SAINTEKS
Volume 16 No 1, April 2019
ISSN: 0852-1468 (33 – 41)

Pengujian Algoritma *Load Balancing* pada Virtualisasi Server

(Testing the Load Balancing Algorithm on Server Virtualization)

Dimara Kusuma Hakim[1], Johan Kun Riyanto[2], Achmad Fauzan[3]

[1,2,3] Program Studi Teknik Informatika Fakultas Teknik dan Sains
Universitas Muhammadiyah Purwokerto, Indonesia
Jalan Raya Dukuh Waluh PO BOX 202 Kembaran Banyumas 53182

[1]dimarakusumahakim@gmail.com
[2]akbarnara@gmail.com
[3]mr.achmadfauzan@gmail.com

ABSTRAK

User internet lambat laun semakin bertambah, dengan kebutuhan yang banyak atau informasi yang harus dicari melewati internet sekarang tidak lagi jadi masalah dengan adanya penyedia layanan informasi seperti *website*. Masalah datang ketika *user* yang sangat banyak kecepatan mengakses sebuah website dalam waktu yang bersamaan atau *concurrent acces* menjadi kendala. Hal itu bisa ditangani menggunakan metode *load balancing* dan juga virtualisasi server. *Load balancing* adalah metode atau teknik mendistribusikan beban *traffic* pada dua atau lebih jalur koneksi secara seimbang. Agar *traffic* berjalan optimal dan stabil dan mengatasi c10k *problem*, yaitu mengoptimalkan *socket* jaringan pada saat menerima request mencapai 10 ribu koneksi dalam waktu yang bersamaan. Virtualisasi server adalah metode yang dilakukan untuk menjadikan beberapa server fisik menjadi satu wadah server fisik, untuk membagi beban komputasi dalam satu wadah server fisik. Hasil yang didapat dengan uji kecepatan akses secara bersamaan menggunakan aplikasi *webserver stress tool* dengan simulasi user yang berbeda-beda yaitu 25 *user*, 55 *user*, 75 *user*, 100 *user*, 125 *user*, 250 *user* dan 1000 *user*. Hasil dari uji kecepatan akses tersebut di uji lagi perbedaan rata-rata menggunakan uji ANOVA, yang yang mana memungkinkan peneliti untuk menguji hipotesis perbandingan rata-rata lebih dari dua kelompok dari algoritma *round robin*, *weighted round robin* dan *leastconn* dengan hasil, bahwa pada saat 25 dan 55 *user* algoritma *round robin* lebih baik dari pada *weighted round robin* dan *leastconn*, namun pada saat 75, 100, 125, 250 dan 1000 *user* algoritma *weighted round robin* lebih baik dan optimal dari pada algoritma *round robin dan leastconn* pada saat akses ribuan data dari web server dan algoritma *leastconn* tidak memberikan performa yang baik terbukti dari ketujuh pengujian dan simulasi yang berbeda tetap tidak memberikan hasil rata-rata yang kurang dari algoritma *round robin* dan *weighted roundrobin*. Dari ketujuh simulasi yang telah dilakukan. Maka penggunaan algoritma *weighted round robin* lebih baik dari pada *round robin* dan *leastconn*.

Kata Kunci : *Virtualisasi Server, Website, Proxmox, Load Balancing*

ABSTRACT

Internet users will gradually increase, with a lot of needs or information that must searched through the internet. now, there is no problem with information service providers such as websites. The problem comes when users who are very much speed accessing a website at the same time or concurrent access will be an obstacle. It can be handled using load balancing methods and also server virtualization. Load balancing is a

method or technique of distributing traffic loads on two or more connection lines equally. So that traffic runs optimally, stable and overcomes c10k problems, namely optimizing network sockets when receiving requests reaching 10 thousand connections at the same time.Server virtualization is amethod used to make several physical servers into one physical server container, to divide the computational load in one physical server container. The results obtained by testing the access speed simultaneously using the webserver stress tool application with different user simulations are 25 users, 55 users, 75 users, 100 users, 125 users, 250 users and 1000 users.Furthermore, the results of the access speed test were re-tested the average difference using the ANOVA test, which allowed researchers to test the average comparison hypothesis of more than two groups of the round robin algorithm, weighted round robin and leastconn. the results that at the time 25 and 55 users of round robin algorithm were better than weighted round robin, but when 75, 100, 125, 250 and 1000 users of weighted round robin algorithm were better and optimal than the round robin algorithm when accessing thousands of data from the web server and leastconn algorithm does not provide good performance as evidenced from the six different tests and simulations still did not provide average results that were less than the round robin algorithm and weighted roundrobin.it can be concluded that from the six simulations that have been carried out, so the use of weighted round robin algorithm was better than round robin and least connection..

***Keywords** : Server Virtualization, Website, Proxmox, Load Balancing*

PENDAHULUAN

Perkembangan teknologi memberikan pengaruh besar bagi kehidupan salah satunya dalam penyampaian suatu informasi. Kebutuhan akan informasi dan komunikasi menjadi salah satu kebutuhan pokok dalam kehidupan sehari – hari. Kebutuhan akan koneksi internet semakin meningkat seiring berlimpahnya perangkat yang bisa dipakai untuk menjelajahi dunia maya (Andreolini *et al*,2004). Semakin banyak yang mengakses melalui situs web membuat beban kerja yang lebih pada suatu penyedia layanan yang disebut web server dan menjadi kurang optimal. Suatu *single* server bisa mengalami kegagalan yang disebabkan oleh meningkatnya jumlah *request* yang mencapai ribuan bahkan jutaan pada waktu bersamaan(*concurrency access*) atau disebut dengan overload. Hal ini akan merugikan pihak yang mempercayakan situsnya pada suatu web server. Karena situs tersebut tidak dapat diakses untuk waktu tertentu(Rosalia *et al*, 2016).

Request yang mencapai ribuan bahkan jutaan pada waktu yang bersamaan tersebut menjadi beban kinerja pada server meningkat dan berujung dengan down-nya server. Masalah tersebut dapat dioptimalkan dengan *load balancing* dengan banyaknya pilihan algoritma yang dapat digunakan. Dapat membagi beban traffic menuju web server. Terlepas dari *load balancing* terdapat c10k problem yaitu masalah mengoptimalkan soket jaringan untuk menangani sepuluh ribu koneksi secara bersamaan. Perhatikan bahwa koneksi *concurrency access* tidak sama dengan permintaan per detik, meskipun mereka serupa menangani banyak permintaan per detik membutuhkan membutuhkan *throughput* tinggi, sementara jumlah *concurrency access* yang tinggi membutuhkan penjadwalan koneksi yang efisien. Dengan *load balancing* dapat membuat penjadwalan dengan algoritma yang dapat disesuaikan dengan kebutuhan.

Virtualisasi server ini dibuat untuk membuat beberapa server disatukan dalam satu wadah satu server fisik dan *high availability* server menggunakan metode *load balancing* algoritma *round robin*, *weighted round robin* dan *leastconn* untuk meningkatkan kehandalan dan ketersediaan. Kemudian akan dibandingkan performansi ketiga algoritma

yaitu algoritma *roundrobin*, *weighted roundrobin* dan *leastconn* dengan uji ANOVA membandingkan kelompok lebih dari dua.

Virtualisasi

Virtualisasi/*Virtualization* adalah bentuk teknik sebuah atau cara untuk membuat sesuatu dalam bentuk virtual, tidak seperti kenyataan yang ada. Virtualisasi juga digunakan untuk mengemulasikan perangkat fisik komputer, dengan cara membuatnya seolah – olah perangkat tersebut tidak ada (disembunyikan) atau bahkan menciptakan perangkat yang tidak ada menjadi ada.

Perangkat lunak virtualisasi adalah sebuah program yang memiliki kemampuan untuk membuat sebuah komputer secara virtual (Purnoma, 2010). Disebut komputer virtual karena komputer itu tidak ada secara fisik, dengan komputer virtual ini dimungkinkan untuk menginstal OS lain. Misalkan komputer yang menggunakan *Windows* 7 dapat diinstalasi OS lain seperti *Linux*, Mac, BSD atau *Windows* versi lain menggunakan komputer virtual tersebut. Salah satu dari sekian banyak perangkat lunak virtualisasi adalah VirtualBox dan Proxmox.

Proxmox

Menurut Suryono dan Afif (2012) menyatakan bahwa proxmox merupakan software *open source virtualization platform* untuk menjalankan virtual *appliance* dan virtual machine. *Proxmox VE (Virtual Environment)* adalah distro khusus yang didekasikan secara khusus sebagai mesin virtualisasi yaitu *KVM* dan *OpenVZ*. *Proxmox VE* menggunakan *Container Virtualization* dan *Full Virtualization* :

1. *Container Virtualization (OpenVZ)* merupakan teknologi yang disarankan untuk menjalankan server linux. *OpenVZ* membuat beberapa container yang secure dan terisolasi (disebut juga *CT,VE* atau *VPS*). Setiap Container melakukan dan mengeksekusi persis seperti layaknya sebuah *stand alone server*, sebuah *container* dapat di-*reboot* secara independen dan memiliki akses *super user, IP address*, memori, proses, file, aplikasi, *system library* dan konfigurasi tersendiri.
2. *Full Virtualization (KVM)* merupakan singkatan dari *(Kernel-based Virtual Machine)* adalah solusi virtulaisasi penuh untuk hardware berbasis x86 yang memiliki ekstensi virtualisasi (Intel VT atau AMDV CPU). Setiap virtual *machine* memiliki *hardware* pribadi yang virtual: *network card*, *disk*, adapter grafis, dll. *KVM* mirip dengan XEN akan tetapi *KVM* merupakan bagian dari Linux dan menggunakan *system scheduler* dan *memory* managemen regular dari Linux.

Fitur *central web based management* menurut Suryono dan Afif (2012) dengan adanya fitur *web based* management tidak perlu menginstal alat manajemen yang terpisah, semua dapat dilakukan melalui *web browser*. Dengan tampilan konsol terintegrasi pada virtual mesin , integrasi dan manajemen pada cluster *VE* yang baik, ditambahkan teknologi *AJAX* untuk mengupdate secara dinamis pada sumber daya dan akses yang aman, ke semua mesin virtual melalui enkripsi *SSL* (https).

Web Sserver

Menurut Lukitasari dan Oklilas (2010) *web server* adalah perangkat lunak yang menjadi tulang belakang dari *world wide web (www). Web server* menunggu permintaan dari client yang menggunakan *browser* seperti Netscape Navigator, Internet Explorer, Modzilla, dan program *browser* lainnya. Jika ada permintaan dari *browser*, maka *web server* akan memproses permintaan itu kemudian memberikan hasil prosesnya berupa data yang diinginkan kembali ke *browser*. Data ini mempunyai format yang standar, disebut dengan format SGML (*standar general markup language*). Data yang berupa

format ini kemudian akan ditampilkan oleh *browser* sesuai dengan kemampuan *browser* tersebut.

HAproxy

HAproxy adalah produk *open source* yang digunakan untuk menciptakan sistem *load balancing* atau pembagian beban secara otomatis dan *failover* dari aplikasi yang berbasis TCP dan HTTP. Perangkat lunak ini sangat cocok digunakan untuk website yang *traffic* hariannya tinggi. HAProxy memiliki beberapa fitur sebagai berikut :

a. Dapat dibuat master dan *slave load balancing*, jika *load balancing* master mati makadapat menggunakan *load balancing slave*.
b. Mendukung load balancing untuk beberapa server.
c. Terdapat statistik untuk melakukan monitoring.

Implementasi HAProxy diterapkan pada *server front-end*. *Server front-end* umumnya adalah server yang memiliki IP statis dan teregistrasi dengan DNS. Aplikasi server dapat dipasang bersama didalam front-end atau terpasang secara terpisah. *Front-end* berfugsi untuk menghubungkan klien dengan aplikasi server yang tersedia (Bahrudin dan Rosmansyah , 2009).

Load Balancing

Dalam Perdana *et al* (2017), menyatakan bahwa teknik untuk mendistribusikan trafik pada dua atau lebih jalur koneksi secara seimbang. Agar trafik berjalan optimal, memaksimalkan throughput, memperkecil waktu tanggap dan menghindari *overload* pada salah satu jalur koneksi.

Round Robin

Algoritma *Round Robin* mendistribusikan pekerjaan secara merata ke semua prosesor slave. Semua pekerjaan ditugaskan untuk prosesor slave berdasarkan urutan *Round Robin*, artinya prosesor itu melakukan pemilihan secara seri dan akan kembali ke prosesor pertama jika prosesor yang terakhir telah tercapai. Pemilihan prosesor dilakukan secara lokal pada masing – masing prosesor, terlepas dari alokasi prosesor lain(Deepika *et al*, 2014).

Weighted Round Robin

Pengembangan dari algoritma Round Robin, sedikit perbedaannya adalah algoritma ini bisa membagi beban lebih tinggi ke server atau *cluster* yang memiliki *resource* yang lebih besar atau dapat melakukan perhitungan perbedaan kemampuan *processing* dari masing – masing server anggota cluster. Administrator memasukkan secara manual parameter beban yang akan ditangani oleh masing – masing server anggota *cluster,* kemuadian *scheduling sequence* secara otomatis dilakukan berdasarkan beban server. *Request* kemudian diarahkan ke server yang berbeda(Perdana *et al*, 2017).

Leastconnection

Algoritma Leastconn adalah salah satu algoritma penjadwalan dinamis,karena algoritma leastconn perlu menghitung koneksi langsung untuk setiap server secara dinamis(Mustafa, 2017).

ANOVA

Santoso (2017) menyatakan bahwa ANOVA atau uji F digunakan untuk pengujian lebih dari dua sampel yang bertujuan untuk mengetahui apakah ada perbedaan yang signifikan antara rata-rata hitung beberapa kelompok data. Pengujian statistik dilakukan dengan tujuan untuk mengetahui apakah ada pengaruh atau perbedaan yang signifikan dan seberapa besar jika ada pengaruh terhadap output yang dihasilkan dengan membandingkan tingkat *error*.

Taniredja & Mustafidah (2011) menyatakan bahwa anava atau disebut juga anova (*Analysis of Variance*) merupakan perluasan dari uji rata-rata k sampel, dengan k>2.

- $H_0 : \mu_1 = \mu_2 = \ldots = \mu_k$ VS H_1 : minimal 2 *mean* tidak sama.
- $F_{hit} = \frac{MST}{MSE} \sim F_{k-1,\ N-k}$

  Dimana MST = *Mean Square of Treatment*

  MSE = *Mean Square of Error*

Pada anava jika H_0 ditolak maka masih mempunyai pekerjaan untuk menentukan *mean-mean* mana saja yang berbeda. Untuk itu kita lakukan dengan MCA (*Multiple Comparison Analysis*) atau Analisis Pembandingan Ganda.

C10K *Problem*

Masalah C10K mengoptimalkan soket jaringan untuk menangani sejumlah besar klien atau *user* pada saat bersamaan(Afis *et al*, 2019).

METODE

Penelitian ini dilaksanakan bertempat di Lab. Sistem Cerdas Gedung Teknik Lantai 4 Universitas Muhammadiyah Purwokerto. Alat yang digunakan pada penelitian ini dibagi menjadi dua yaitu perangkat keras dan Lunak. Perangkat keras antara lain ada *Processor* 2 CPU *cores, core i5* 32/64 bit, RAM 4 Gb, harddisk 500 Gb, Network Card Gigabit atau 10 Gigabit. Sedangkan perangkat lunak ada windows 7, Proxmox VE 5.3, Ubuntu server, Filezila, Xampp, HAProxy dan Webserver stress tool.

Pembagian *Web Server* Algoritma *Round Robin*

Pembagian pada Tabel 3. ini menggunakan algoritma roundrobin tanpa pembagian beban yang ditentukan dengan beberapa user, yang disimulasikan ada 25 beban *user*, 55 *user*, 75 *user*, 100 *user*, 125 *user* dan 250 user mengakses website secara bersamaan dengan alat uji *webserver stress tool*. Algoritma *Roundrobin* ini mengarahkan ke 2 server secara bergantian.

Tabel 1. Jumlah *User* Akses Data Algoritma *Round Robin*

No	Jumlah User	Algoritma
1	25	*Round Robin*
2	55	*Round Robin*
3	75	*Round Robin*
4	100	*Round Robin*
5	125	*Round Robin*
6	250	*Round Robin*
7	1000	*Round Robin*

Pembagian *Web Server* Algoritma *Weighted Round Robin*

Pada Tabel 2, algoritma ini mampu membagi beban user yang mengakses aplikasi sesuai dengan yang kita tentukan.

Pembagian *Web Server* Algoritma *Leastconnection*

Pada Tabel 3 algoritma *Leastconn* adalah salah satu algoritma penjadwalan dinamis, karena algoritma leastconn perlu menghitung koneksi langsung untuk setiap server secara dinamis.

Tabel 1. Jumlah *User* Akses Data Algoritma *Weighted Round Robin*

No	Jumlah User	Algoritma
1	25 (13-12)	*Weighted Round Robin*
2	55 (27-28)	*Weighted Round Robin*
3	75 (37-38)	*Weighted Round Robin*
4	100 (50-50)	*Weighted Round Robin*
5	125 (60-65)	*Weighted Round Robin*
6	250(125-125)	*Weighted Round Robin*
7	1000(500-500)	*Weighted Round Robin*

Tabel 3. Jumlah *User* Akses Data Algoritma *Leastconn*

No	Jumlah User	Algoritma
1	25	*Leastconn*
2	55	*Leastconn*
3	75	*Leastconn*
4	100	*Leastconn*
5	125	*Leastconn*
6	250	*Leastconn*
7	1000	*Leastconn*

Uji ANOVA

Pengujian dilakukan dengan menggunakan uji statistik ANOVA yang merupakan perluasan dari uji rata rata untuk menguji kesamaan rata-rata k sampel, dengan k > 2. Jika Ho ditolak maka masih mempunyai pekerjaan untuk menentukan rata-rata mana saja yang berbeda. Untuk itu perlu dilakukan MCA (*Multiple Comparison Analysis*) atau Analisis Perbandinganan Ganda (Tanireja & Mustafidah, 2011). Berikut tahapan yang dilakukan adalah:

1. Menentukan hipotesis
   H_0 : Tidak ada perbedaan antara rata-rata kecepatan akses ketiga algoritma *load balancing.*
   H_1 : Ada perbedaan antara rata-rata kecepatan akses ketiga algoritma *load balancing*
2. Menentukan nilai alpha (α) (dalam penelitian ini digunakan α = 5% atau 0.05 adalah ukuran standar yang sering digunakan dalam penelitian).
3. Menentukan alat uji
   Alat uji yang digunakan yaitu menggunakan uji F sebagai berikut:
   $H_0 : \mu_1 = \mu_2 = \ldots = \mu_k$ vs H_1 : minimal 2 *mean* tidak sama.

$$F_{hit} = \frac{MST}{MSE} \sim F_{k-1,N-k} \quad \text{.....(1)}$$

   MST = *Mean Square of Treatment*
   MSE = *Mean Square of Error*
4. Pengambilan kesimpulan

Kesimpulan diambil berdasarkan nilai signifikan yang diperoleh (sig.) dengan ketentuan H_0 ditolak jika nilai sig $< \alpha$ dan H_0 diterima bila nilai sig > a.

HASIL DAN PEMBAHASAN

Pegujian dilakukan menggunakan aplikasi *webserver stress tool*, algoritma yang diuji adalah algoritma *round robin* Tabel 5, *weighted round robin* Tabel 6 dan *leastconn.*

Pengujian ini adalah mengakses data *sales record* pada web server virtual dengan proxmox v 5.3, jumlah baris ada 5001, 14 kolom dan dengan ukuran 1,5 mb.

Tabel 4. Hasil total rata-rata per *user* Algoritma *Round Robin*

No	Jumlah User	Jumlah rata-rata Kecepatan(ms)
1	25	3,459
2	55	6,296
3	75	9,550
4	100	6,435
5	125	8,519
6	250	27,911
7	1000	43,904

Tabel 5. Hasil total rata-rata per *user* Algoritma *Weighted Round Robin*

No	Jumlah *User*	Jumlah rata-rata Kecepatan(ms)
1	25 (13-12)	7,588
2	55 (27-28)	13,811
3	75 (37-38)	7,210
4	100 (50-50)	7,149
5	125 (60-65)	7,951
6	250(125-125)	6,382
7	1000(500-500)	41,342

Tabel 6. Hasil total rata-rata per *user* Algoritma *Leastconn*

No	Jumlah *User*	Jumlah rata-rata Kecepatan(ms)
1	25	4,112
2	55	6,643
3	75	8,899
4	100	11,147
5	125	13,475
6	250	8,980
7	1000	39,196

Pada hasil Tabel 7 menunjukkan bahwa signifikan terjadi pada ketujuh pengujian simulasi dan ketiga algoritma < a (0,05 atau 5 %), sehingga terdapat perbedaan yang signifikan antara ketiga algoritma load balancing yang diuji yaitu *round robin*, *weighted round robin* dan *leastconn.* Maka perlu di lakukan MCA*(Multiple Comparison Analysis)* untuk mengetahui mana saja rata-rata yang berbeda, ada beberapa pilihan untuk menggunakan MCA. namun pada pengujian ini menggunan Duncan yaitu untuk mengetahui detail rata-rata terkecil dan terbesar dari ketiga algoritma. Berikut adalah hasil dari MCA dari semua algoritma pada Tabel 8.

Tabel 7. Hasil Signifikan per User

No	User	Hasil Uji ANOVA (Sig.)
1	25	0,000
2	55	0,000
3	75	0,000
4	100	0,001
5	125	0,000
6	250	0,000
7	1000	0,000

Tabel 8. Hasil MCA Duncan per User

No	User	Hasil Duncan per Algoritma		
		Round Robin	***Weighted Round Robin***	***Leastconn***
1	25	3,46	12,74	4,11
2	55	6,30	19,98	8,42
3	75	8,90	7,71	8,90
4	100	11,58	9,98	11,15
5	125	14,41	12,82	14,04
6	250	31,30	7,37	9,72
7	1000	37,36	37,22	45,30

Pada hasil Tabel 8 maka dapat disimpulkan algoritma *round robin* pada saat user 25 dan 55 paling kecil rata-ratanya, maka akses kecepatan pada saat yang bersamaan memberikan performa yang baik dapat berjalan dengan lancar dan optimal. Pada saat user 75, 100, 125 250 dan 100 algoritma *weighted round robin* paling kecil rata-ratanya, maka akses kecepatan pada saat yang bersamaan memberikan performa yang baik dapat berjalan dengan lancar dan optimal. Namun pada algoritma *leastconn* tidak memberikan performa yang baik diantara semua user yang disimulasikan.

KESIMPULAN

Berdasarkan analisis yang telah dilakukan, pengujian kecepatan akses dengan algoritma *round robin*, *weighted round robin* dan *leastconn*, dan uji ANOVA untuk mengetahui perbedaan rata-rata signifikan atau tidak, didapat kesimpulan sebagai berikut:

1. Hasil algoritma leastconn tidak menghasilkan perbedaan yang signifikan sama sekali
2. Hasil dari algoritma round robin pada saat user 25 dan 55 menghasilkan perbedaan yang signifikan, namun
3. Hasil algoritma weighted round robin pada saat user 75, 100, 125, 250 dan 1000 sehingga algoritma ini paling memberikan performa yang optimal.

DAFTAR PUSTAKA

Andreolini, M., Casalicchio, E., Colajanni, M., & Mambelli, M. (2004). *A Cluster-Based Web System Providing Differentiated and Guaranteed Services*. Italy.

Deepika, Wadhwa, D., & Kumar, N. (2014). Performance Analysis of Load Balancing Algorithms in. *Advance in Electronic and Electric Engineering*, *4*(1), 59–66.

Mustafa, D. M. (2017). Load balancing algorithms round-robin (rr), leastconnection, and least loaded efficiency. GESJ: Computer Science and Telecommunications 2017|No.1(51).

Perdana, F. P., Irawan, B., Latuconsina, R., Teknik, F., Telkom, U., & Balancing, L. (2017). *Analisis Performansi Load Balancing dengan Algoritma Weighted Round Robin pada Software Defined Network (SDN)*. *4*(3), 4161–4168.

Purnoma. (2010). Membangun Virtual PC dengan VirtualBox. Penerbit Andi Yogyakarta. Yogyakarta.

Rosalia, M., Munadi, R., & Mayasari, R. (2016). *Implementasi High Availability Server Menggunakan Metode Load Balancing dan Failover pada Virtual Web Server*. *3*(3), 4496–4503.

Santoso, S. (2017). *Menguasai Statistik dengan SPSS 24*. Jakarta: PT. Elex Media Komputindo.

Taniredja, T. & Mustafidah, H., (2011). *Penelitian Kuantitatif (Sebuah Pengantar)*. Bandung: Alfabeta.