Metode *Load Balancing Haproxy* pada OpenNebula

Muh Syahrir[1)], Dahlia Nur[2)], Kasim[3)]
[1,2,3] Program Studi Teknik Komputer dan Jaringan, Jurusan Teknik Elektro, Politeknik Negeri Ujung Pandang
[1)]muhsyahrir005@gmail.com, [2)]dahlia@poliupg.ac.id, [3)]kasim@poliupg.ac.id

Abstrak

Server tunggal opennebula dapat *down* atau *overload* jika user yang akses secara bersamaan, banyaknya data yang dibuat, banyaknya konsumsi daya, processor kurang dan bahkan *storage* pada server kurang memadai sehingga beban kerja pada server tidak dibagi secara merata. Tujuan dari penelitian ini agar dapat meminimalisir server *down* akibat peningkatan beban kerja dan meningkatkan kinerja server yang lebih cepat dengan menggunakan *load balancing haproxy* pada server untuk membagi permintaan layanan yang meminta *request* pada server. Algoritma yang digunakan yaitu *least bandwith* adalah algoritma dikonfigurasi menggunakan metode *bandwith* paling sedikit kemudian diteruskan ke server dalam satuan waktu sedangkan *least response time* adalah algoritma yang mengarahkan user ke server dengan *response time* yang terendah, dengan pengujian menggunakan metode *failover* dan parameter pengukuran berdasarkan standar TIPHON yaitu *response time* (*ms*), *packet loss/error rate* (%) dan *throughput* (*Kbps*). Berdasarkan Hasil pengukuran load balancing dengan request user 11100 dengan standar berdasarkan *Telecommunications and Internet Protocol Harmonization Over Networks* (TIPHON) dimana *response time* untuk algoritma *round robin* sebesar 446,89 ms masuk kategori sedang, algoritma *least bandwith* sebesar 402,45 ms kategori sedang dan *least response time* sebesar 393,67 ms masuk dalam kategori baik. Kemudian *packet loss/Error rate* (%) untuk algoritma *round robin* 20,37 %, *least bandwith* 12,65 % dan *least response time* 7,12 % masuk kategori baik berdasarkan standar. Sedangkan *throughput round robin* 2894,1 *Kbps*, *least bandwith* 3618 *Kbps* dan *least response time* 3589 *Kbps* masuk dalam indeks 4 kategori sangat baik

Keywords: *Cloud Computing Opennebula, Load Balancing Haproxy, Response Time, Packet Loss/Error Rate, Throughput, Round Robin, Least Bandwith, Least Response Time*

I. PENDAHULUAN

Kemajuan ilmu pengetahuan dan teknologi di bidang penyedia informasi semakin meningkat terutama kebutuhan akan akses internet. Banyak di jumpai penggunaan Internet terutama di Warnet, Kantor-kantor, Sekolah maupun Kampus menggunakan lebih dari satu koneksi dalam berlangganan Internet [1].

Terutama *Cloud Computing* mengalami peningkatan popularitas yang mengesankan baik di industri perangkat lunak maupun dunia penelitian. Fitur paling menarik yang dihadirkan *Cloud Computing* dari sudut pandang klien *Cloud* seperti memudahkan dalam manajemen bisnis, biaya cenderung lebih hemat, mempunyai fitur back-up dan pemulihan data dan memiliki kapasistas penyimpanan tanpa batas contohnya yang populer adalah *Cloud Computing* OpenNebula [2]. Untuk melakukan perkiraan secara *realtime*, dibutuhkan pengolahan data secara *streaming.*

OpenNebula adalah sebuah layanan *Cloud Computing* yang menjadi mesin virtual dengan menyediakan grup yang *efisien*, dinamis, terukur yang berinteraksi dengan server virtual dan fisik. Server tunggal pada opennebula mengalami *down* jika banyak user yang akses secara bersamaan, banyaknya data yang dibuat, banyaknya konsumsi daya, processor kurang dan bahkan *storage* pada server kurang memadai. oleh karena itu menawarkan kesederhanaan dalam memberikan solusi yang lengkap untuk membuat dan melakukan manajemen virtualisasi data center dalam tingkat *enterprise* dengan proses *cluster* server *load balancer* [3].

Load balancing pada server merupakan salah satu cara yang dapat digunakan untuk meningkatkan kinerja dan ketersediaan server, yaitu dengan membagi permintaan layanan yang datang ke beberapa server sekaligus, sehingga beban yang ditanggung oleh masing-masing server lebih sedikit [4]. Dalam menerapkan *load balancing haproxy* pada Opennebula permintaan paket dapat diteruskan dari pengguna ke server. Adapun algoritma yang digunakan yaitu *least response time* dan *least bandwith*. Algoritma *least response time* merupakan algoritma yang mengarahkan user ke server dengan least response time dan waktu respon terendah. *Least response time* digunakan dimana dua parameter (koneksi paling tidak aktif dan waktu respon terendah) sedangkan algoritma least *bandwith* merupakan algoritma yang dikonfigurasi menggunakan metode *bandwith* paling sedikit kemudian diteruskan ke server dalam *megabit per detik* (*Mbps*) [5].

Penelitian sebelumnya yang dilakukan oleh (Thapliyal & Dimri, n.d.) [6]. dengan berjudul "*Load Balancing in Cloud Computing Based on Honey Bee Foraging Behavior and Load Balance Min-Min Scheduling Algorithm*". Algoritma yang digunakan adalah min-min load balancing untuk menyeimbangkan beban di jaringan komputasi awan dimana menghasilkan hasil komparatif dengan memperhatikan parameter seperti konsumsi daya, waktu respon, skalabilitas dan daya tahan dalam perencanaan sumber daya terintegrasi. Penelitian yang dilakukan oleh (Madakatte & Nagesh) dengan judul "*Arr And Fusion Based Virtual Machine Scheduling Techniques For Eucalyptus Cloud*" yaitu menggunakan teknik penjadwalan

mesin *Eucalyptus* yaitu *Advanced Round-Robin (ARR)* dan metodologi Fusion [7].

Dengan demikian penelitian ini dirancang dengan Metode *load balancing haproxy* pada Opennebula menggunakan *least response time* dan *least bandwith* agar supaya web server dapat meminimalisir terjadinya server down akibat peningkatan jumlah beban kerja pada server dan meningkatkan kinerja server yang lebih cepat.

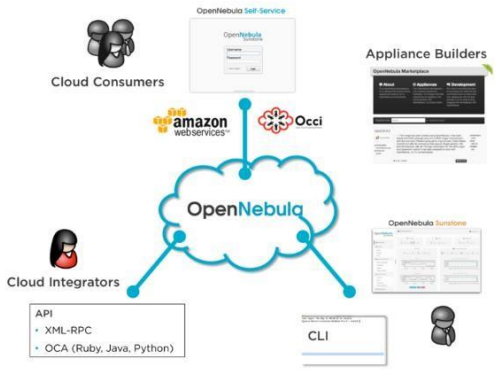
II. KAJIAN LITERATUR

A. OpenNebula

OpenNebula adalah sebuah layanan cloud yang dikembangkan untuk kebutuhan virtualisasi pusat data dan menjadi mesin virtual yang menyediakan grup yang efisien, dinamis, terukur yang berinteraksi dengan server virtual dan fisik [3]. Adapun fitur yang disediakan adalah sebagai berikut.

II. A. Interface yang disediakan oleh OpenNebula

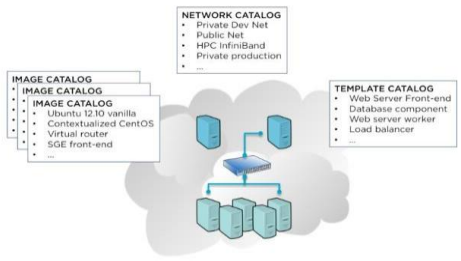
OpenNebula menyediakan banyak interfaces berbeda yang dapat digunakan untuk berinteraksi dengan fungsionalitas yang ditawarkan untuk mengelola sumber daya fisik dan virtual. Ada empat perspektif utama yang berbeda untuk berinteraksi dengan OpenNebula seperti Gambar 1



Gambar 1. Interface OpenNebula

II. B. OpenNebula kepada Konsumen Cloud

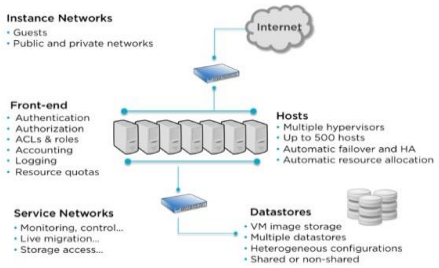
OpenNebula menyediakan platfrom cloud multi-penyewa yang kuat. Skalabel dan aman untuk pengiriman cepat dan elastisitas sumber daya virtual. Aplikasi multi-tier dapat digunakan sebagai peralatan virtual pra-konfigurasi dari katalog seperti Gambar 2



Gambar 2. Konsumen Cloud OpenNebula

II. C. OpenNebula kepada Operator Cloud

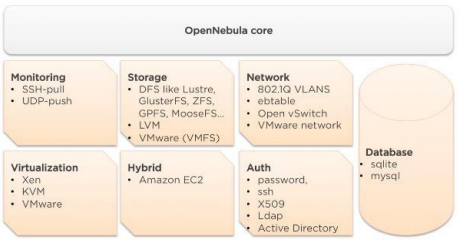
OpenNebula terdiri dari subsitem seperti pada Gambar 3



Gambar 3. Operator Cloud OpenNebula

II. D. OpenNebula kepada Pembuat Cloud

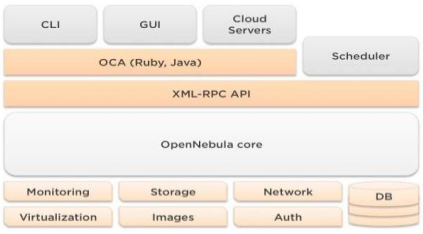
OpenNebula menawarkan dukungan luas untuk komoditas dan hypervisor tingkat Perusahaan, pemantauan, penyimpanan, jaringan dan layanan manajemen pengguna seperti Gambar 4



Gambar 4. Pembuat Cloud OpenNebula

II. E. OpenNebula kepada Cloud Integrators

OpenNebula sepenuhnya independent terhadap platfrom dan menawarkan banyak alat untuk integrator cloud seperti Gambar 5



Gambar 5. Cloud Integrators OpenNebula

B. Quality of Service (QoS)

Quality of Service (QoS) adalah sebuah metode pengukuran tentang seberapa baik jaringan dan merupakan suatu untuk mendefinisikan karakteristik dan sifat dari satu servis kemudian mengacu pada tingkat kecepatan dan kehandalan penyampaian berbagai jenis beban data di dalam sebuah komunikasi [14]. Pada Tabel 1 menunjukkan nilai indeks presentase QoS dimana setiap indeks *presentase* menunjukkan level yang berbeda.

Tabel 1. Indeks Parameter *QoS*

Kategori	Presentase	Indeks
Buruk	25 – 49,75	1 – 1,99
Sedang	50 – 74,75	2 – 2,99
Baik	75 – 94,75	3 – 3,79
Sangat Baik	95 - 100	3,8 - 4

(Sumber: TIPHON)

Terdapat beberapa parameter sesuai standar Telecomunications and Internet Protocol Harmonization Over Networks (TIPHON) untuk mengukur kinerja layanan baik server maupun jaringan internet yaitu:

II. A. *Throughput*

Throughput merupakan kecepatan transfer data yang dikirim untuk suatu titik jaringan atau dari suatu titik ke titik jaringan lainnya dalam satuan bps (*bit per second*) [15]. Besaran *throughput* dapat diperoleh dalam persamaan (1) sebagai berikut

$$Throughput = \frac{jumlah\ data\ yang\ dikirim}{waktu\ pengiriman\ data} \quad (1)$$

Berdasarkan standar TIPHON nilai throughput dikategorikan menjadi empat bagian yang dapat dilihat pada Tabel 2.

Tabel 2. Kategori *Throughput*

Kategori Throughput	Besar Throughput (bps)	indeks
Buruk	<25	1
Sedang	50	2
Baik	75	3
Sangat Baik	100	4

(Sumber: TIPHON)

II. B. *Latency/Response Time*

Latency/Response Time merupakan parameter QoS yang menunjukkan waktu yang dibutuhkan paket untuk mendapatkan jarak dari sumber ke tujuan. Latency pada suatu jaringan internet dapat dijadikan acuan untuk menilai kualitas dalam proses transmisi data [16]. Berdasarkan *Telecommunications and Internet Protocol Harmonization Over Networks* (TIPHON) besarnya *latency* dapat dilihat pada persamaan (2) sebagai berikut

$$Latency = \frac{Total\ Latency}{Jumlah\ Total\ Packet} \quad (2)$$

Response Time versi TIPHON dapat dikelompokkan menjadi empat kategori seperti yang terlihat pada Tabel 3

Tabel 3. Kategori *Latency*

Kategori Latency	Besar Latency (ms)	Indeks
Buruk	> 450	1
Sedang	> 350 - 450	2
Baik	> 150 - 300	3
Sangat Baik	≤ 150	4

(Sumber: TIPHON)

II. C. *Error Rate/Packet Loss*

Packet loss merupakan salah satu parameter dari QoS yang menggambarkan suatu keadaan yang menunjukkan total paket yang gagal atau hilang. *Presentase* permintaan yang gagal diterima oleh server atau parameter yang menunjukkan jumlah koneksi yang tidak berhasil dibuat karena gagal dieksekusi atau diproses oleh server *back-end* dan direpresentasikan dalam persen [15]. Adapun persamaan matematis untuk mendapatkan nilai *packet loss* dapat dilihat pada persamaan (3) sebagai berikut

$$Packet\ Loss = \frac{Paket\ Data\ Dikirim - Paket\ Data\ Diterima}{Paket\ Data\ Dikirim} \quad (3)$$

Berdasarkan standar TIPHON nilai *packet loss* dibagi menjadi empat indeks yang dapat dilihat pada Tabel 4

Tabel 4. Kategori *Packet Loss*

Kategori *Packet Loss*	Besar *Packet Loss* (%)	Indeks
Buruk	25	1
Sedang	15	2
Baik	3	3
Sangat Baik	0	4

(Sumber: TIPHON)

C. Jmeter

Apache Jmeter adalah aplikasi open source berbasis java yang menyediakan fitur untuk melakukan *performance test* semua parameter [17].

III. METODE PENELITIAN

Prosedur penelitian ditunjukkan pada Gambar 6. Gambar tersebut merangkum rangkaian pelaksanaan penelitian secara keseluruhan dengan menggunakan metode *waterfall*, dimana model ini berkembang secara sistematis dari satu tahap ke tahap berikutnya.

Gambar 6. Prosedur Penelitian

A. Alat dan Bahan

Beberapa alat dan bahan yang digunakan pada penelitian ini dikelompokkan dalam dua kategori yaitu perangkat keras (*Hardware*) dan perangkat lunak (*Software*) seperti pada Tabel 5 dan Tabel 6.

Tabel 5. Kebutuhan Perangkat Keras (*Hardware*)

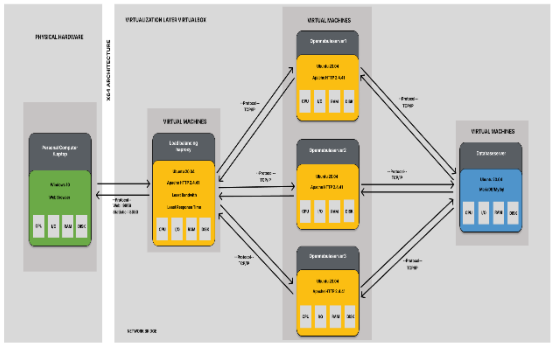
No.	Perangkat Keras	Keterangan
1.	Laptop: a. Minimal *Processor* Intel Inside 2.7 GHz b. *Memory* 4GB, HDD 500 GB	Digunakan dalam proses konfigurasi sistem
2.	Laptop (Server): a. Minimal *Processor* Intel *Core i5* b. *RAM* minimal 8 GB c. Harddisk minimal 500 GB	Digunakan sebagai tempat konfigurasi server

Tabel 6. Kebutuhan Perangkat Lunak (*Software*)

No.	Perangkat Lunak	Keterangan
1.	Sistem Operasi *Windows* 10 64 *Bit*	Sistem operasi yang digunakan untuk membangun sistem ini
2.	Sistem Operasi *Debian 9*/ ubuntu 16.04 atau 20.04	Sistem operasi yang digunakan untuk menjalankan *service* yang ada pada sistem dan sebagai server
4.	*Haproxy*	Sebagai perangkat lunak dalam mendistribusikan atau membagikan trafik ke beberapa server
5.	Virtualbox/Proxmox	Sebagai perangkat lunak untuk membuat virtualisasi
6.	OpenNebula	Digunakan untuk membuat cloud computing

B. Desain Sistem Server OpenNebula

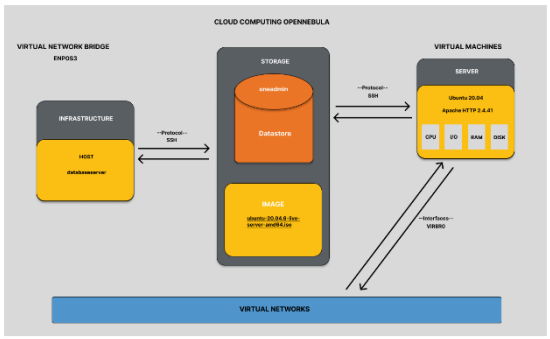
Beberapa Proses yang terjadi pada sistem OpenNebula dimana diakses beberapa user atau klien secara bersamaan dengan jumlah banyak menggunakan akses internet, untuk menuju ke server opennebula, terlebih dahulu yang harus dilewati adalah server *load balancing* dengan menggunakan apache HTTP 2.4.41 yang dimana diterapkan 2 algoritma yaitu *least bandwith* dan *least response time* kemudian akan membagi beban atau mendistribusikan trafik ke masing - masing server yang tersedia menggunakan *protocol* TCP/IP. Server di bagi atas 3 bagian agar supaya tidak memakan banyak beban seperti CPU, RAM dan lain - lain. Kemudian untuk databasenya terpisah agar supaya tidak terbebani di web server, untuk akses opennebula dengan menggunakan *protocol* HTTP di port 9869 dan *statistic* algoritma yang berjalan pada haproxy menggunakan port 8080. Untuk lebih jelasnya dapat dilihat pada Gambar 7



Gambar 7. Desain Sistem OpenNebula

C. Desain Sistem Server Cloud

Pada desain sistem *virtual machines* di cloud komputing opennebula terdapat *virtual networks* yang terhubung antara *guest* dan *host* dengan mempunyai 2 *interfaces* dengan *ip address* yang berbeda. Untuk *host* sebagai *infrastructure* yaitu databasenya dengan *storage* sebagai *datastore* penyimpanan *image* dan HDD dan lain sebagainya. Apache yang digunakan yaitu apache HTTP 2.4.41 dengan *image ubuntu-20.04.6-live-server-amd64.iso* untuk protokolnya menggunakan SSH. Untuk lebih jelasnya dapat dilihat pada Gambar 8.

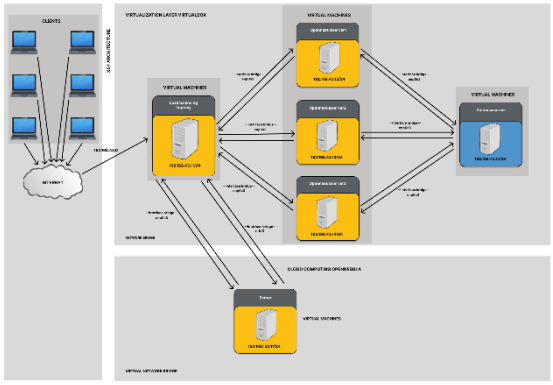


Gambar 8. Desain Sistem Server Cloud

D. Skema Jaringan

Skema jaringan menjelaskan bagaimana cara kerja yang dilakukan oleh load balancing haproxy pada opennebula berdasarkan dengan ip address yang digunakan. Pertama - tama banyak user yang mengakses internet kemudian setelah itu *download* dan install virtualbox sebagai virtualisasi dengan adanya koneksi internet yang memadai. Selanjutnya di dalam virtualbox buat 4 server dan di dalam *cloud computing* opennebula buat 1 server sebagai skenario. Adapun server yang dibuat yaitu load balancing haproxy, opennebulaserver1, opennebulaserver2,opennebulaserver3 dan databaseserver. Kemudian untuk di *cloud* buat 1 server dengan nama server 1. Di dalam virtualbox menggunakan *bridge adapter*, *Bridged adapter* ini memungkinkan OS *guest* untuk menerima data maupun mengirimkan data ke jaringan fisik. Jadi artinya OS guest dan OS *host* adalah dua komputer berbeda yang terhubung ke dalam jaringan yang sama. Bila OS *host* memiliki lebih dari satu *Ethernet* maka harus menyetting ke jaringan *virtual machine*/OS guest akan

disambungkan dan IP yang diberikan ke *virtual machine* harus dari subnet yang sama dengan jaringan yang dipakai oleh OS *host*. *Interfaces* yang digunakan yaitu enp0s3 Untuk lebih jelasnya dapat dilihat Gambar 9



Gambar 9. Skema Jaringan

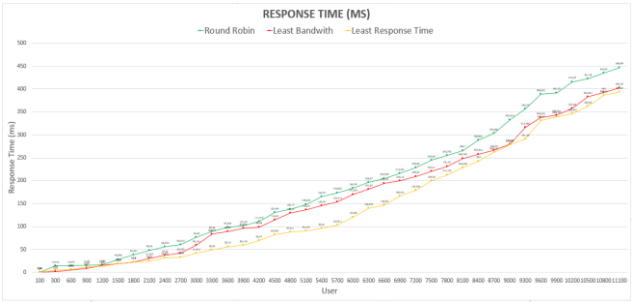
E. Pengujian

Pada tahap pengujian ini dilakukan agar bisa memastikan *load balancing haproxy* pada opennebula berjalan dengan baik sesuai dengan diharapkan. Ada beberapa yang perlu di uji baik tingkat *performance* dan metode *failover* dengan melihat parameter yaitu *Latency/Response Time*, *Error Rate/Packet Loss* dan *Troughput*.

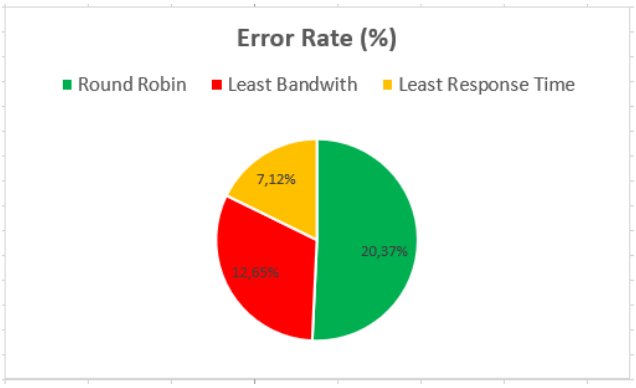
IV. HASIL DAN PEMBAHASAN

Hasil dari penelitian ini beruapa penerapan dan pengujian metode *load balancing haproxy* pada opennebula dimana menerapkan algoritma *Least Bandwith* dan *Least Response Time*. Kemudian adapun skala pembanding yaitu menggunakan algoritma *default* dari load balancing yaitu *Round Robin.* Adapun analisis penerapan dan pengujian *load balancing haproxy* pada opennebula adalah sebagai berikut.

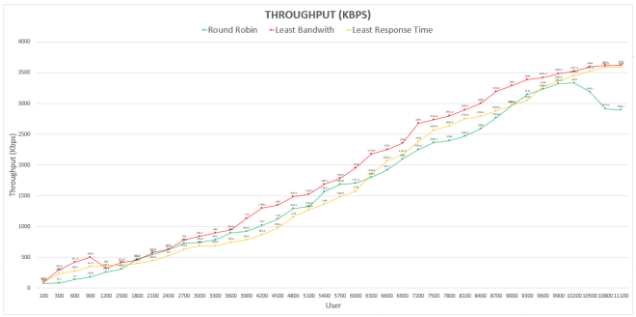
Adapun hasil *latency/response time*, *error rate/packet loss* dan *throughput* dapat dilihat pada Gambar 9 – 11.



Gambar 9. Grafik *Latency/Response Time*



Gambar 10. *Error Rate/Packet Loss*



Gambar 11. Grafik *Throughput*

Hasil parameter QoS menggunakan semua algoritma dan server *load balancing haproxy* opennebula dapat dilihat pada Tabel 7

Tabel 7. Hasil parameter QoS

Parameter QoS	Users	Algoritma	Nilai	Standar TIPHON	Indeks	Keterangan
Throughput (bps)	11100	*Round Robin*	2894,1 Kbps	100 bps	4	Sangat Bagus
		Least Bandwith	3618 Kbps	100 bps	4	Sangat Bagus
		Least Response Time	3589 Kbps	100 bps	4	Sangat Bagus
Packet Loss (%)	11100	*Round Robin*	20,37 %	15 %	2	Sedang
		Least Bandwith	12,65 %	3 %	3	Baik
		Least Response Time	7,12 %	3 %	3	Baik
Response Time (ms)	11100	*Round Robin*	446,89 ms	> 350 – 450 ms	2	Sedang
		Least Bandwith	402,45 ms	> 350 – 450 ms	2	Sedang
		Least Response Time	393,67 ms	> 350 – 450 ms	2	Sedang

Hasil parameter QoS untuk data pada server opennebula diperoleh nilai dengan masing-masing parameter yaitu user yang akses 11100 maka *throughput round robin* sebesar 2894,1 Kbps mendapatkan indeks 4 dengan kategori sangat bagus berdasarkan standar THIPON karena memenuhi nilai 100 bps kemudian algoritma *least bandwith* sebesar 3618 Kbps dengan kategori sangat bagus sedangkan *least response time throughputnya* 3589 Kbps berada pada indeks 4 sangat bagus. Selanjutnya untuk *packet loss* 11100 user diperoleh *round robin* sebanyak 20,37 % yang mendapatkan indeks 2 karena masuk pada nilai 15 % keatas dengan kategori sedang berdasarkan standar THIPON kemudian *least bandwith* sebesar 12,65 % mempunyai indek 3 yaitu baik sedangkan untuk *least response time packet loss* nya sebesar 7,12 % dengan mempunyai index 3 kategori baik. Selanjutnya untuk *response time* user 11100 dengan algoritma pembanding yaitu *round robin* sebesar 446,89 ms masuk pada standar > 350 – 450 ms dengan indeks 2 kategori sedang. Kemudian untuk algoritma *least bandwith* sebesar 402,45 ms berada pada standar > 350 – 450 ms dengan indeks 2 kategori sedang sedangkan algoritma *least response time* sebesar 393,67 ms berada pada standar TIPHON> 350 – 450 ms dengan indeks 2 kategori *response time* sedang.

V. KESIMPULAN

Berdasarkan hasil pengujian, maka diperoleh kesimpulan sebagai berikut.

1. *Load Balancing Haproxy* berhasil membagi beban secara merata ke masing-masing server opennebula sehingga mendapatkan kinerja server yang lebih baik.
2. Hasil pengukuran *load balancing* dengan *request* user 11100 dengan standar berdasarkan *Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON)* dimana *response time* untuk algoritma *round robin* sebesar 446,89 ms masuk kategori sedang, algoritma *least bandwith* sebesar 402,45 ms kategori sedang dan *least response time* sebesar 393,67 ms masuk dalam kategori baik. Kemudian *packet loss/Error rate* (%) untuk algoritma *round robin* 20,37 %, *least bandwith* 12,65 % dan *least response time* 7,12 % masuk kategori baik berdasarkan standar. Sedangkan *throughput round robin* 2894,1 Kbps, *least bandwith* 3618 Kbps dan *least response time* 3589 Kbps masuk dalam indeks 4 kategori sangat baik.

UCAPAN TERIMA KASIH

Terima kasih kepada Tuhan Yang Maha Esa, orang tua, teman, kedua dosen pembimbing, serta seluruh dosen dan staf Teknik Elektro Program Studi Teknik Komputer dan Jaringan Politeknik Negeri Ujung Pandang.

REFERENSI

[1] D. Leman, "LOAD BALANCING 2 JALUR INTERNET MENGGUNAKAN MIKROTIK ROUND ROBIN," *RJOCS (Riau J. Comput. Sci.*, vol. 5, no. 2, pp. 137–143, 2019.

[2] Y. Khair, A. Dennai, and Y. Elmir, "An Experimental Performance Evaluation of OpenNebula and Eucalyptus Cloud Platform Solutions," in *International Conference on Artificial Intelligence in Renewable Energetic Systems*, 2021, pp. 450–457.

[3] D. Sudirman, "OpenNebula VS OpenStack, 2 Layanan Cloud Pilihan," *www.virtualiable.com*, 2020. https://virtualiable.com/opennebula-vs-openstack/#:~:text=OpenNebula telah dibentuk selama hampir satu dekade yang,mana mereka dapat menyajikan fitur pembaharuan untuk pengguna. (accessed Jul. 07, 2022).

[4] S. D. Riskiono and D. Pasha, "Analisis Metode Load Balancing Dalam Meningkatkan Kinerja Website E-Learning," *J. TeknoInfo*, vol. 14, no. 1, pp. 22–26, 2020.

[5] B. Alankar, G. Sharma, H. Kaur, R. Valverde, and V. Chang, "Experimental setup for investigating the efficient load balancing algorithms on virtual cloud," *Sensors*, vol. 20, no. 24, p. 7342, 2020.

[6] N. Thapliyal and P. Dimri, "Load Balancing in Cloud Computing Based on Honey Bee Foraging Behavior and Load Balance Min-Min Scheduling Algorithm".

[7] B. K. Madakatte and H. R. Nagesh, "Arr And Fusion Based Virtual Machine Scheduling Techniques For Eucalyptus Cloud," *Webology (ISSN: 1735-188X)*, vol. 18, no. 6, 2021.

[8] I. M. Ibrahim *et al.*, "Web server performance improvement using dynamic load balancing techniques: A review," *system*, vol. 19, p. 21, 2021.

[9] S. Majumder, "Least response time method," *Citrix.com*, 2021. https://docs.citrix.com/en-us/citrix-adc/current-release/load-balancing/load-balancing-customizing-algorithms/leastresponsetime-method.html (accessed Apr. 28, 2022).

[10] S. Majumder, "Least bandwidth method," *Citrix.com*, 2021. https://docs.citrix.com/en-us/citrix-adc/current-release/load-balancing/load-balancing-customizing-algorithms/leastbandwidth-method.html (accessed Apr. 28, 2022).

[11] A. Widarma and Y. H. Siregar, "Analisis Kinerja Teknologi Virtualisasi Server (Study Kasus: Universitas Asahan)," in *Seminar Nasional Multi Disiplin Ilmu Universitas Asahan*, 2019.

[12] S. Syamsu, "Implementasi Cluster Database Berbasis MySQL Dan Haproxy Sebagai Pembagi Beban Kerja Server," *Inspir. J. Teknol. Inf. dan Komun.*, vol. 8, no. 1, pp. 48–58, 2018.

[13] D. Hartawan, "Bagaimana Cara Kerja Database Server? Cari Tahu Semuanya di Sini," *toffeedev.com*,2021. https://toffeedev.com/blog/cara-kerja-database-server/ (accessed Aug. 15, 2022).

[14] N. Hikmah, A. Zaini, and H. Santoso, "Analisis

Efektifitas Quality Of Service Pada Jaringan Kabel di Lingkungan SMK PGRI Turen," *RAINSTEK J. Terap. Sains Teknol.*, vol. 5, no. 1, pp. 84–94, 2023.

[15] R. TIPHON, "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) General aspects of Quality of Service (QoS)," DTR/TIPHON-05006 (cb0010cs. PDF), 1999.

[16] N. Maulana, O. D. Nurhayati, and E. D. Widianto, "Perancangan sistem sensor pemonitor lingkungan berbasis jaringan sensor nirkabel," *J. Teknol. dan Sist. Komput.*, vol. 4, no. 2, pp. 353–360, 2016.

[17] K. A. Prasetia, S. R. Akbar, and R. Primananda, "Implementasi Lingkungan Test pada Moodle dengan Apache JMeter," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, p. 964X.