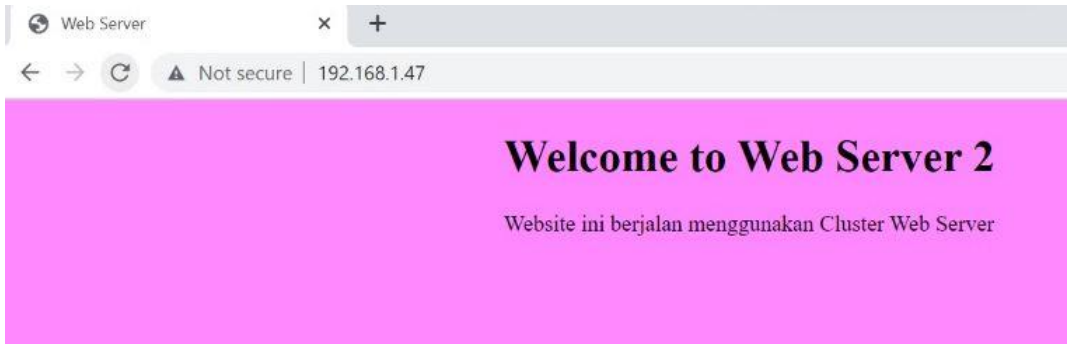
BAB 4
HASIL DAN PEMBAHASAN

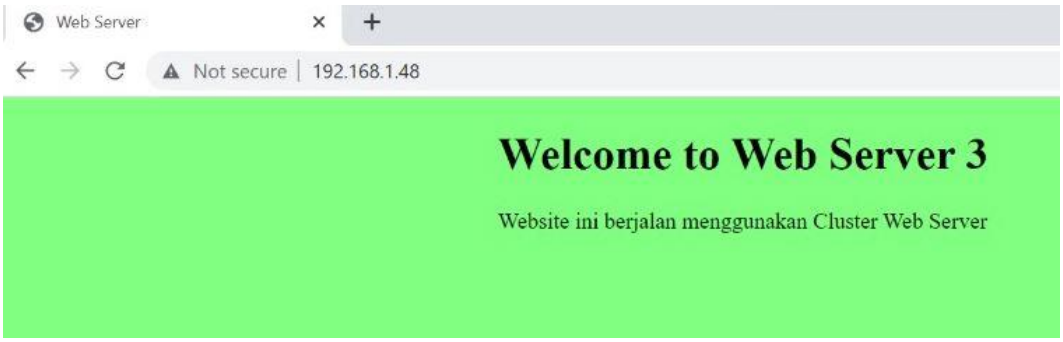
4.1 Pengujian *Web Server*

Setiap *web server* akan memiliki keluaran yang berbeda dan dapat diperhatikan pada Gambar 13 sampai Gambar 15.

Gambar 13. *Web server* pertama berwarna biru



Gambar 14. *Web server* kedua berwarna pink

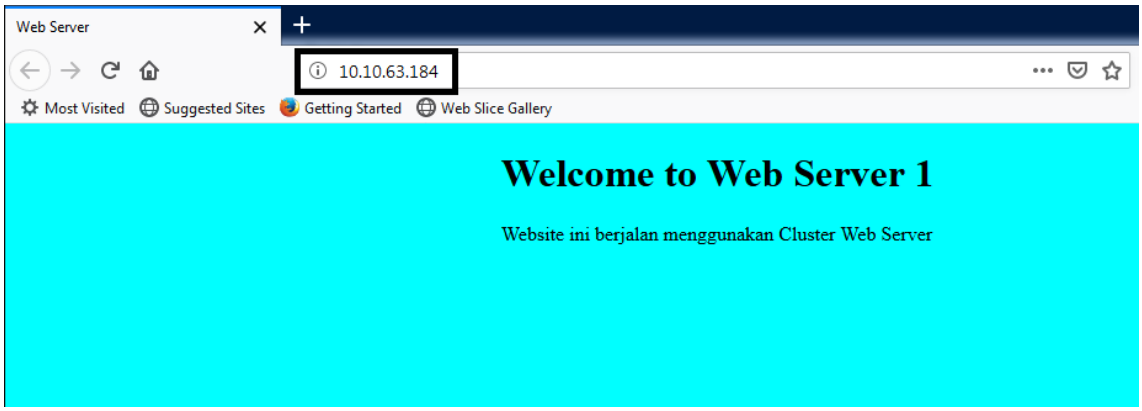


Gambar 15. *Web server* ketiga berwarna hijau

Berdasarkan Gambar 13 sampai Gambar 15, menjelaskan jika setiap *web server* akan memiliki perbedaan pada setiap keluaran. Dan ketika *load balancing* dijalankan akan terdapat variasi untuk membedakan setiap *web server*.

4.2 Hasil Pengujian *Load Balancing*

Pengujian dengan menjalankan Haproxy untuk mengetahui apakah *load balancing* dapat berjalan dengan baik atau tidak. Dengan skenario yang dimulai dari pengguna mengakses sebuah *web server* dengan menggunakan alamat dari *load balancer*. Kemudian *load balancing* akan memproses setiap permintaan pengguna untuk dibagikan ke setiap *web server*. Hasil dari pengujian Haproxy dapat diperhatikan pada Gambar 16.

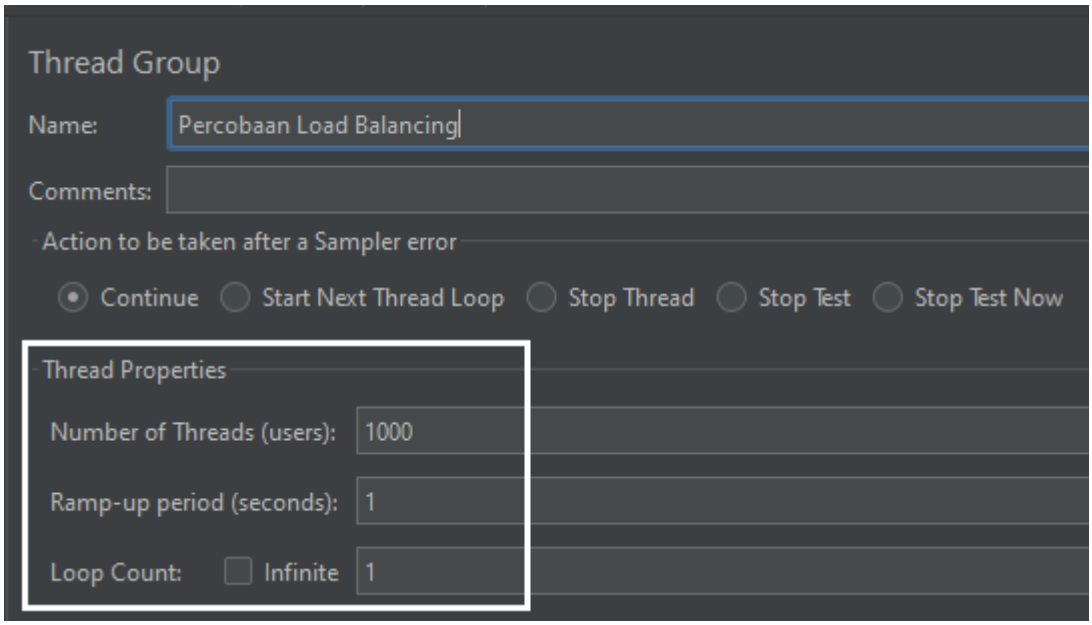


Gambar 16. Hasil pengujian *load balancing*

Load balancing berhasil dijalankan berdasarkan Gambar 16 yang berjalan menggunakan alamat *load balancer* pada ethernet 1. Peran *load balancing* adalah menerima setiap permintaan pengguna dan permintaan tersebut akan dibagikan ke setiap *web server* secara bergantian menggunakan algoritma *round robin*.

4.3 Hasil Pengujian *Throughput Load Balancing*

Pengujian dilakukan untuk mengetahui nilai *throughput* yang dapat diterima *load balancing*. *Throughput* adalah jumlah paket data yang dapat dikirim dalam satu waktu berdasarkan data yang dihasilkan dengan menggunakan JMeter. Konfigurasi JMeter dapat diperhatikan pada Gambar 17.

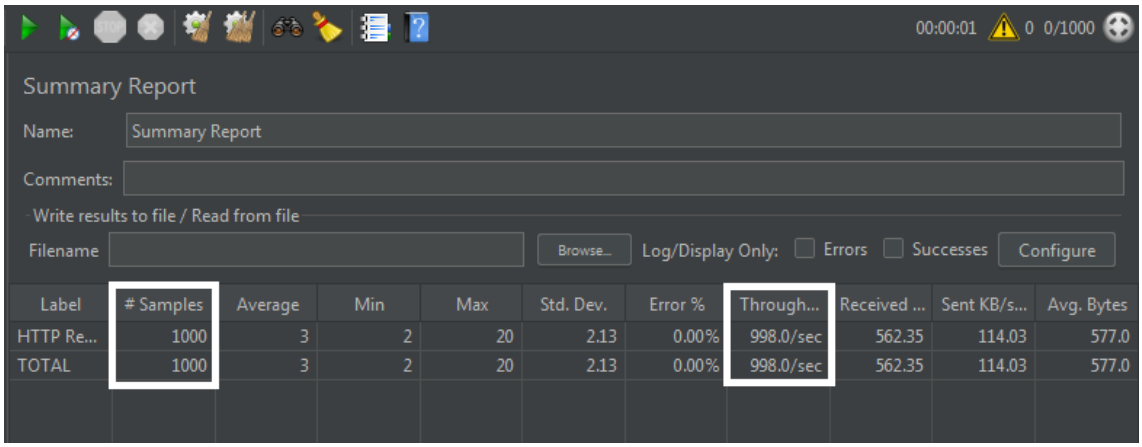


Gambar 17. Konfigurasi JMeter

Pada Gambar 17 terdapat tanda berwarna putih yang menjelaskan konfigurasi untuk banyaknya pengguna yang akan mengakses *web server* dalam waktu yang bersamaan. Konfigurasi pada Gambar 17 memperlihatkan jika ada 1000 pengguna yang akan mengakses *web server* dan dalam 1 detik akan diklik secara bersamaan dengan perulangan mengakses sebanyak 1 kali.

4.3.1 Pengujian *Load Balancing* pada 3 *Web Server*

Skenario pengujian pertama adalah pengujian yang dilakukan dengan semua *web server* dalam kondisi aktif. Pengujian akan mengambil nilai *throughput* yang dapat dikirim dalam satu waktu dengan jumlah pengguna sebanyak 1000. Hasil pengujian skenario pertama dapat diperhatikan pada Gambar 18.



Gambar 18. *Throughput* 3 *web server*

Hasil pengujian pada Gambar 18 memperlihatkan *load balancing* dalam menerima 1000 pengguna dapat mengirim jumlah paket yang besar dalam satu waktu. Pada kotak putih bagian kanan adalah nilai *throughput* yang dapat dihasilkan berdasarkan jumlah pengguna sebanyak 1000 pada kotak putih kiri.

4.3.2 Pengujian *Load Balancing* pada 2 *Web Server*

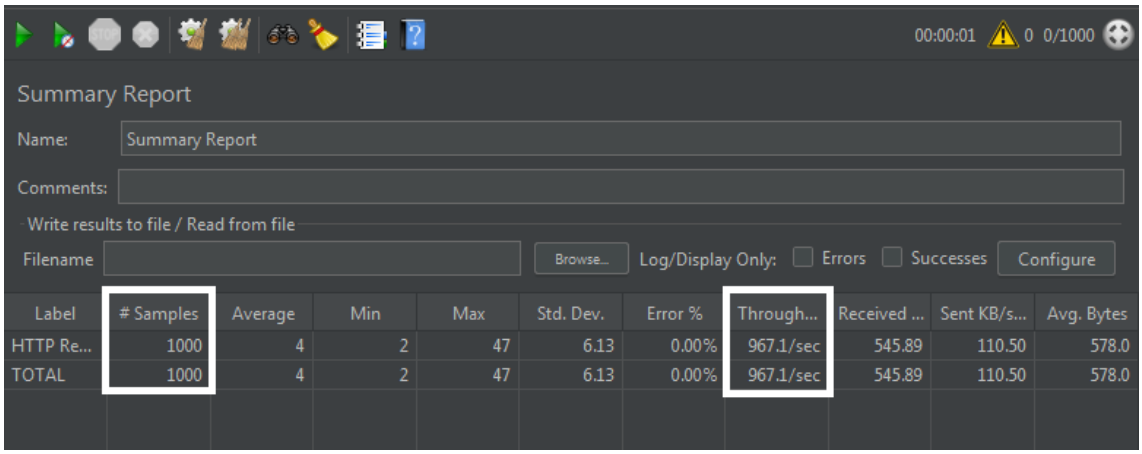
Skenario pengujian kedua yaitu jika salah satu dari *web server* sedang dalam kondisi *down* seperti pada Gambar 19.

```
Jul 08 23:22:28 web1 systemd[1]: Stopped The Apache HTTP Server.
Jul 08 23:22:28 web1 systemd[1]: apache2.service: Consumed 4.982s CPU time.
lines 1-15/15 (END)
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: inactive (dead) since Fri 2022-07-08 23:22:28 WIB; 1h 11min ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 7334 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
    Process: 7417 ExecStop=/usr/sbin/apachectl graceful-stop (code=exited, status=0/SUCCESS)
   Main PID: 7338 (code=exited, status=0/SUCCESS)
        CPU: 4.982s

Jul 08 23:12:20 web1 systemd[1]: Starting The Apache HTTP Server...
Jul 08 23:12:20 web1 systemd[1]: Started The Apache HTTP Server.
Jul 08 23:22:28 web1 systemd[1]: Stopping The Apache HTTP Server...
Jul 08 23:22:28 web1 systemd[1]: apache2.service: Succeeded.
Jul 08 23:22:28 web1 systemd[1]: Stopped The Apache HTTP Server.
Jul 08 23:22:28 web1 systemd[1]: apache2.service: Consumed 4.982s CPU time.
~
```

Gambar 19. *Web server* berhenti

Gambar 19 adalah status jika *web server* berhenti menjalankan program. Hasil pengujian dapat diperhatikan pada Gambar 20.



Gambar 20. *Throughput* 2 *web server*

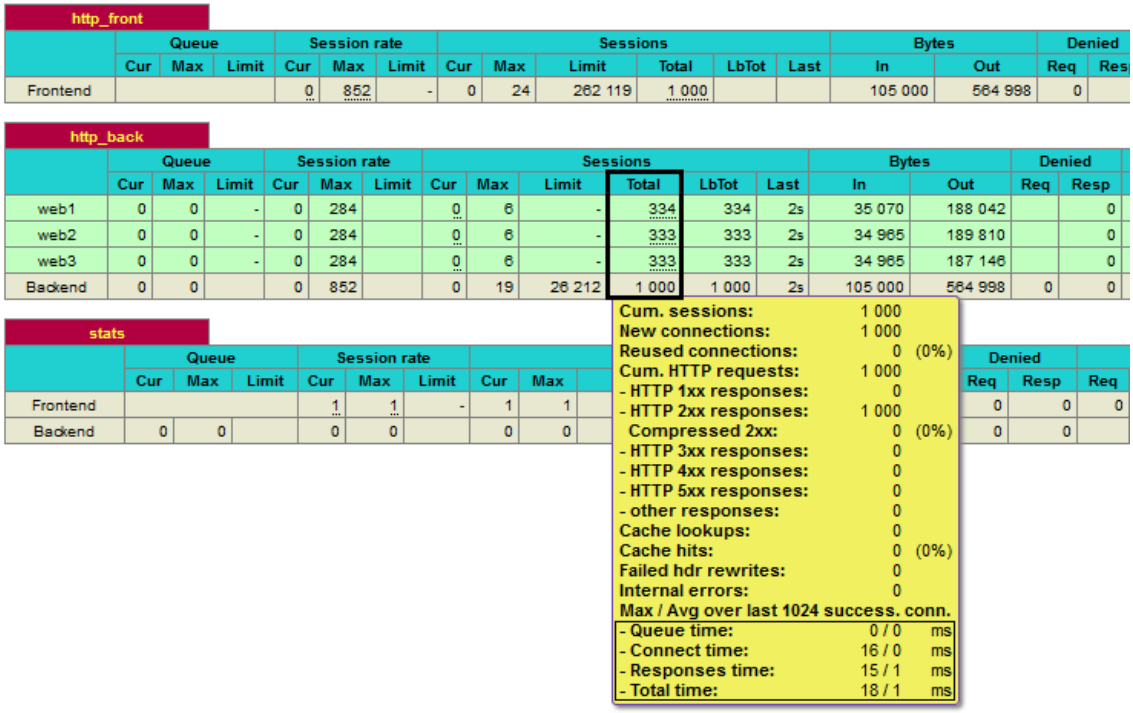
Gambar 20 memperlihatkan hasil *load balancing* dalam mengirim jumlah paket untuk dikirim ke 2 *web server*. Pada kotak putih bagian kanan adalah nilai *throughput* dari yang dapat dikirim dalam satu waktu. Semakin sedikit jumlah *server* yang tersedia maka akan mempengaruhi nilai *throughput.*

4.4 Hasil Pengujian *Response Time Web Server*

Pengujian dilakukan untuk mengetahui *response time web server* dalam menggunakan klaster. Pengukuran *response time* menggunakan data yang ditampilkan Haproxy dalam satuan *milliseconds* dengan menjalankan 2 skenario pada kondisi *web server*. Jumlah pengguna pengujian berdasarkan Gambar 17.

4.4.1 Pengujian *Response Time* pada 3 *Web Server*

Skenario pengujian pertama yaitu dengan semua *web server* dalam kondisi aktif untuk menerima paket yang dikirim oleh *load balancing* dan kemudian diproses *web server* untuk dikirim kembali ke pengguna. Hasil dari pengujian skenario pertama *response time* dapat dilihat pada Gambar 21.



http_front

	Queue			Session rate			Sessions						Bytes		Denied	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Res
Frontend				0	852	-	0	24	262 119	1 000			105 000	564 998	0	

http_back

	Queue			Session rate			Sessions						Bytes		Denied	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp
web1	0	0	-	0	284		0	6	-	334	334	2s	35 070	188 042		0
web2	0	0	-	0	284		0	6	-	333	333	2s	34 965	189 810		0
web3	0	0	-	0	284		0	6	-	333	333	2s	34 965	187 146		0
Backend	0	0		0	852		0	19	26 212	1 000	1 000	2s	105 000	564 998	0	0

stats

	Queue			Session rate					Denied		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Req	Resp	Req
Frontend				1	1	-	1	1	0	0	0
Backend	0	0		0	0		0	0	0	0	

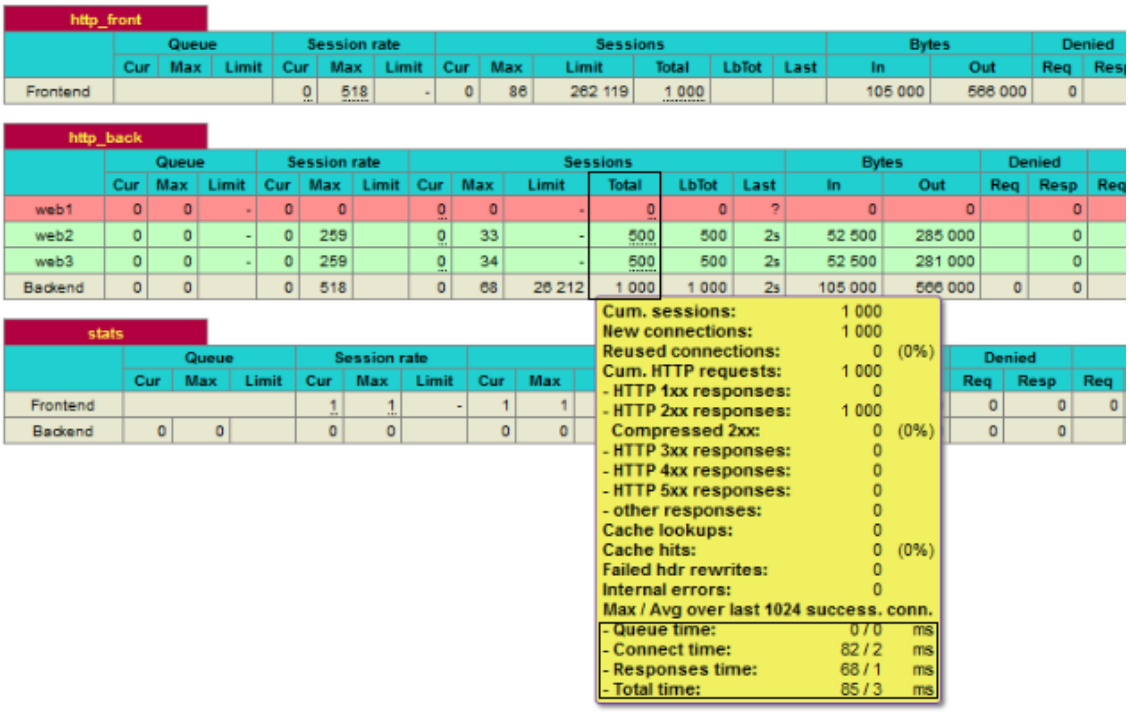
Cum. sessions:	1 000	
New connections:	1 000	
Reused connections:	0	(0%)
Cum. HTTP requests:	1 000	
- HTTP 1xx responses:	0	
- HTTP 2xx responses:	1 000	
Compressed 2xx:	0	(0%)
- HTTP 3xx responses:	0	
- HTTP 4xx responses:	0	
- HTTP 5xx responses:	0	
- other responses:	0	
Cache lookups:	0	
Cache hits:	0	(0%)
Failed hdr rewrites:	0	
Internal errors:	0	
Max / Avg over last 1024 success. conn.		
- Queue time:	0 / 0	ms
- Connect time:	16 / 0	ms
- Responses time:	15 / 1	ms
- Total time:	18 / 1	ms

Gambar 21. Statistik Haproxy

Pada Gambar 21 terdapat tanda yang menjelaskan setiap proses yang masuk pada *web server*. Tanda atas adalah hasil pembagian paket *request* dari *load balancing*, tanda bawah adalah nilai rata-rata *response time* dari semua *web server*.

4.4.2 Pengujian *Response Time* pada 2 *Web Server*

Skenario pengujian kedua yaitu jika salah satu dari *web server* sedang dalam kondisi *down*. Bagaimana cara *load balancing* dalam membagi paket *request* dalam kondisi ini. Hasil dari pengujian skenario kedua *response time* dapat dilihat pada Gambar 22.



Gambar 22. Statistik Haproxy

Gambar 22 menjelaskan perbedaan *response time* dan pembagian paket *request* yang lebih besar dibandingkan pada Gambar 21. *Response time* yang tinggi sangat berpengaruh dalam mengakses *website* dan menentukan tingkat kelayanan yang dimiliki. Tanda atas adalah hasil pembagian paket *request* dari *load balancing*, tanda bawah adalah nilai rata-rata *response time* dari *web server* yang tersedia.

4.5 Pembahasan Pengujian

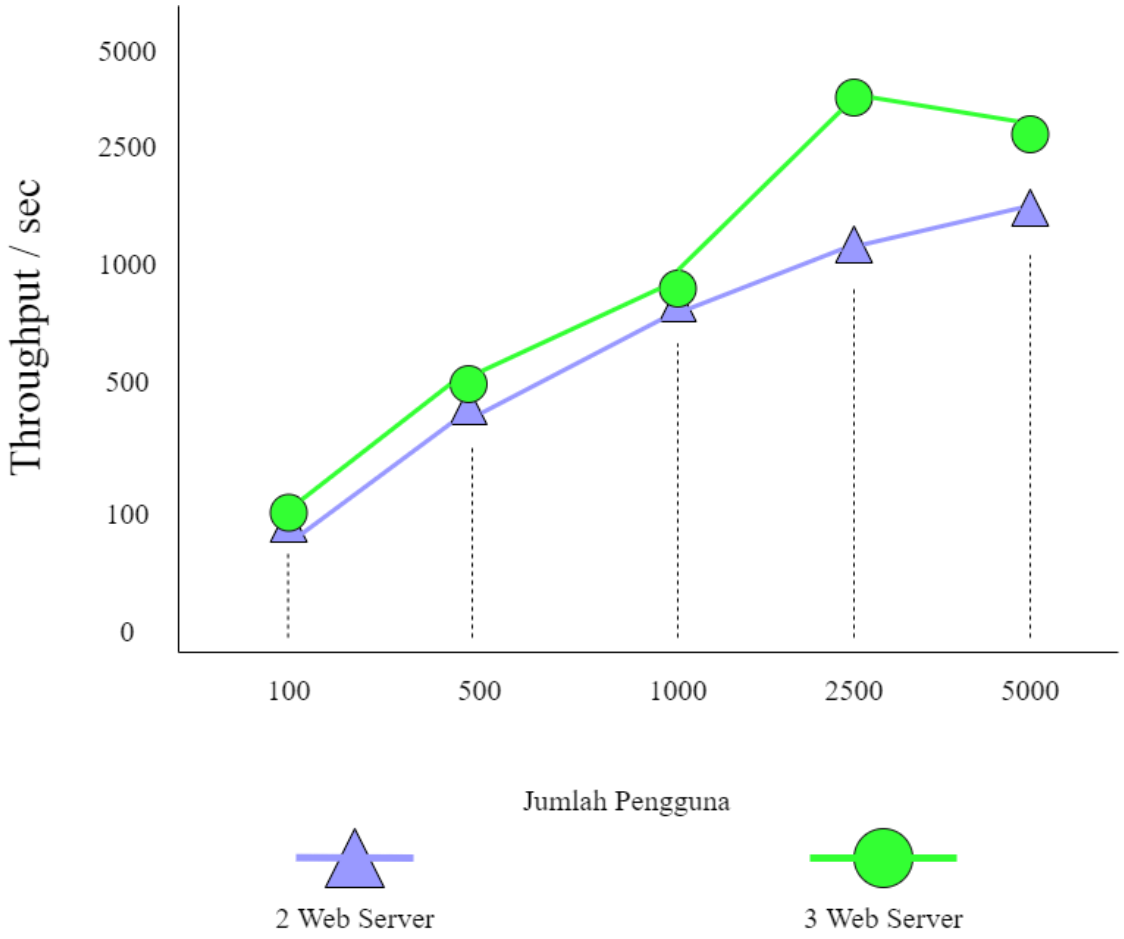
Pengujian dilakukan sebanyak 30 kali pada 2 skenario *web server* untuk pengambilan nilai rata-rata seperti *throughput load balancing*, *response time web server*. Data tersebut dapat dillihat pada Tabel 5 dan 6.

Tabel 5. Rata-rata *throughput load balancing*

Skenario	*Throughput* (*sec*)				
	100 *user*	500 *user*	1000 *user*	2500 *user*	5000 *user*
3 Web Server	102.5	500.7	989	3037.5	2607.2
2 Web Server	101.3	498.2	965.2	1056.4	1204.1

Berdasarkan Tabel 5, nilai *throughput* pada jumlah 5000 pengguna mengalami penurunan pada skenario 3 *web server*. Menurunnya nilai *throughput* dikarenakan daya tampung 1

load balancing yang dapat dikirim telah mencapai batas dan tidak dapat melebihi nilai 5000. Diperlukannya tambahan *load balancing* untuk pengujian yang lebih lanjut. Grafik dari nilai *throughput* pada Tabel 5 dapat dilihat pada Gambar 23.



Gambar 23. Grafik *throughput load balancing*

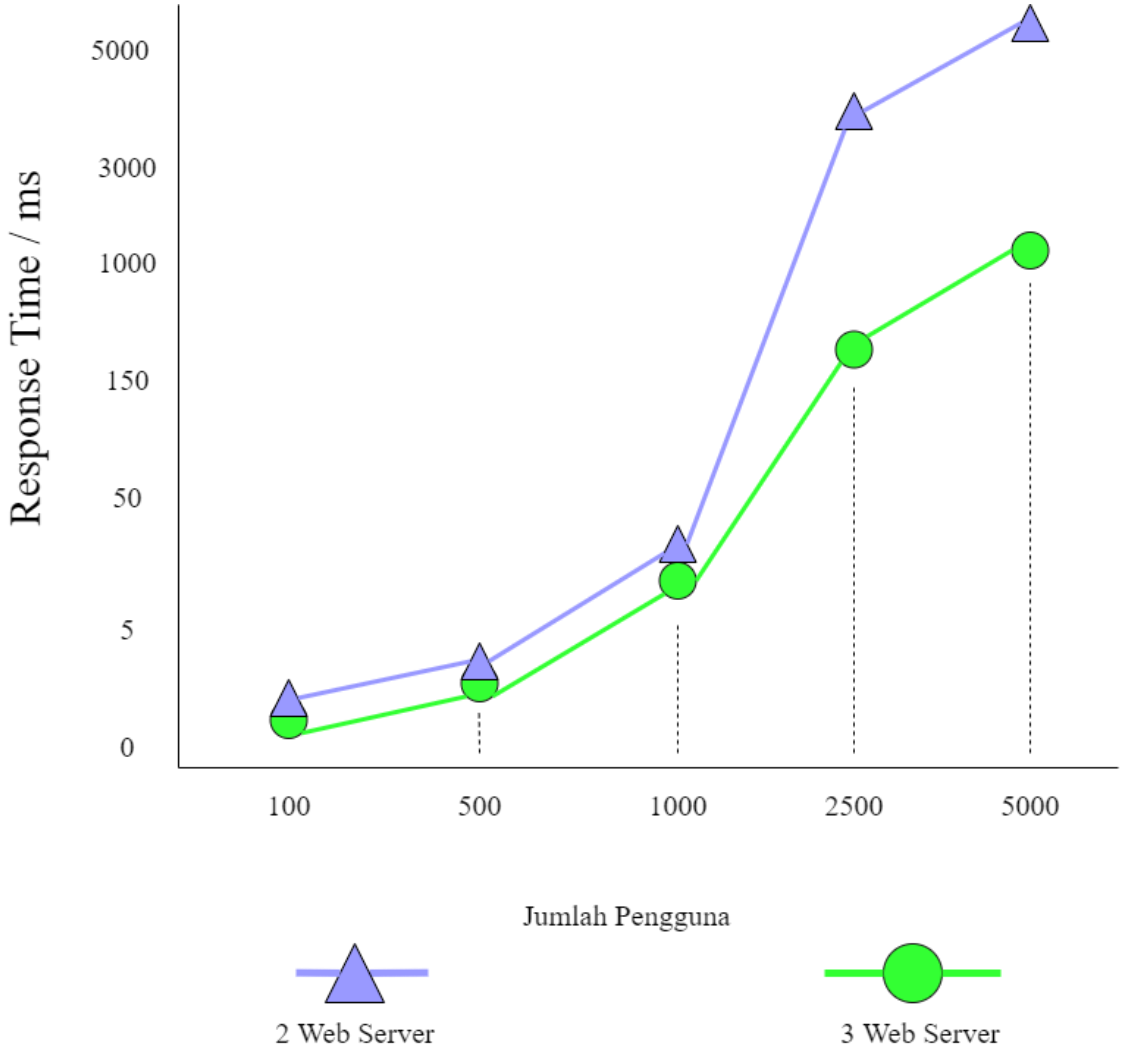
Gambar 23 memperlihatkan peningkatan yang jauh pada 2500 pengguna dengan nilai *throughput* melebih jumlah penggunanya. Hal ini diakibatkan dengan penggunaan switch dan kabel UTP yang mendukung gigabit.

Tabel 6. Rata-rata *response time web server*

Skenario	*Response Time* (*ms*)				
	100 *user*	500 *user*	1000 *user*	2500 *user*	5000 *user*
3 Web Server	1.7	2.2	23.9	179.3	1252.2
2 Web Server	1.8	3.3	32.8	3345.2	5396.4

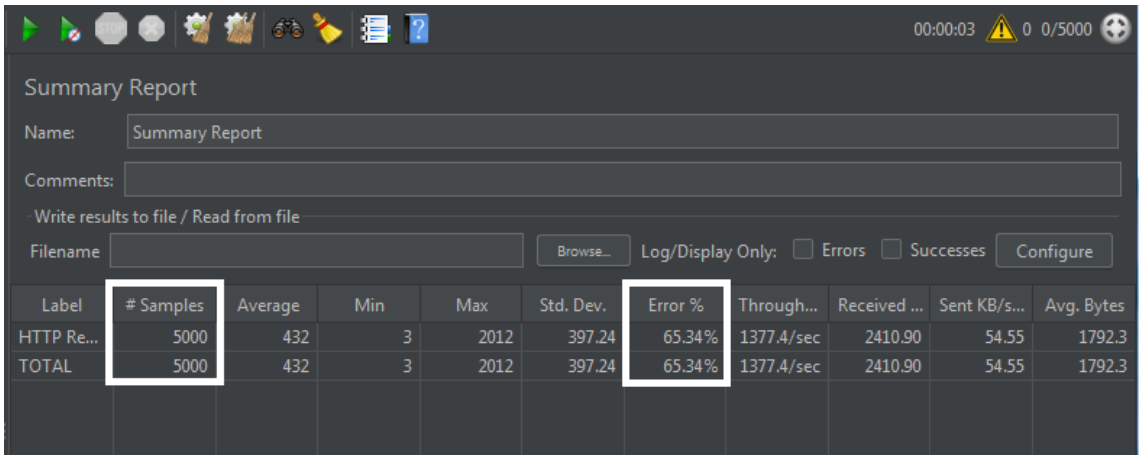
Berdasarkan nilai *throughput* pada Tabel 5, nilai tersebut akan sangat berpengaruh pada

response time yang dihasilkan. Tabel 6 menjelaskan mengenai nilai rata-rata *response time*, jika nilai *throughput* besar maka *response time* yang dihasilkan akan lebih cepat seperti yang terlihat pada perbedaan 2 skenario pada Tabel 6. Dalam penggunaan Raspberry Pi sebagai *web server* memiliki nilai *response time* yang cukup baik dalam jumlah pengguna mencapai 1000 (Putra, Yahya and Bhawiyuga, 2019). Grafik *response time* berdasarkan Tabel 6 dapat dilihat pada Gambar 24.



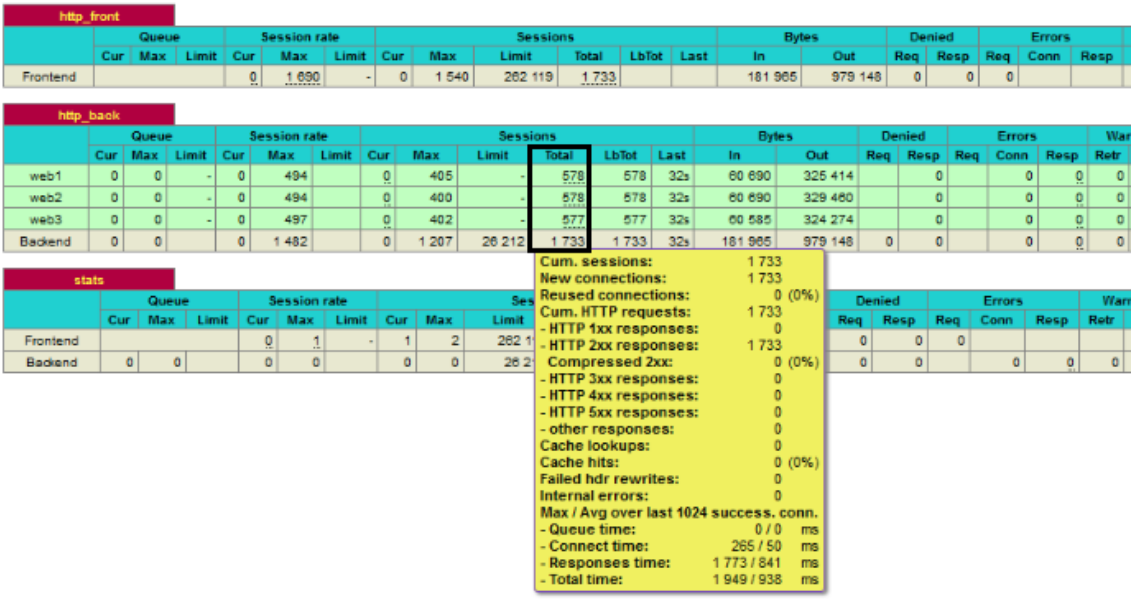
Gambar 24. Grafik *response time web server*

Gambar 24 menjelaskan perbedaan *response time* dari 2 skenario, semakin sedikit paket yang diterima setiap *web server* maka waktu untuk merespon akan semakin lebih cepat. Sebagai catatan, pada pengujian sampel terakhir dengan jumlah pengguna 5000, *load balancing* sempat mengalami *down* pada kedua skenario dalam menerima permintaan pengguna, sehingga *request* yang diterima *web server* tidak sesuai dan dilakukan pengujian ulang. *Load balancing* mengalami down dapat dilihat pada Gambar 25.



Gambar 25. ***Load balancing down***

Gambar 25 menampilkan jumlah error sekitar 65% dari total 5000 pengguna yang mengakibatkan pembagian paket akan tidak sesuai dengan jumlah pengguna. Gambar pembagian paket yang masuk ke *web server* dapat dilihat pada Gambar 26.



Gambar 26. Pembagian jumlah paket

Pembagian paket yang diterima *web server* tidak sesuai dengan jumlah pengguna yang mencapai 5000 yang terlihat pada Gambar 26. Paket yang hilang tidak akan diproses yang menyebabkan akses pengguna akan terputus.

Pengujian selanjutnya adalah perbandingan penggunaan CPU pada Raspberry Pi dengan 2 skenario yang telah dijalankan. Hal ini bertujuan untuk mengetahui tinjauannya dalam Islam sebagai komponen alternatif yang dapat menggantikan komputer sebagai server dengan penggunaan biaya yang rendah. Pengujian dilakukan dengan jumlah pengguna

berdasarkan pada Gambar 19, pengujian dilakukan sebanyak 10 kali untuk mengetahui nilai rata-rata dari penggunaan CPU pada setiap skenario. Pengambilan nilai penggunaan CPU dilakukan dengan *looping request* untuk dapat mengambil nilai yang sesuai karena Raspberry Pi akan terus menjalankan *web server*. Hasil pengujian dapat dilihat pada Tabel 7.

Tabel 7. Penggunaan CPU pada Raspberry Pi

Skenario	Penggunaan CPU (%)		
	Server 1	Server 2	Server 3
3 Web Server	55	51	52
2 Web Server	-	78	79

Tabel 7 adalah nilai rata-rata dari penggunaan CPU masing-masing Raspberry Pi yang menjalankan *web server*. Pada skenario 2 *web server*, kinerja CPU meningkat, dikarenakan klaster akan memaksimalkan kinerja dari perangkat jika yang digunakan lebih sedikit ketika menangani permintaan dalam jumlah yang banyak, sehingga menyebabkan CPU harus berjalan secara maximal. Dibutuhkannya *web server backup* adalah untuk mengantisipasi jika daya CPU terlalu tinggi yang bisa mengakibatkan layanan *web server* mengalami menurunan performa seperti *response time* yang lama, sehingga dengan adanya *backup web server* dapat menyeimbangkan beban pada CPU serta meningkatkan performa dari layanan *web server*.

Pengerjaan penelitian dengan menggunakan klaster Raspberry Pi telah sesuai dengan prinsip yang ada pada fiqih muamalah. Sebagaimana dalam kaidah fiqih dikatakan:

للوسائل حكم الغاية

Artinya: “Hukum sarana sama seperti hukum tujuannya.”

Berdasarkan arti dari kaidah fiqih tersebut yang menjelaskan sebuah sarana sama dengan hukum tujuannya. Dalam menyelesaikan penelitian ini hasil yang didapat akan memberikan manfaat bagi yang mempelajarinya dan memungkinkan untuk dapat digunakan kembali oleh orang lain di masa mendatang, sehingga menjadikan hukumnya adalah mubah karena memiliki tujuan yang baik untuk dapat orang lain pelajari maupun digunakan. Sama halnya dengan penggunaan Raspberry Pi sebagai komponen alternatif

pengganti komputer untuk menjalankan sebuah klaster dengan penggunaan biaya yang rendah.