, dan 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 *Trial and Error* 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 tetap.

Untuk pengujian beban statis ini, hasilnya adalah:

```
Total: connections 5000 requests 49600 replies 49600 test-duration
50.158 s

Connection rate: 99.7 conn/s (10.0 ms/conn, <=47 concurrent
connections)
Connection time [ms]: min 54.1 avg 187.3 max 418.9 median 186.5
stddev 63.5
Connection time [ms]: connect 12.8
Connection length [replies/conn]: 10.000

Request rate: 988.9 req/s (1.0 ms/req)
Request size [B]: 75.0

Reply rate [replies/s]: min 945.1 avg 989.4 max 1013.4 stddev 19.5
(10 samples)
```

```
Reply time [ms]: response 17.5 transfer 0.0
Reply size [B]: header 241.0 content 44.0 footer 0.0 (total 285.0)
Reply status: 1xx=0 2xx=49600 3xx=0 4xx=0 5xx=0

CPU time [s]: user 4.69 system 41.90 (user 9.4% system 83.5% total
92.9%)
Net I/O: 347.7 KB/s (2.8*10^6 bps)

Errors: total 40 client-timo 40 socket-timo 0 connrefused 0
connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

Ada enam kelompok statistik dalam hasil pengujian di atas yang dipisahkan oleh baris kosong, yaitu:

1. **Hasil Total**

Hasil total yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Total: connections 5000 requests 49600 replies 49600 test-duration
50.158 s
```

Bagian ini menyatakan bahwa ada 5000 koneksi TCP yang dibuat oleh klien, 49600 permintaan yang dikirim keluar, 49600 balasan yang diterima, dan total waktu tes 50,158 detik.

**2. Hasil Koneksi TCP**

Pesat koneksi TCP dan jumlah koneksi TCP bersamaan yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Connection rate: 99.7 conn/s (10.0 ms/conn, <=47 concurrent
connections)
```

Baris ini menunjukkan pesat koneksi sebesar 99.7 koneksi/detik (10,0 milidetik/koneksi), dan paling sedikit ada 47 koneksi yang dibuka secara bersamaan ke server pada suatu waktu.

Statistik waktu hidup satu koneksi TCP penuh yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Connection time [ms]: min 54.1 avg 187.3 max 418.9 median 186.5
stddev 63.5
```

Baris ini menunjukkan statistik waktu-hidup koneksi yang berhasil. Waktu-hidup koneksi adalah waktu yang terhitung sejak koneksi TCP diadakan hingga koneksi ditutup. Sebuah koneksi dinyatakan berhasil apabila ia memiliki paling sedikit satu permintaan yang telah dibalas oleh server. Dari baris di atas dapat kita lihat bahwa waktu-hidup minimum koneksi adalah 54,1 milidetik, rata-rata 187,3 milidetik, maksimum 418,9 milidetik, dan median 186,5 milidetik, dengan deviasi standar 63,5 milidetik.

Rata-rata waktu yang dibutuhkan untuk membentuk suatu koneksi TCP dengan server, termasuk koneksi TCP yang berhasil maupun yang gagal atau tidak mendapat balasan, yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Connection time [ms]: connect 12.8
```

Baris ini menyatakan bahwa dibutuhkan 12,8 milidetik untuk membentuk koneksi TCP dengan server.

Rata-rata balasan HTTP per koneksi TCP yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Connection length [replies/conn]: 10.000
```

Baris ini menyatakan bahwa rata-rata balasan per koneksi TCP adalah 10 (menunjukkan protokol HTTP/1.1 – *persistent connection*). Untuk protokol HTTP/1.0, rata-rata balasan per koneksi TCP adalah 1,0.

**3. Hasil Permintaan HTTP**

Hasil permintaan HTTP yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
  Request rate: 988.9 req/s (1.0 ms/req)
```

Baris ini menunjukkan pesat permintaan HTTP yang dikeluarkan oleh klien ke server, dan waktu yang dibutuhkan untuk mengeluarkan satu permintaan HTTP. Dari hasil di atas dapat kita lihat bahwa pesat permintaan HTTP yang dihasilkan adalah 988,9 permintaan/detik yang berarti, dibutuhkan waktu 1,0 milidetik untuk menghasilkan satu permintaan HTTP.

Rata-rata ukuran permintaan HTTP dalam satuan *byte* yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Request size [B]: 75.0
```

Baris ini menunjukkan bahwa rata-rata ukuran permintaan HTTP yang dihasilkan adalah 75 *byte*.

**4. Hasil Balasan HTTP**

Statistik pesat balasan HTTP yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Reply rate [replies/s]: min 945.1 avg 989.4 max 1013.4 stddev 19.5
(10 samples)
```

Baris ini menunjukkan statistik pesat balasan HTTP: minimum 945,1 balasan/detik, rata-rata 989,4 balasan/detik, dan maksimum 1013,4 balasan/detik, dengan deviasi standar 19,5 balasan/detik.

Waktu respons dan transfer server yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Reply time [ms]: response 17.5 transfer 0.0
```

Baris ini memberikan informasi berapa lama waktu yang dibutuhkan oleh server untuk merespons permintaan klien dan waktu yang dibutuhkan klien untuk membaca balasan server. Waktu respons dihitung sejak *byte* pertama permintaan dikirim hingga *byte* pertama balasan diterima oleh klien. Waktu baca (transfer) adalah waktu yang dibutuhkan untuk membaca keseluruhan balasan, terutama apabila ukuran balasan cukup besar sehingga harus terfragmentasi dalam beberapa segmen TCP. Dari hasil di atas dapat kita lihat bahwa waktu respons adalah sebesar 17,5 milidetik dan waktu transfer sebesar 0 milidetik.

Ukuran kepala, isi, kaki, dan total balasan HTTP dalam satuan *byte* yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Reply size [B]: header 241.0 content 44.0 footer 0.0 (total 285.0)
```

Baris ini menunjukkan bahwa ukuran kepala balasan adalah 241 *byte*, ukuran isi balasan 44 *byte*, ukuran kaki balasan 0 *byte*, dan ukuran total balasan adalah 285 *byte*.

Status balasan HTTP yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Reply status: 1xx=0 2xx=49600 3xx=0 4xx=0 5xx=0
```

Baris ini menunjukkan bahwa ada 49600 balasan dalam kode status 2xx, yang berarti balasan telah "sukses" dikirimkan.

**5. Penggunaan CPU dan Jaringan**

Waktu penggunaan CPU yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
CPU time [s]: user 4.69 system 41.90 (user 9.4% system 83.5% total
92.9%)
```

Baris ini menunjukkan bahwa dibutuhkan waktu 4,69 detik (9,4%) dalam mode "user" dan 41,90 detik (83,5%) dalam mode "system" untuk mengeksekusi program. Idealnya jumlah dalam kedua mode adalah 100%. Apabila prosentase penggunaan CPU secara signifikan kurang dari 100%, maka itu berarti ada program lain yang sedang dijalankan saat program dieksekusi, sehingga pengujian harus diulang.

Nilai *throughput* jaringan yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Net I/O: 347.7 KB/s (2.8*10^6 bps)
```

Nilai *throughput* jaringan dihitung dari jumlah *byte* yang dikirim dan diterima dalam koneksi TCP. Hal ini berarti, hanya bagian *payload* protokol saja yang dihitung (tidak termasuk *header* protokol), tanpa memperhitungkan transmisi-ulang yang mungkin terjadi pada tingkat TCP. Baris di atas menunjukkan bahwa nilai *throughput* jaringan adalah sebesar 347,7 KBps atau 347,7 x 1024 x 8 = 2848358,4 bps ($2{,}8 \times 10^6$ bps = 2,8 Mbps).

**6. Galat**

Statistik galat yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Errors: total 40 client-timo 40 socket-timo 0 connrefused 0
connreset 0
```

Baris ini menunjukkan bahwa total terjadi 40 galat, ada galat yang disebabkan oleh terlewatinya batas waktu (*client-timo* = 40), tidak terjadi kegagalan koneksi TCP akibat terlewatinya batas waktu pada tingkat socket (*socket-timo* = 0), tidak terjadi kegagalan koneksi TCP akibat penolakan server, dan tidak ada kegagalan koneksi TCP akibat penutupan oleh server.

Galat lainnya yang muncul dalam baris mode teks Linux tampak sebagai berikut.

```
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

Baris ini menunjukkan bahwa klien tidak pernah memproduksi beban melebihi batas yang ada dalam *file-descriptor*, klien selalu mendapat nomor *port*, tabel *file-descriptor* tidak pernah penuh, dan tidak terjadi jenis galat lainnya.

Setelah pengujian pertama di atas, selanjutnya diadakan pengujian kedua hingga keenam. Hasilnya tampak dalam Tabel 1 berikut ini.

Tabel 1. **Hasil Pengujian Statis Algoritma *Never Queue***

| Pesat Permintaan HTTP (rps) | Pesat Balasan HTTP (replies/s) | Waktu Tanggapan (ms) | Throughput (Mbps) | Pesat Koneksi TCP (cps) | Galat (galat) |
|---|---|---|---|---|---|
| 1000 | 989,4 | 17,5 | 2,8 | 99,7 | 40 |
| 1200 | 1058,3 | 140,4 | 3,1 | 114,5 | 358 |
| 1400 | 1036,6 | 191,3 | 3,0 | 132,5 | 1063 |
| 1600 | 1015,3 | 204,0 | 3,0 | 151,3 | 1604 |
| 1800 | 1041,9 | 202,7 | 3,0 | 167,9 | 1870 |
| 2000 | 1049,4 | 203,5 | 3,0 | 187,4 | 2182 |

Sumber: **Hasil Pengujian Laboratorium**

## KESIMPULAN

Kesimpulan yang bisa diambil dari penelitian ini adalah. dalam pengujian beban statis dengan pesat permintaan HTTP 1000 rps, 1200 rps, 1400 rps, 1600 rps, 1800 rps, dan 2000 rps: jika dilihat dalam Tabel I maka parameter pesat balasan HTTP cenderung stabil di rata-rata pesat balasan HTTP sebesar 1031,8 replies/s. Sedangkan untuk parameter pesat koneksi TCP, waktu tanggapan, dan galat terjadi kenaikan seiring dengan bertambahnya pesat permintaan HTTP yang dihasilkan. Adapun *througput* rata-rata berada di angka 2,983 Mbps.

## DAFTAR PUSTAKA

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

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

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

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

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

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

G. Gilder. *The Coming Creativity Boom*. (October 23rd, 2008). *http://www.forbes.com/forbes/2008/1110/036.html.*

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