IMPLEMENTASI SISTEM *LOAD BALANCING* MENGGUNAKAN METODE *LEAST TIME FIRST BYTE* DAN *MULTI-AGENT SYSTEM* PADA LINGKUNGAN VIRTUAL

Kurnia Siwi Kinasih [1)], Dyah Ayu Wiranti [2)], Shinta Rizki Firdina Sugiono [3)], Khadijah Fahmi Hayati Holle [4)], Agung Teguh Wibowo Almais [5)]
[1] Sains dan Teknologi, UIN Maulana Malik Ibrahim Malang
email: kurniasiwi1@gmail.com
[2] Sains dan Teknologi, UIN Maulana Malik Ibrahim Malang
email: adyah178@gmail.com
[3] Sains dan Teknologi, UIN Maulana Malik Ibrahim Malang
email: shintarizki49@gmail.com
[4] Sains dan Teknologi, UIN Maulana Malik Ibrahim Malang
email: khadijah.holle@uin-malang.ac.id
[5] Sains dan Teknologi, UIN Maulana Malik Ibrahim Malang
email: agung.twa@gmail.com

Abstact

In this modern era, the technology is growing rapidly, the Internet is misled. This condition will be related to the service provider or commonly referred to as a server. Increasing the number of clients, the server also has to work heavier so that it often occurs overload. The Load balancing mechanism uses the Least Time First Byte and Multi-Agent system methods. This mechanism allows the server to overcome the number of users who perform service requests so that the load from the server can be resolved. This solution is considered efficient and effective because the request process on the information system will be shared evenly on multiple server back ends. The results of the research that can be proved if using this mechanism the server can work well when the request is from a user or client dating, this method successfully distributes the balancer evenly through the server backend. So the server is no longer experiencing overload. This can be proved when a system that has used the Load balancing method with 300 connections generates a throughput of 123.1 KB/s as well as response time value of 4.72 MS and a system that does not use the Load balancing method has a throughput of 108.4 KB/s as well as a response time value of 120.3 Ms. Therefore by implementing Load balancing the performance of the system can always be improved.

Keywords: *server, load balancing, least time first byte, multi-agent system*

1. PENDAHULUAN

Pada zaman modern ini, *server* merupakan hal yang begitu penting dan sangat diperlukan. Dikarenakan *server* yang kini menjadi pusat pelayanan dari semua pengguna. *Server* sendiri dapat diartikan sebagai suatu perangkat *computer* atau program yang menyediakan layanan kepada komputer lain serta penggunaanya yang biasa disebut dengan *client*. Dalam pusat data, juga ada yang dinamakan *server* yang dapat dirtikan suatu *computer* fisik yang menjalankan sebuah program. Di suatu model pemrograman *server* atau client, program *server* memenuhi serta menunggu permintaan dari program client yang mungkin berjalan di *computer* yang sama atau yang lain. Aplikasi dalam *computer* dapat berfungsi sebagai *client* dengan permintaan layanan dari program lain dan dapat juga sebagai *server* permintaan dari program lain.

Spesifikasi sumber daya pada sebuah *system computer* yang dimiliki *server* sangat tinggi jika dibandingkan degan kompter lainnya. Pengolahan data yang diolah *server* pun sangat besar dan kompleks. Jadi semakin banyak pengguna yang menggunakan, maka semakin kompleks pula infrastruktur *server* yang harus dibangun. Semua request yang datang dari seluruh pengguna harus bisa ditampung oleh infrastruktur yang dibangun. *Load server* akan semakin tinggi sesuai dengan seberapa banyak pengguna layanan yang telah disediakan oleh *server*. Dapat dipastikan semakin tinggi *load server* dapat berdampak pada berhenti bekerjanya *server*, dikarenakan beban yang semakin tinggi sehingga berakibat pula pada penurunan kecepatan proses yang berjalan [1]

Berdasarkan masalah yang terjadi tesebut, maka sangat dibutuhkan teknologi yang dapat

mengatasinya. Salah satunya adalah system terdistribusi. Dimana system terdistribusi akan mendistribusikan request yang masuk dari sejumlah pengguna kepada server-server agar diproses. Namun, selain system terdistribusi dikenal pula sebuah mekanisme *Load balancing* yang menurut beberapa ilmuwan dapat menangani masalah tersebut secara andal. Dengan menggunakan tambahan mekanisme *Load balancing* diharapkan mampu mengatasi banyaknya pengguna dengan skala yang cukup besar sehingga reliabilitas sebuah layanan dapat ditingkatkan.

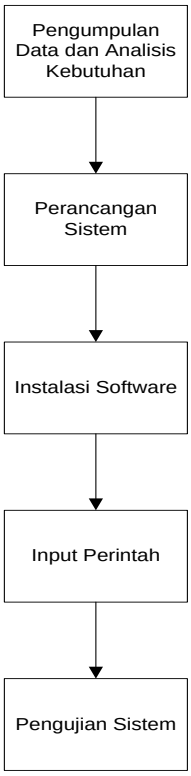
Load balancing merupakan metodologi yang digunakan untuk mendistribusikan atau membagi beban kerja pada beberapa *computer* untuk mencapai serta memaksimalkan throughput, menghindari beban yang berlebih, dan meminimalkan waktu respon. Dalam proses skala yang cukup besar, *Load balancing* ini sangat membantu. *Load balancing* akan mendistribusikan resources permintaan pengguna pada *server*, sehingga diharapkan tidak akan terjadi overload.

System *Load balancing* dapat diaplikasikan dengan menggunakan metode Multi Agent Sistem. Multi Agent System merupakan salah satu ilmu *artificial intelligence* (AI). Dengan menerapkan metode Multi Agent System akan berdampak pada efisiennya sebuah sistem. Metode ini menggunkaan agent-agent yang memiliki kemampuan khusus untuk terus-menerus melakukan tugas yang sebelumnya diberikan oleh pengguna dalam suatu lingkungan yang khusus. Agent dalam sistem ini dapat berpindah dari satu node menuju node yang lainnya secara bebas sesuai dengan tugas yang telah diberikan [2].

Dalam jurnal ini penulis mengimplementasikan mekanisme *Load balancing* dengan menggunakan algoritma *Least Time First Byte dan Multi Agent System*. Di mana agent akan bertugas memantau keadaan *resource backend server*. Di sini agent akan berkomunikasi dengan node *server* untuk memperoleh informasi resources. Dengan menerapkan metode ini, permintaan akan dilakukan lebih cepat dalam kata lain hanya membutuhkan waktu yang sebentar. Dengan demikian, metode ini menerapan satu *server* pusat yang tugasnya memantau tugas agent dan membagi atau mengatur beban yang masuk ke dalam *server* agar tetap *idle*. *Server* mampu melakukan mekanisme ini dengan efisien ke backend *server* yang telah tersedia. Tidak akan ada lagi kemacetan aliran data dan waktu respon akan menjadi cepat [3].

2. METODE PENELITIAN

Metode yang kami gunakan dalam penelitian ini adalah metode *experimental*. Untuk memperkuat hasil penelitian ini langkah yang harus dilakukan adalah mengumpulakan berbagai *literature* yang berhubungan dengan metode *Load balancing* dengan menggunakan least time fisrt byte dan multi agent system serta aspek lainnya sebagai dukungan [4]. Dalam perancangannya, terdapat beberapa tahapan yang digunakan yaitu sebagai berikut:



Gambar 1. Alur Perancangan

Keterangan gambar 1:

A. Pengumpulan Data dan Analisis Kebutuhan

Dalam penelitian ini teknik pengumpulan data yang kami gunakan adalah studi literature dan observasi (mengamati virtual website). Studi literature dimaksudkan untuk mencari referensi teori yang sesuai dengan penelitian yang dilakukan. Sedangkan observasi di sini dmaksudkan dengan melakukan pengamatan pada object yang kami bahas yaitu lingkungan vitual (*virtual website*).

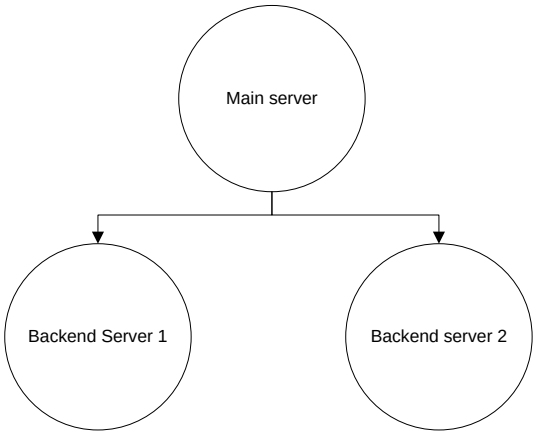
Pada penelitian ini peneliti menggunakan metode *Load balancing* yang telah menggunakan agent. *Load balancing server*

merupakan metode agent yang telah diimplementasi secara dinamis menggunakan agent. Dimana agent digunakan untuk mengumpulkan informasi yang dibutuhkan dalam program yang telah dituliskan di dalamnya. *Load balancing* secara dinamis digunakan untuk meningkatkan kinerja pada sistem.

Load balancing digunakan untuk meningkatkan sumber daya dengan memanfaatkan penggunaan paralelisme, memperpendek waktu respons, dan meningkatkan *throughput*. Selain itu *Load balancing* digunakan untuk membagi beban pada sistem ke node secara optimal.

B. Perancangan Sistem

Berdasarkan permasalahan yang ada, peneliti menggunakan lebih dari satu komputer/laptop untuk melakukan pengecekan berjalannya sistem *Load balancing* yang telah diterapkan. Peneliti menggunakan satu main *server* dan dua backend *server* dengan masing-masing memiliki kapasitas 50 GB untuk main *server* dan 10 GB untuk backend *server*.



Gambar 2. Perancangan Sistem

Keterangan gambar 2:

Pada gambar 2 yaitu perancangan sistem menjelaskan bahwa dalam sistem *Load balancing* ini membagi beban *server* secara merata dari main *server* ke backend-backend *server*. Yang kemudian akan dikirimkan agent pada setiap *backend server* yang membawa informasi yang kemudian dikembalikan lagi untuk diputuskan di main *server*.

Selain menggunakan system *Load balancing* peneliti juga menggunakan multi agent system. Aplikasi yang digunakan untuk mengkonfigurasi multi agent system yaitu dengan menggunakan JADE yang berbasis JAVA. Terdapat dua kelompok fungsi yang digunakan pada multi agent system. Fungsi yang pertama pada multi agent system yaitu agent yang berfungsi meminta informasi resources dari *server* backend, yang kedua adalah agent yang digunakan untuk memberikan informasi resources dari *server* backend. Arsitektur yang digunakan multi agent system menggunakan enam agent. Dimana agent 1 bertujuan untuk melakukan request informasi resources yang berada pada *server Load balancing*.

Tabel 1. Konfigurasi Dan Spesifikasi Mesin Virtual

NO	CPU	RAM	Storage	Keterangan
1.	1 core	1 GB	10 GB	*Load balancing* dan Agent *Server*
2.	1 core	1 GB	10 GB	Backend *Server*
3.	1 core	1 GB	10 GB	Backend *Server*
4.	1 core	1 GB	10 GB	Backend *Server*
5.	2 core	2 GB	50 GB	Main *Server*

Keterangan Tabel 1 :

Pada tabel 1, dijelaskan bahwa konfigurasi dan spesifikasi mesin virtual menggunakan lima ubuntu *server*. Terdapat empat *server* ubuntu yang mengguanakan CPU 1 core, RAM 1 GB, storage 10 GB. Sedangkan satu *server* ubuntu menggunakan CPU 2 core, RAM 2 GB, storage 50 GB, dimana satu *server* tersebut berperan sebagai Main *Server*.

C. Installasi Software

Software yang dibutuhkan dalam penelitian ini adalah :

1) Virtual Box

Oracle VM VirtualBox merupakan suatu perangkat lunak virtualisasi yang dapat kita gunakan untuk dapat megoperasikan sistem operasi "tambahan" yang telah berada di dalam sistem operasi "utama". Sebagai contoh dalam permasalahan ini anda telah memiliki sistem operasi Windows yang telah tepasang di *computer*, maka anda dapat pula menjalankan sistem operasi lain yang anda inginkan dalam sistem operasi Windows. Hal ini sangat penting jika anda menginginkan untuk melakukan pengujian installasi suatu sistem tanpa harus kehilangan sistem yang telah ada.

2) Apache

Apache adalah suatu program yang digunakan untuk melayani, ataupun menjalankan serta memfungsikan suatu situs web dalam sebuah komputer. (Emanuel, 2006) [5]. Kita dapat menjalankan Apache di banyak SO (Sistem Operasi) seperti contohnya BSD, Linux, Microsoft Windows dan Novell Netware serta platform lainnya. Dikarenakan perangkat ini merupakan perangkat lunak sumber terbuka yang dikembangkan oleh komunitas terbuka yang terdiri dari pengembang di bawah naungan Apache Software Foundation. Dalam Apache kita dapat menemukan fitur-fitur canggih seperti pesan kesalahan yang dapat dikonfigurasi, autentikasi berbasis basis data dan banyak hal lainnya. Selain itu apache dapat digunakan sebagai load balancer dengan cara menambahkan beberapa modul modul konfigurasi tertentu di dalamnya. Pada penelitian ini apache akan di gunakan sebagai load balancer adalah apache versi 2.4 karena apache pada versi 2.4 sudah mendukung modul untuk *Load balancing* serta space memory yang digunakan lebih sedikit daripada versi sebelumnya.

3) Java Agent Development Framework

Java Agent Development Framework atau yang biasa disebut JADE merupakan suatu perangkat lunak yang diimplementasikan ke dalam bahasa java. JADE menyederhanakan perangkat multi-agent system melalui middle-ware yang telah sesuai berdasarkan spesifikasi FIPA dan melalui alat grafis yang mendukung. Sebuah sistem berbasis JADE dapat didistribusikan di seluruh mesin meskipun dengan OS yang tidak sama dan konfigurasi dapat dikontrol melalui GUI remote. Konfigurasi bahkan dapat diubah pada saat run-time dengan memindahkan agen dari satu mesin yang lain dan apabila diperlukan. Java Agent Development Framework ini sepenuhnya menggunakan bahasa java.

4) Oracle

Oracle merupakan *Relational Database Management System* (RDBMS) yang digunakan untuk mengelola suatu informasi terbuka serta secara terintegrasi. Kemampuan oracle yang mampu menyediakan solusi yang efektif dan efisien di dunia IT. Oracle mempunyai 3 makna yaitu oracle database *server*, oracle sebagai platform, dan oracle corporation. Oracle mampu menangani data yang cukup besar, namun meskipun data yang diolah memiliki ukuran yang cukup besar namun oracle mampu mengolahnya dengan cepat dengan data yang akurat sesuai permintaan user. Tidak hanya itu oracle masih memiliki kelebihan-kelebihan yang lain yaitu diantaranya mampu bekerja di lingkungan client/*server*, mampu memanagement user dimana setiap user dapat diatur hak aksesnya pada suatu database oleh administrator, dan masih banyak lagi.

D. Input Perintah

Tahap ini diperlukan untuk menjalankan Multi Agent System dengan mengeksekusi query menggunakan tools Netbeans IDE. Query ini diperlukan agar tiap backend *server* memiliki agent yang nantinya akan membantu main *server* meringankan beban [6].

```
import java.util.List;
import genius.core.AgentID;
import genius.core.Bid;
import genius.core.actions.Accept;
import genius.core.actions.Action;
import genius.core.actions.Offer;
import
genius.core.parties.AbstractNegotiationParty;
import
genius.core.parties.NegotiationInfo;

public class ExampleAgent extends
AbstractNegotiationParty {
    private final String description =
"Example Agent";
    private Bid lastReceivedOffer;
    private Bid myLastOffer;

    @Override
    public void init(NegotiationInfo info) {
      super.init(info);
    }

    @Override
    public Action chooseAction(List<Class<?
extends Action>> list) {
      double time =
getTimeLine().getTime();
      if (time < 0.5) {
        return new Offer(this.getPartyId(),
this.getMaxUtilityBid());
      } else {
        if (lastReceivedOffer != null
          && myLastOffer != null
          &&
this.utilitySpace.getUtility(lastReceivedOffer)
> this.utilitySpace.getUtility(myLastOffer)) {

          return new
Accept(this.getPartyId(), lastReceivedOffer);
        } else {
          myLastOffer =
generateRandomBid();
          return new
Offer(this.getPartyId(), myLastOffer);
        }
      }
    }

    @Override
    public void receiveMessage(AgentID
sender, Action act) {
      super.receiveMessage(sender, act);

      if (act instanceof Offer) {
        Offer offer = (Offer) act;
        lastReceivedOffer = offer.getBid();
      }
    }

    @Override
    public String getDescription() {
      return description;
    }

    private Bid getMaxUtilityBid() {
      try {
        return
this.utilitySpace.getMaxUtilityBid();
      } catch (Exception e) {
        e.printStackTrace();
      }
      return null;
    }
  }
```

E. Pengujian Sistem

Pengujian pada penelitian ini difokuskan pada aspek efisiensi dari penerapan *Load balancing* menggunakan multi agent system. Proses pengujian efisiensi dilakukan dengan membandingkan antara *Load balancing* tanpa agent dan *Load balancing* menggunakan multi agent system. Efisiensi *Load balancing* dinilai dari error yang ditimbulkan, request yang terlayani dan throughput yang dihasilkan. Dalam pengujian system ini peneliti juga meggunakan metode test. Metode test adalah cara untuk pengujian yang dilakukan dengan parameter-parameter system. Untuk mendapatkan informasi secara rinci maka penguji membutuhkan metode test. Di mana akan memperoleh hasil akhir yang berupa rata-rata waktu respons, *throughput*, jumlah error, dan nilai dari system *Load balancing*. Untuk melakukan perhitungan *Load balancing* diperoleh rumus sebagai berikut.

$$load = \frac{\%CPU,\%memory}{N_{request}} \quad \text{..... (1)}$$

Pada rumus (1) di atas merupakan persamaan berdasarkan persentase beban dari CPU (%CPU) dikalikan dengan persentase beban memori (%memory). Di mana nilai dari perhitungan tersebut akan

dibagi oleh jumlah request yang masuk. Rumus (2) merupakan cara menghitung nilai rata-rata waktu respons.

$$Average = \frac{1}{N_{request}} \sum_{i=1}^{n} t_i \quad \text{.....} \quad (2)$$

Keterangan Pada rumus (2) di atas dimana variabel n merupakan banyak request yang diuji. Variabel t yaitu nilai dari waktu respons setiap request. Persamaan lainnya dapat diperoleh dengan menggunakan variabel waktu dan request dari pengguna. Rumus berikutnya merupakan cara menghitung median, varian, dan standar deviansi.

$$Median = \frac{1}{2}\left(t\frac{n_{request}}{2} + t\left(\frac{n_{request}}{2} + 1\right)\right) \ldots (3)$$

Pada rumus (3) telah diketahui cara menghitung median dengan nilai waktu respons (t) dengan banyak request (n) akan dibagi dua kemudian ditambahkan dengan nilai waktu respons setelahnya.

$$Variance = \frac{1}{n}\sum_{i=1}^{n}(t-t)^2 \quad \text{.....} \quad (4)$$

Pada rumus (4) adalah rumus cara menghitung variance. Dimana jumlah kuadrat dari selisih nilai waktu respons (t) dari nilai rata-ratanya, kemudian dibagi dengan jumlah nilai request (n). Nilai variance sendiri digunakan untuk menghitung seberapa jauh persebaran nilai waktu respons terhadap rata-rata.

$$Std.Deviation = \sqrt{variance} \quad \text{.....} \quad (5)$$

Pada rumus (5) adalah rumus cara menghitung standar deviasi. Nilai standar deviasi diperoleh dari akar nilai variasi data sebelumnya.

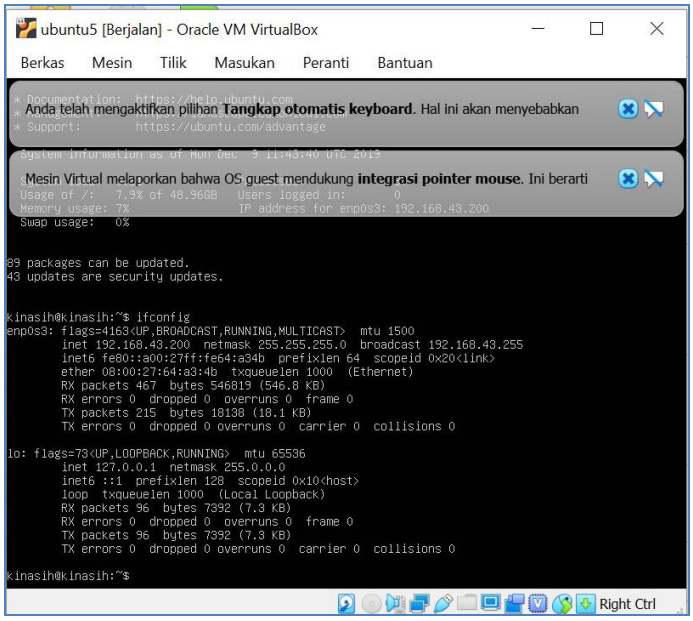
$$Error = \frac{n_{error}}{n_{request}} \times 100\% \quad \text{.....} \quad (6)$$

Pada rumus (6) adalah rumus cara menghitung error. Di mana nilai error diperoleh dari pengujian yang telah dilakukan oleh penguji. Nilai error merupakan request yang tidak diproses oleh *server* karena melebihi kapasitas waktu yang telah ditentukan.

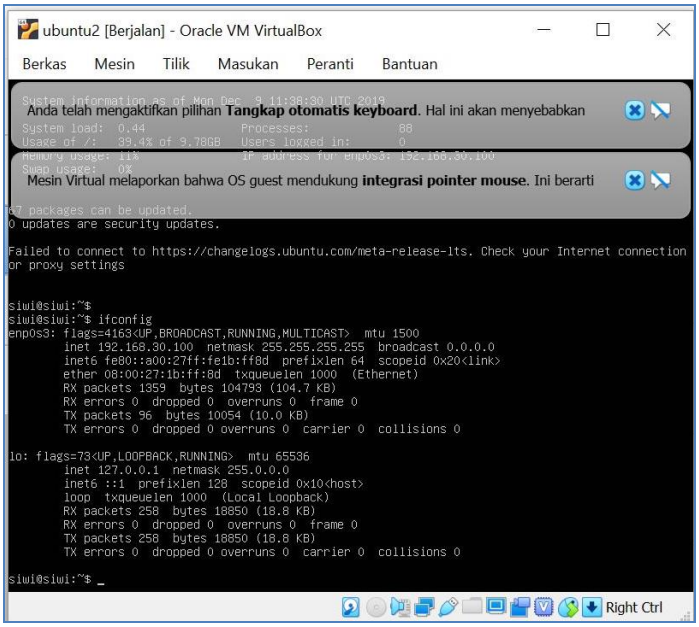
3. HASIL DAN PEMBAHASAN

Hasil pengujian dan analisis pengujian dengan menggunakan metode *Load balancing* yang telah dikolaborasikan dengan Multi Agent Sistem (MAS). Metode MAS diuji kinerjanya dengan memproses data. Analisis pengujian dibagi menjadi dua yaitu pengujian metode *Load balancing* dengan menggunakan Multi Agent Sistem dan pengujian metode *Load balancing* tanpa menggunakan Multi Agent Sistem.

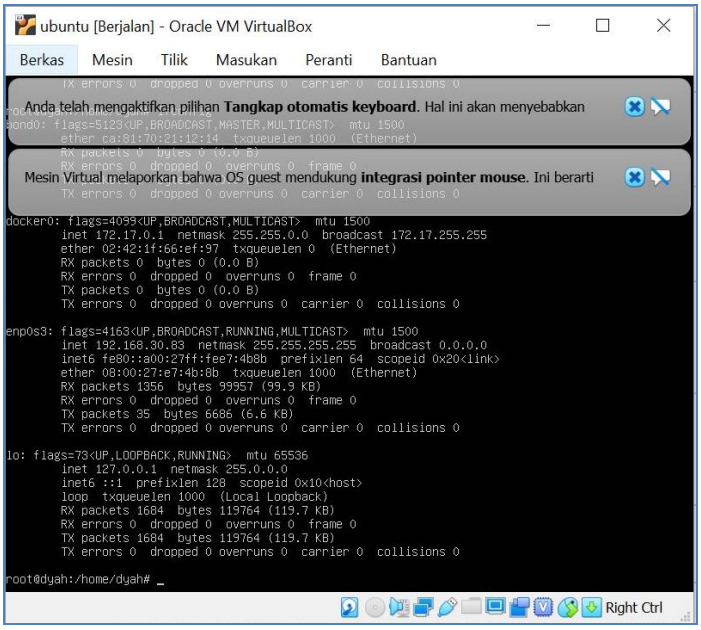
Di mana hasil analisis pengujian didapatkan saat melakukan pengujian dengan sistem *Load balancing*. Data pengujian sistem ini dipaparkan berupa data tabel dan bentuk grafik. Peneliti melakukan pengujian sistem dengan menggunakan 450 request ke dalam sistem *Load balancing* selama 10 secound, hal ini diharapkan dapat memberikan gambaran kinerja dari setiap metode yang telah diuji.



Gambar 3. Hasil *Load balancing* 1



Gambar 4. Hasil *Load balancing* 2



Gambar 5. Hasil *Load balancing* 3

Keterangan Gambar 3 :

Pada gambar 3, ketika *Load balancing* pada *server* berjalan maka *server* akan menampilkan hasil berupa jumlah error yang ditimbulkan, collision, dan juga ip yang dapat akses di browser guna pengecekan kesuksesan jalannya *server*. Error dan Collision digunakan dalam perhitungan untuk membandingkan kedua metode. Hasil Perhitungan ditampilkan pada tabel 2.

Tabel 2. Data Pengujian Metode Lfb Tanpa Agent Dan Lfb-Mas Dengan 450 Request

Metode	Request	Average	Std. Dev. (ms)	Error %	Throughput (request/secound)
Tanpa Agent	450	3.172	1.626,22	6.1	11,85
MAS	450	4.047	2.292,64	0,00	10,75

Keterangan Tabel 2 :

Dari hasil perngujian pada tabel 2 dapat diketahui bahwa dengan sistem *Load balancing* dapat ditingkatkan kinerjanya dengan menerapkan metode LFB-MAS dengan *Multi-Agent Sistem* dapat dijadikan referensi *load balance* dalam memproses dengan skala besar yang efisien dan mengurangi risiko *overload*.

4. KESIMPULAN

Dari hasil analisis dan pengujian yang telah dijelaskan sebelumnya dapat diambil keputusan bahwa sistem rancang bangun LFB dengan menerapkan metode *Load balancing* dan MAS (*Multi Agent Sistem*) telah berhasil membagi beban *server* secara merata. Dan menurut penelitian ini dengan menggunakan metode *Load balancing* menggunakan *multi agent sistem* memiliki hasil yang lebih baik jika dibandingkan menggunakan metode *Load balancing* yang tanpa menggunakan multi agent sistem. Dimana dengan menggunakan *Load balancing* dan multi agent sistem akan memperoleh tingkat error 0.00% dan dengan waktu yang lebih cepat.

Berdasarkan penelitian ini diharapkan dapat dijadikan sebagai acuan penelitian selanjutnya. Dan sistem implementasi *Load balancing* menggunakan *multi agent sistem* pada lingkungan virtual ini dapat diterapkan pada sistem SIAKAD maupun sistem lainnya..

5. REFERENSI

[1] A. Rahmatulloh and F. MSN, "Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi," *J. Nas. Teknol. dan Sist. Inf.*, vol. 3, no. 2, pp. 241–248, 2017.

[2] M. F. Afriansyah, M. Somantri, and M. A. Riyadi, "Sistem Load Balancing Menggunakan Least Time First Byte dan Multi Agent System," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 6, no. 3, 2017.

[3] S. D. Riskiono, "Implementasi Metode Load Balancing Dalam Mendukung Sistem Kluster Server," *SEMNAS RISTEK*, pp. 456–457, 2018.

[4] I. P. A. P. Wibawa, I. D. Giriantari, and M. Sudarma, "Komputasi Paralel Menggunakan Model Message Passing Pada SIM RS (Sistem Informasi Manajemen Rumah Sakit)," *Maj. Ilm. Teknol. Elektro*, vol. 17, no. 3, p. 439, 2018.

[5] A. S. Foundation, "Apache Http Server," 2016. [Online]. Available: http://httpd.apache.org/docs/2.4/mod/mod_pr oxy.html. [Accessed: 30-Nov-2019].

[6] N. O. Source, "NetBeans Http Server," 2019. [Online]. Available: https://netbeans.org/index_id.html. [Accessed: 30-Nov-2019].