

Laporan
Grimoire Kelompok B04



dibuat oleh:

Rayssa Ravelia

(5025211219)

Immanuel Pascanov Samosir

(5025211257)

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
2023

Nomor 8

Karena diminta untuk menuliskan grimoire, buatlah analisis hasil testing dengan 200 request dan 10 request/second masing-masing algoritma Load Balancer.

Jawab:

A. Nama Algoritma Load Balancer dan Report hasil testing pada Apache Benchmark

A.1. Algoritma Load Balancer: Round Robin

Algoritma Round Robin pada load balancer adalah metode sederhana namun efektif untuk mendistribusikan permintaan jaringan secara merata di antara sekumpulan server. Pada dasarnya, algoritma ini bekerja dengan memberikan setiap permintaan yang masuk kepada server berikutnya dalam daftar secara bergantian, memastikan bahwa setiap server mendapatkan jumlah permintaan yang seimbang tanpa mempertimbangkan beban atau kapasitas saat ini dari server tersebut. Ini serupa dengan sistem antrian, di mana setiap server mendapatkan giliran untuk menangani permintaan. Algoritma ini sangat berguna untuk beban kerja yang seragam dan ketika semua server memiliki kapasitas yang hampir sama, tetapi mungkin kurang optimal untuk skenario dengan beban kerja yang sangat bervariasi atau server dengan kapasitas yang tidak merata.

A.1.1. Apache Benchmark

Berikut ini adalah hasil testing Apache Benchmark untuk algoritma load balancer Round Robin:

```
root@Revolte:~# ab -n 200 -c 10 http://www.granz.channel.B04.com/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

```
Benchmarking www.granz.channel.B04.com (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests
```

```
Server Software:      nginx/1.14.2
Server Hostname:      www.granz.channel.B04.com
Server Port:          80
```

```
Document Path:        /
Document Length:       605 bytes
```

```
Concurrency Level:     10
Time taken for tests:   0.974 seconds
Complete requests:     200
```

Failed requests: 133
 (Connect: 0, Receive: 0, Length: 133, Exceptions: 0)
 Total transferred: 148533 bytes
 HTML transferred: 121133 bytes
 Requests per second: 205.27 [#/sec] (mean)
 Time per request: 48.717 [ms] (mean)
 Time per request: 4.872 [ms] (mean, across all concurrent requests)
 Transfer rate: 148.87 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	4	12 6.0	11	44
Processing:	17	35 9.1	34	82
Waiting:	16	35 9.0	33	81
Total:	21	47 13.3	44	106

Percentage of the requests served within a certain time (ms)

50%	44
66%	46
75%	49
80%	50
90%	58
95%	77
98%	101
99%	104
100%	106 (longest request)

A.1.2. htop pada masing-masing PHP worker

Melakukan htop pada masing-masing worker PHP setelah melakukan tes algoritma di load balancer adalah langkah penting untuk memastikan efisiensi dan stabilitas sistem. Hal ini memungkinkan kita untuk memonitor penggunaan sumber daya seperti CPU dan memori secara real-time, membantu dalam mengidentifikasi beban kerja yang tidak seimbang antara worker. Dengan ini, kita bisa menilai efektivitas distribusi beban oleh load balancer, mengidentifikasi worker yang mungkin mengalami kelebihan beban atau kekurangan sumber daya, dan mengambil tindakan korektif. Proses ini juga membantu dalam mengoptimalkan kinerja keseluruhan sistem dan memastikan bahwa aplikasi berjalan dengan lancar, efisien, dan dapat diandalkan, yang pada akhirnya meningkatkan pengalaman pengguna akhir.

Berikut ini adalah hasil htop pada masing-masing PHP worker:

A.1.2.1. htop pada Lawine

```

1  [||] 5.7% Tasks: 11, 0 thr; 1 running
2  [||] 1.1% Load average: 0.12 0.27 0.63
Mem[|||||||||||||||||] 757M/3.82G Uptime: 00:39:08
Swp[|] 0K/2.00G

  PID USER   PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---
112  root     20    0  2456    4      0  S   0.0  0.0   0:00.00 /gns3/bin/busyb
104  root     20    0  2456    4      0  S   0.0  0.0   0:00.01 /gns3/bin/busyb
101  root     20    0  2456   132    80  S   0.0  0.0   0:00.01 /tmp/gns3/bin/u
  1  root     20    0  4000  3332  2824  S   0.0  0.1   0:00.10 bash
9066 root     20    0  4788  3228  2796  R   0.6  0.1   0:00.25 htop
8976 root     20    0 61592  1736    52  S   0.0  0.0   0:00.00 nginx: master p
8977 www-data 20    0 62228  6784  4736  S   0.0  0.2   0:00.03 nginx: worker p
8978 www-data 20    0 62228  6776  4736  S   0.0  0.2   0:00.27 nginx: worker p
8498 root     20    0 188M   6472  3092  S   0.0  0.2   0:00.33 php-fpm: master
8499 www-data 20    0 189M   9792  6216  S   0.0  0.2   0:00.09 php-fpm: pool w
8500 www-data 20    0 189M 10512  6844  S   0.0  0.3   0:00.08 php-fpm: pool w

F1Help F2Setup F3Search F4Filter F5Free F6SortBy F7Nice -F8Nice +F9Kill F10Quit

```

Dari gambar hasil `htop` pada PHP worker "Lawine", dapat dilihat bahwa sistem memiliki sejumlah kecil tugas yang sedang berjalan dengan total 11 tugas, di mana 1 di antaranya aktif. Rata-rata beban kerja sistem adalah rendah, menunjukkan bahwa sistem tidak berada di bawah tekanan yang signifikan pada saat tangkapan layar diambil.

Pada daftar proses, kita bisa melihat bahwa tidak ada proses yang secara signifikan menggunakan CPU atau memori. Proses nginx dan php-fpm muncul, yang menunjukkan bahwa PHP worker ini terlibat dalam melayani permintaan web. Bar memori di bagian atas menunjukkan bahwa ada cukup banyak memori yang tersedia dan tidak sepenuhnya digunakan, sementara bar CPU menunjukkan penggunaan CPU yang sangat rendah. Ini menunjukkan bahwa dari perspektif sumber daya, PHP worker ini beroperasi dalam keadaan normal dan stabil saat pengambilan gambar.

A.1.2.2. htop pada Linie

```

< X Revolte X Lawine X Linie X > - X

1  [||] 3.6% Tasks: 11, 0 thr; 1 running
2  [||] 1.3% Load average: 0.14 0.25 0.62
Mem[|||||] 757M/3.82G Uptime: 00:39:43
Swp[ ] 0K/2.00G

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
9058 root 20 0 4788 3184 2760 R 0.7 0.1 0:00.05 htop
1 root 20 0 4000 3188 2688 S 0.0 0.1 0:00.13 bash
85 root 20 0 2456 140 80 S 0.0 0.0 0:00.00 /tmp/gns3/bin/u
92 root 20 0 2456 4 0 S 0.0 0.0 0:00.02 /gns3/bin/busyb
99 root 20 0 2456 4 0 S 0.0 0.0 0:00.00 /gns3/bin/busyb
8497 root 20 0 188M 6664 3288 S 0.0 0.2 0:00.18 php-fpm: master
8498 www-data 20 0 189M 9892 6320 S 0.0 0.2 0:00.08 php-fpm: pool w
8499 www-data 20 0 189M 10712 7048 S 0.0 0.3 0:00.09 php-fpm: pool w
8982 root 20 0 61592 1752 72 S 0.0 0.0 0:00.00 nginx: master p
8983 www-data 20 0 62228 5680 3772 S 0.0 0.1 0:00.24 nginx: worker p
8984 www-data 20 0 62228 6804 4756 S 0.0 0.2 0:00.02 nginx: worker p

F1Help F2Setup F3Search F4Filter F5Free F6SortBy F7Nice -F8Nice +F9Kill F10Quit

```

Gambar `htop` yang ditampilkan menunjukkan status dari PHP worker bernama "Linie". Dari informasi yang disajikan, terlihat bahwa sistem memiliki 11 tugas dengan 1 tugas yang aktif, serupa dengan worker "Lawine" sebelumnya. Rata-rata beban kerja (load average) yang ditunjukkan cukup rendah, menandakan bahwa sistem tidak mengalami beban yang berat.

Dalam daftar proses, terlihat bahwa ada beberapa proses nginx, yang merupakan web server, dan php-fpm, yang merupakan FastCGI Process Manager untuk PHP. Penggunaan sumber daya oleh proses-proses ini juga rendah, dengan proses paling aktif hanya menggunakan sedikit CPU dan sejumlah kecil memori. Keseluruhan, bar memori dan CPU menunjukkan banyak sumber daya yang masih tersedia dan tidak digunakan secara intensif. Hal ini mengindikasikan bahwa PHP worker "Linie" berada dalam kondisi yang baik dan tidak menghadapi isu performa atau beban yang berlebihan pada waktu tangkapan layar diambil.

A.1.2.3. htop pada Lugner

The screenshot shows the htop interface with the following system status at the top:

```

1  [|||] 7.0% Tasks: 11, 0 thr; 1 running
2  [|||] 7.0% Load average: 0.25 0.26 0.61
Mem[|||||] 758M/3.82G Uptime: 00:40:11
Swp[ ] 0K/2.00G

```

Below the status, a table of processes is displayed:

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
8994	www-data	20	0	62228	6676	4628	S	0.0	0.2	0:00.25	nginx: worker p
8509	root	20	0	188M	6472	3092	S	0.0	0.2	0:00.14	php-fpm: master
1	root	20	0	4000	3344	2848	S	0.0	0.1	0:00.11	bash
8510	www-data	20	0	189M	9884	6308	S	0.0	0.2	0:00.09	php-fpm: pool w
8511	www-data	20	0	189M	10516	6848	S	0.0	0.3	0:00.09	php-fpm: pool w
9055	root	20	0	4788	3156	2732	R	0.7	0.1	0:00.04	htop
77	root	20	0	2456	4	0	S	0.0	0.0	0:00.03	/gns3/bin/busyb
8995	www-data	20	0	62228	6676	4628	S	0.0	0.2	0:00.01	nginx: worker p
86	root	20	0	2456	4	0	S	0.0	0.0	0:00.00	/gns3/bin/busyb
91	root	20	0	2456	132	80	S	0.0	0.0	0:00.00	/tmp/gns3/bin/u
8993	root	20	0	61592	1740	52	S	0.0	0.0	0:00.00	nginx: master p

At the bottom, there is a menu bar with options: F1 Help, F2 Setup, F3 Search, F4 Filter, F5 Free, F6 SortBy, F7 Nice, F8 Kill, F9 Kill, F10 Quit.

Dalam gambar hasil `htop` ini, kita melihat status dari PHP worker yang diberi nama "Lugner". Sistem ini menunjukkan jumlah tugas yang sama dengan sebelumnya, yaitu 11 tugas, dengan 1 yang aktif. Load average, yang mencerminkan beban kerja rata-rata sistem selama 1, 5, dan 15 menit terakhir, tetap pada tingkat yang rendah, menandakan bahwa tidak ada beban berlebih pada sistem.

Bar memori dan CPU memberikan gambaran visual bahwa tidak ada konsumsi berlebih pada sumber daya tersebut. Proses php-fpm terlihat beberapa kali dalam daftar, menandakan bahwa PHP sedang melayani permintaan. Penggunaan CPU dan memori oleh proses-proses ini sangat rendah, menunjukkan bahwa worker "Lugner" tidak mengalami tekanan sumber daya yang berarti. Ini menunjukkan bahwa sistem berjalan dengan lancar dan efisien pada waktu pengambilan gambar dan tidak ada indikasi masalah kinerja yang jelas.

A.2. Algoritma Load Balancer: Least-Connection

Algoritma Least Connection dalam load balancing adalah metode canggih yang bertujuan untuk mendistribusikan permintaan ke server dengan jumlah koneksi aktif terendah saat ini. Berbeda dengan Round Robin yang mendistribusikan permintaan secara bergantian tanpa mempertimbangkan beban saat ini, algoritma Least Connection memprioritaskan server yang paling sedikit terbebani. Ini memastikan bahwa server yang sudah menerima banyak koneksi tidak akan terbebani lebih lanjut, sehingga membantu dalam mengelola beban kerja secara lebih efisien dan mencegah overloading pada server tertentu. Algoritma ini sangat efektif dalam lingkungan di mana ada perbedaan signifikan

dalam waktu respons atau beban kerja antar server, serta dalam skenario di mana permintaan mungkin memerlukan waktu yang bervariasi untuk diproses.

A.2.1. Apache Benchmark

Berikut ini adalah hasil testing Apache Benchmark untuk algoritma load balancer Least Connection:

```
root@Revolte:~# ab -n 200 -c 10 http://www.granz.channel.B04.com/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

```
Benchmarking www.granz.channel.B04.com (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests
```

```
Server Software:      nginx/1.14.2
Server Hostname:      www.granz.channel.B04.com
Server Port:          80
```

```
Document Path:        /
Document Length:      606 bytes
```

```
Concurrency Level:    10
Time taken for tests:  0.474 seconds
Complete requests:    200
Failed requests:       61
  (Connect: 0, Receive: 0, Length: 61, Exceptions: 0)
Total transferred:    148539 bytes
HTML transferred:     121139 bytes
Requests per second:  421.66 [#/sec] (mean)
Time per request:     23.716 [ms] (mean)
Time per request:     2.372 [ms] (mean, across all concurrent requests)
Transfer rate:        305.82 [Kbytes/sec] received
```

```
Connection Times (ms)
      min mean[+/-sd] median max
Connect:    1   6  1.8    6   10
Processing:  6  18  2.9   18   24
Waiting:    4  17  2.9   18   24
Total:      6  23  3.9   23   32
```

Percentage of the requests served within a certain time (ms)

50%	23
66%	25
75%	26
80%	26
90%	28
95%	29
98%	30
99%	31
100%	32 (longest request)

A.2.2. htop pada masing-masing PHP worker

Melakukan `htop` pada masing-masing worker PHP setelah menguji algoritma di load balancer yang menggunakan metode Least Connection adalah langkah penting untuk memastikan efisiensi dan stabilitas sistem. `htop` memberikan pandangan mendetail tentang penggunaan sumber daya oleh masing-masing proses, termasuk penggunaan CPU dan memori. Dengan memantau ini, kita dapat mengevaluasi seberapa baik distribusi beban kerja di antara worker PHP oleh algoritma Least Connection. Jika terdapat ketimpangan dalam distribusi beban, `htop` akan membantu mengidentifikasi worker yang kelebihan beban atau kurang terpakai. Hal ini penting untuk mengoptimalkan kinerja dan mencegah titik kegagalan yang bisa disebabkan oleh overloading pada worker tertentu. Selain itu, analisis ini juga membantu dalam penyesuaian kapasitas dan skala sumber daya untuk memenuhi kebutuhan beban kerja secara lebih efektif.

Berikut ini adalah hasil htop pada masing-masing PHP worker:

A.2.2.1. htop pada Lawine


```

1  [|||] 1.4% Tasks: 11, 0 thr; 1 running
2  [|||] 1.4% Load average: 0.04 0.17 0.17
Mem[|||||] 777M/3.82G Uptime: 01:19:32
Swp[ ] 0K/2.00G

  PID USER      PRI  NI  VIRT   RES   SHR S  CPU% MEM%   TIME+  Command
 109 root        20   0  2456    4      0 S   0.0  0.0   0:00.00 /gns3/bin/busyb
 102 root        20   0  2456   136    80 S   0.0  0.0   0:00.01 /tmp/gns3/bin/u
   99 root        20   0  2456   136    80 S   0.0  0.0   0:00.01 /tmp/gns3/bin/u
    1 root        20   0  4000  3304   2792 S   0.0  0.1   0:00.05 bash
 9133 root        20   0  4788   3124   2696 R   0.0  0.1   0:00.00 htop
9092 root        20   0 61592  1732    52 S   0.0  0.0   0:00.00 nginx: master p
9093 www-data    20   0 62228  5660   3748 S   0.0  0.1   0:00.00 nginx: worker p
9094 www-data    20   0 62228  5660   3748 S   0.0  0.1   0:00.01 nginx: worker p
8496 root        20   0  188M   6700   3316 S   0.0  0.2   0:00.24 php-fpm: master
8497 www-data    20   0  189M  9996   6428 S   0.0  0.2   0:00.00 php-fpm: pool w
8498 www-data    20   0  189M 10868   7200 S   0.0  0.3   0:00.00 php-fpm: pool w

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit

```

Dari gambar `htop` yang ditampilkan, kita dapat melihat berbagai informasi mengenai penggunaan sumber daya oleh worker PHP bernama Lawine dan proses lainnya. Pada bagian atas, terdapat indikator untuk penggunaan CPU dan memori, yang menunjukkan bahwa sebagian besar memori sedang digunakan, namun CPU tampaknya tidak terlalu sibuk, dengan rata-rata beban kerja yang cukup rendah. Ini mengindikasikan bahwa meskipun ada penggunaan memori yang signifikan, CPU masih memiliki kapasitas untuk menangani lebih banyak tugas.

Dalam daftar proses, kita bisa melihat bahwa ada beberapa instansi PHP-fpm (PHP FastCGI Process Manager) yang sedang berjalan, yang sesuai dengan worker PHP Lawine. PID (Process ID), pengguna (USER), prioritas (PRI), nilai nice (NI), serta penggunaan memori (RES) dan CPU (CPU%) dapat dilihat. Proses-proses ini tidak menunjukkan penggunaan CPU yang tinggi pada saat tangkapan layar diambil, yang mungkin mengindikasikan bahwa load balancer dengan algoritma Least Connection telah berhasil mendistribusikan beban permintaan secara merata atau bahwa tidak banyak permintaan yang masuk pada saat itu. Ini juga dapat mengindikasikan bahwa konfigurasi server cukup efisien untuk menangani beban kerja saat ini tanpa menimbulkan tekanan yang berlebihan pada sumber daya.

A.2.2.2.htop pada Linie

```

< X Revolte X Linie X > - X

1 [||| 2.7%] Tasks: 11, 0 thr; 1 running
2 [||| 2.0%] Load average: 0.15 0.18 0.17
Mem[|||||] 777M/3.82G Uptime: 01:20:11
Swp[ 0K/2.00G]

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
9125 root 20 0 4788 3272 2840 R 0.0 0.1 0:00.00 htop
1 root 20 0 4000 3220 2724 S 0.0 0.1 0:00.07 bash
83 root 20 0 2456 136 80 S 0.0 0.0 0:00.01 /tmp/gns3/bin/u
85 root 20 0 2456 4 0 S 0.0 0.0 0:00.02 /gns3/bin/busyb
94 root 20 0 2456 4 0 S 0.0 0.0 0:00.00 /gns3/bin/busyb
8600 root 20 0 188M 6672 3292 S 0.0 0.2 0:00.03 php-fpm: master
8601 www-data 20 0 189M 10000 6436 S 0.0 0.2 0:00.00 php-fpm: pool w
8602 www-data 20 0 189M 10764 7100 S 0.0 0.3 0:00.00 php-fpm: pool w
9083 root 20 0 61592 1748 60 S 0.0 0.0 0:00.00 nginx: master p
9084 www-data 20 0 62228 5712 3800 S 0.0 0.1 0:00.02 nginx: worker p
9085 www-data 20 0 62228 5712 3800 S 0.0 0.1 0:00.00 nginx: worker p

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit

```

Dari gambar `htop` yang disediakan, terlihat bahwa PHP worker bernama Linie juga menjadi bagian dari sistem yang saat ini diawasi. Indikator di bagian atas menunjukkan beban rata-rata yang sedikit lebih tinggi dibandingkan dengan Lawine, namun masih pada level yang terkelola dengan baik. Beban rata-rata ini diukur dalam interval 1, 5, dan 15 menit, memberikan indikasi bahwa sistem tidak mengalami kelebihan beban pada saat sekarang.

Dalam daftar proses, terlihat beberapa proses PHP-fpm yang berjalan, yang menunjukkan penggunaan memori (RES) dan CPU (CPU%). Penggunaan CPU oleh proses-proses ini terlihat rendah, mengindikasikan bahwa tidak ada permintaan berat yang sedang diproses pada saat tangkapan layar ini. Penggunaan memori yang ditunjukkan oleh proses PHP-fpm ini terlihat konsisten dan tidak menunjukkan kebocoran memori atau penggunaan yang tidak semestinya. Hal ini menunjukkan bahwa load balancer dengan algoritma Least Connection mungkin telah melakukan pekerjaannya dengan baik, mendistribusikan beban secara merata di antara worker PHP tanpa menyebabkan kelebihan beban pada worker individu. Ini membantu dalam menjaga kinerja sistem yang optimal dan menjamin bahwa tidak ada server yang menjadi titik gagal karena kelebihan permintaan.

A.2.2.3. htop pada Lugner

```

1  [|] 2.0% Tasks: 11, 0 thr; 1 running
2  [|] 1.4% Load average: 0.10 0.16 0.17
Mem[|||||]778M/3.82G Uptime: 01:20:36
Swp[ ] 0K/2.00G

  PID USER   PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---
  1 root      20    0  4000   3324   2828  S   0.0  0.1   0:00.05 bash
8606 root      20    0  188M   6416   3040  S   0.7  0.2   0:00.03 php-fpm: master
  86 root      20    0  2456     4      0  S   0.0  0.0   0:00.01 /gns3/bin/busyb
9084 www-data  20    0  62228   5616   3700  S   0.0  0.1   0:00.01 nginx: worker p
  83 root      20    0  2456   136     80  S   0.0  0.0   0:00.00 /tmp/gns3/bin/u
  93 root      20    0  2456     4      0  S   0.0  0.0   0:00.00 /gns3/bin/busyb
8607 www-data  20    0  189M   9848   6284  S   0.0  0.2   0:00.00 php-fpm: pool w
8608 www-data  20    0  189M  10416   6752  S   0.0  0.3   0:00.00 php-fpm: pool w
9083 root      20    0  61592   1736     52  S   0.0  0.0   0:00.00 nginx: master p
9085 www-data  20    0  62228   5616   3700  S   0.0  0.1   0:00.00 nginx: worker p
9124 root      20    0  4788   3264   2840  R   0.0  0.1   0:00.00 htop

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit

```

Dalam gambar `htop` untuk PHP worker bernama Lugner, kita dapat melihat detail penggunaan sumber daya yang serupa dengan yang sebelumnya. Beban rata-rata sistem, seperti yang terlihat pada indikator di bagian atas, masih berada pada tingkat yang rendah, yang menandakan bahwa sistem secara keseluruhan tidak mengalami kelebihan beban.

Fokus pada proses yang berkaitan dengan PHP worker Lugner, terlihat bahwa beberapa proses PHP-fpm telah diinstansiasi. Proses-proses ini memiliki penggunaan memori (RES) yang signifikan, namun penggunaan CPU (CPU%) mereka tetap rendah, menunjukkan bahwa permintaan saat ini dapat dikelola tanpa mengakibatkan CPU overloading. Hal ini dapat menunjukkan bahwa distribusi beban kerja yang dilakukan oleh load balancer dengan algoritma Least Connection berjalan efektif, di mana tidak ada server yang terbebani secara tidak proporsional. Observasi ini penting untuk memastikan bahwa semua worker PHP dapat melayani permintaan dengan efisien, yang pada gilirannya mempertahankan kinerja sistem yang stabil dan responsif.

A.3. Algoritma Load Balancer: IP Hash

Algoritma IP Hash pada load balancing adalah metode yang menggunakan alamat IP klien sebagai kunci untuk menentukan server mana yang akan melayani permintaan. Dalam metode ini, algoritma mengambil alamat IP klien dan menerapkan fungsi hash untuk menetapkan secara konsisten permintaan dari IP yang sama ke server yang sama, selama tidak ada perubahan dalam jumlah server. Ini memastikan bahwa klien akan memiliki sesi yang persisten dengan server yang sama, yang berguna untuk aplikasi yang membutuhkan

keadaan tetap (statefulness), seperti keranjang belanja dalam e-commerce atau sesi yang disesuaikan dalam aplikasi web. Metode IP Hash sangat bermanfaat untuk mempertahankan 'affinity' klien-server, namun mungkin tidak seefektif metode lain dalam kasus beban kerja yang tidak merata atau ketika skala server berubah sering, karena ini bisa mengakibatkan distribusi beban yang tidak seimbang.

A.3.1. Apache Benchmark

Berikut ini adalah hasil testing Apache Benchmark untuk algoritma load balancer IP Hash:

```
root@Revolte:~# ab -n 200 -c 10 http://www.granz.channel.B04.com/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

```
Benchmarking www.granz.channel.B04.com (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests
```

```
Server Software:      nginx/1.14.2
Server Hostname:      www.granz.channel.B04.com
Server Port:          80
```

```
Document Path:        /
Document Length:      606 bytes
```

```
Concurrency Level:    10
Time taken for tests:  0.413 seconds
Complete requests:    200
Failed requests:       0
Total transferred:    148600 bytes
HTML transferred:     121200 bytes
Requests per second:  483.91 [#/sec] (mean)
Time per request:     20.665 [ms] (mean)
Time per request:     2.067 [ms] (mean, across all concurrent requests)
Transfer rate:        351.12 [Kbytes/sec] received
```

```
Connection Times (ms)
      min mean[+/-sd] median max
Connect:    1  5  1.3    5    9
Processing:  4 15  3.3   15   25
```

Waiting:	4	15	3.3	15	25
Total:	5	20	3.7	20	29

Percentage of the requests served within a certain time (ms)

50%	20
66%	22
75%	23
80%	23
90%	25
95%	26
98%	27
99%	27
100%	29 (longest request)

A.3.2. htop pada masing-masing PHP worker

Melakukan `htop` pada masing-masing worker PHP setelah menjalankan tes algoritma IP Hash pada load balancer adalah krusial untuk memastikan bahwa distribusi sesi klien ke server adalah seimbang sesuai dengan desain algoritma. Algoritma IP Hash bertujuan untuk konsistensi sesi dengan mengarahkan permintaan dari IP yang sama ke server yang sama, namun hal ini bisa mengakibatkan distribusi beban yang tidak merata jika ada sejumlah klien dengan volume permintaan tinggi yang terus menerus diarahkan ke server tertentu. `htop` menyediakan visualisasi real-time dari penggunaan sumber daya oleh setiap worker PHP, memungkinkan administrator sistem untuk memantau dan menganalisis penggunaan CPU dan memori secara langsung. Hal ini memungkinkan pengidentifikasian cepat terhadap potensi ketidakseimbangan beban atau titik kegagalan, serta membantu dalam penyesuaian dan peningkatan konfigurasi load balancing untuk mencapai optimalisasi kinerja dan keandalan sistem secara keseluruhan. Berikut ini adalah hasil htop pada masing-masing PHP worker:

A.3.2.1. htop pada Lawine

```

1 [ 0.0%] Tasks: 11, 0 thr; 1 running
2 [ 0.0%] Load average: 0.02 0.09 0.14
Mem[|||||]780M/3.82G Uptime: 01:24:04
Swp[ ] 0K/2.00G

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
 109 root         20   0  2456    4      0  S   0.0  0.0   0:00.00 /gns3/bin/busyb
102 root         20   0  2456   136    80  S   0.0  0.0   0:00.00 /gns3/bin/busyb
 99 root         20   0  2456   136    80  S   0.0  0.0   0:00.01 /tmp/gns3/bin/u
  1 root         20   0  4000  3304   2792 S   0.0  0.1   0:00.05 bash
9148 root         20   0  4788  3320   2892 R  200.  0.1   0:00.01 htop
9092 root         20   0  61592 1732    52  S   0.0  0.0   0:00.00 nginx: master p
9093 www-data     20   0  62228 5660   3748 S   0.0  0.1   0:00.00 nginx: worker p
9094 www-data     20   0  62228 5660   3748 S   0.0  0.1   0:00.05 nginx: worker p
8496 root         20   0  188M  6700   3316 S   0.0  0.2   0:00.25 php-fpm: master
8497 www-data     20   0  189M  9996   6428 S   0.0  0.2   0:00.01 php-fpm: pool w
8498 www-data     20   0  189M 10868   7200 S   0.0  0.3   0:00.02 php-fpm: pool w

F1Help F2Setup F3Search F4Filter F5Free F6SortBy F7Nice F8Nice F9Kill F10Quit

```

Dari gambar `htop` di atas, terlihat bahwa PHP worker bernama Lawine beroperasi dalam kondisi yang tidak terlalu menuntut, yang ditandai dengan beban rata-rata yang sangat rendah pada sistem. Ini menunjukkan bahwa meskipun algoritma IP Hash telah dijalankan, permintaan jaringan mungkin belum menimbulkan tekanan yang berarti terhadap sumber daya CPU. Mengingat algoritma IP Hash cenderung mempertahankan koneksi klien dengan server yang sama, kondisi beban rendah ini dapat menunjukkan bahwa permintaan yang masuk ke worker Lawine secara khusus, dalam periode pengamatan ini, adalah terbatas atau permintaan tersebut tidak memerlukan banyak pemrosesan sumber daya.

Secara spesifik pada proses PHP-fpm yang berhubungan dengan worker Lawine, mereka menunjukkan penggunaan memori yang stabil dengan penggunaan CPU yang minimal. Hal ini dapat diinterpretasikan bahwa sesi yang didistribusikan ke Lawine melalui algoritma IP Hash adalah sesi dengan intensitas sumber daya yang rendah, atau bisa juga berarti bahwa pembagian beban permintaan yang ada terdistribusi dengan cukup rata antara worker yang tersedia. Namun, penting untuk mencatat bahwa dalam jangka panjang, algoritma IP Hash bisa mengakibatkan ketidakseimbangan beban jika sekelompok IP klien yang serupa mengirimkan jumlah permintaan yang tinggi secara konsisten. Oleh karena itu, monitoring dengan `htop` ini penting untuk memastikan bahwa tidak ada worker yang mengalami kelebihan beban seiring waktu dan semua sesi dihandle dengan efisien.

A.3.2.2. htop pada Linie

```

< X Eisen X Revolte X Linie X > - X

1 [ 0.0%] Tasks: 11, 0 thr; 1 running
2 [ 0.0%] Load average: 0.01 0.09 0.14
Mem[|||||||780M/3.82G] Uptime: 01:24:25
Swp[0K/2.00G]

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---
9133 root         20    0  4788   3212  2788  R   0.0   0.1   0:00.00 htop
   1 root         20    0  4000    320   272  S   0.0   0.1   0:00.07 bash
  83 root         20    0  2456    136    80  S   0.0   0.0   0:00.01 /tmp/gns3/bin/u
  85 root         20    0  2456     4     0  S   0.0   0.0   0:00.02 /gns3/bin/busyb
  94 root         20    0  2456     4     0  S   0.0   0.0   0:00.00 /gns3/bin/busyb
8600 root         20    0  188M   6672  3292  S   0.0   0.2   0:00.04 php-fpm: master
8601 www-data    20    0  189M  10000  6436  S   0.0   0.2   0:00.00 php-fpm: pool w
8602 www-data    20    0  189M  10764  7100  S   0.0   0.3   0:00.00 php-fpm: pool w
9083 root         20    0  61592  1748    60  S   0.0   0.0   0:00.00 nginx: master p
9084 www-data    20    0  62228   5712  3800  S   0.0   0.1   0:00.02 nginx: worker p
9085 www-data    20    0  62228   5712  3800  S   0.0   0.1   0:00.00 nginx: worker p

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit

```

Gambar `htop` untuk PHP worker bernama Linie menunjukkan bahwa sistem secara keseluruhan berada dalam keadaan istirahat atau hanya sedikit bekerja, seperti yang terindikasi oleh beban rata-rata yang sangat rendah. Proses-proses PHP-fpm yang terkait dengan Linie tampak menggunakan sejumlah memori yang sedang tetapi memiliki persentase CPU yang rendah pada saat pengambilan gambar. Ini bisa menandakan bahwa walaupun ada sesi yang persisten yang ditetapkan oleh algoritma IP Hash kepada Linie, permintaan yang dihandle tidak menuntut intensitas CPU yang tinggi, atau permintaan tersebut tersebar merata sepanjang waktu sehingga tidak membebani worker pada momen tertentu.

Penggunaan sumber daya yang rendah ini dalam konteks algoritma IP Hash dapat mengimplikasikan bahwa Linie mungkin tidak sering dipilih oleh fungsi hash sebagai target untuk permintaan baru, atau pola lalu lintas ke worker ini secara kebetulan lebih ringan. Ini menggarisbawahi pentingnya pemantauan berkelanjutan, karena algoritma IP Hash dapat menyebabkan distribusi beban yang tidak merata dalam jangka panjang, terutama jika skala server berubah atau jika ada fluktuasi dalam pola lalu lintas jaringan. Pemantauan dengan `htop` memberi wawasan penting tentang kinerja real-time dan membantu dalam menyesuaikan load balancing agar dapat merespons dinamika beban kerja secara efektif.

A.3.2.3. htop pada Lugner

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1	root	20	0	4000	3324	2828	S	0.0	0.1	0:00.05	bash
8606	root	20	0	188M	6416	3040	S	0.0	0.2	0:00.04	php-fpm: master
83	root	20	0	2456	136	80	S	0.0	0.0	0:00.01	/tmp/gns3/bin/u
86	root	20	0	2456	4	0	S	0.0	0.0	0:00.01	/gns3/bin/busyb
9084	www-data	20	0	62228	5616	3700	S	0.0	0.1	0:00.01	nginx: worker p
93	root	20	0	2456	4	0	S	0.0	0.0	0:00.00	/gns3/bin/busyb
8607	www-data	20	0	189M	9848	6284	S	0.0	0.2	0:00.00	php-fpm: pool w
8608	www-data	20	0	189M	10416	6752	S	0.0	0.3	0:00.00	php-fpm: pool w
9083	root	20	0	61592	1736	52	S	0.0	0.0	0:00.00	nginx: master p
9085	www-data	20	0	62228	5616	3700	S	0.0	0.1	0:00.00	nginx: worker p
9139	root	20	0	4788	3248	2820	R	0.0	0.1	0:00.00	htop

Gambar `htop` yang diberikan untuk PHP worker bernama Lugner menggambarkan sistem yang beroperasi dengan beban yang sangat ringan. Indikator beban rata-rata yang terlihat di bagian atas menunjukkan bahwa tidak ada banyak tugas yang diproses secara bersamaan, dan sistem memiliki banyak kapasitas sumber daya yang tersisa.

Proses-proses yang terkait dengan PHP worker Lugner, khususnya proses PHP-fpm, menunjukkan penggunaan memori yang moderat dan penggunaan CPU yang sangat rendah. Ini menandakan bahwa meskipun sesi klien mungkin ditetapkan untuk bertahan dengan worker ini karena algoritma IP Hash, permintaan yang datang tidak menimbulkan beban yang signifikan pada CPU. Hal ini dapat diartikan bahwa permintaan yang diarahkan ke Lugner berjalan efisien atau tidak memerlukan pemrosesan intensif. Pemantauan yang teratur dengan alat seperti `htop` sangat penting dalam kasus penggunaan algoritma IP Hash, karena dapat membantu mengidentifikasi apakah pembagian beban sesuai dengan ekspektasi, dan bahwa tidak ada server yang berada di bawah beban yang berlebihan atau kurang dimanfaatkan karena pola distribusi yang tetap.

A.4. Algoritma Load Balancer: Generic Hash

Algoritma Generic Hash pada load balancing, yang juga dikenal sebagai Consistent Hashing, adalah metode yang menggunakan fungsi hash untuk menentukan server mana yang akan melayani permintaan berdasarkan kunci yang didefinisikan, seperti alamat IP klien, header permintaan, atau parameter lainnya. Berbeda dengan IP Hash yang hanya menggunakan alamat IP, Generic Hash lebih fleksibel karena bisa menggunakan berbagai atribut dari permintaan

untuk menghasilkan kunci hash. Setelah kunci di-hash, hasilnya digunakan untuk menentukan server tujuan dengan cara yang merata dan konsisten. Keuntungan dari metode ini adalah kemampuannya untuk menyeimbangkan beban dengan lebih efektif dan menawarkan fleksibilitas dalam mempertahankan affinity klien-server bahkan saat server ditambahkan atau dihapus, dengan meminimalkan pengalihan lalu lintas yang tidak perlu. Ini sangat berguna dalam sistem yang dinamis di mana server sering berubah atau dalam penggunaan cloud yang skalabilitasnya sering berfluktuasi.

A.4.1. Apache Benchmark

Berikut ini adalah hasil testing Apache Benchmark untuk algoritma load balancer Generic Hash:

```
root@Revolte:~# ab -n 200 -c 10 http://www.granz.channel.B04.com/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

```
Benchmarking www.granz.channel.B04.com (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests
```

```
Server Software:      nginx/1.14.2
Server Hostname:      www.granz.channel.B04.com
Server Port:          80
```

```
Document Path:        /
Document Length:      606 bytes
```

```
Concurrency Level:    10
Time taken for tests:  0.422 seconds
Complete requests:    200
Failed requests:       0
Total transferred:    148600 bytes
HTML transferred:     121200 bytes
Requests per second:  474.23 [#/sec] (mean)
Time per request:     21.087 [ms] (mean)
Time per request:     2.109 [ms] (mean, across all concurrent requests)
Transfer rate:        344.10 [Kbytes/sec] received
```

```
Connection Times (ms)
      min mean[+/-sd] median  max
```

Connect:	1	5	1.6	5	14
Processing:	6	16	3.2	16	28
Waiting:	3	16	3.2	16	28
Total:	6	20	3.9	21	31

Percentage of the requests served within a certain time (ms)

50%	21
66%	22
75%	23
80%	23
90%	25
95%	27
98%	30
99%	31
100%	31 (longest request)

A.4.2. htop pada masing-masing PHP worker

Melakukan `htop` pada masing-masing worker PHP setelah menjalankan tes algoritma Generic Hash pada load balancer merupakan langkah vital untuk memverifikasi distribusi beban yang dilakukan oleh algoritma tersebut. Algoritma Generic Hash dirancang untuk menyebarkan beban kerja secara efisien dan merata berdasarkan kunci yang di-hash, yang idealnya menghasilkan distribusi yang seimbang tanpa memperhatikan server mana yang sedang sibuk atau tidak. Dengan `htop`, kita dapat memantau penggunaan CPU dan memori secara real-time untuk setiap worker, memastikan bahwa tidak ada satu worker pun yang secara tidak proporsional menerima lebih banyak beban, yang dapat mengindikasikan ketidakseimbangan dalam hashing atau potensi masalah dengan konfigurasi server. Memantau setiap worker PHP juga membantu dalam mengidentifikasi isu-isu seperti kebocoran memori, overloading server, dan membantu dalam pengambilan keputusan untuk penyesuaian dan peningkatan infrastruktur yang berkelanjutan.

Berikut ini adalah hasil htop pada masing-masing PHP worker:

A.4.2.1. htop pada Lawine

```

1 [ 0.0%] Tasks: 11, 0 thr; 1 running
2 [ 0.0%] Load average: 0.02 0.05 0.10
Mem[|||||779M/3.82G] Uptime: 01:28:11
Swp[ 0K/2.00G]

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
109 root 20 0 2456 4 0 S 0.0 0.0 0:00.00 /gns3/bin/busybox
102 root 20 0 2456 136 80 S 0.0 0.0 0:00.01 /tmp/gns3/bin/udev
1 root 20 0 4000 3304 2792 S 0.0 0.1 0:00.05 bash
9163 root 20 0 4788 3212 2788 R 0.0 0.1 0:00.00 htop
9092 root 20 0 61592 1732 52 S 0.0 0.0 0:00.00 nginx: master process
9093 www-data 20 0 62228 5660 3748 S 0.0 0.1 0:00.00 nginx: worker process
9094 www-data 20 0 62228 5660 3748 S 0.0 0.1 0:00.05 nginx: worker process
8496 root 20 0 188M 6700 3316 S 0.0 0.2 0:00.26 php-fpm: master process
8497 www-data 20 0 189M 9996 6428 S 0.0 0.2 0:00.01 php-fpm: pool www
8498 www-data 20 0 189M 10868 7200 S 0.0 0.3 0:00.02 php-fpm: pool www

F1Help F2Setup F3Search F4Filter F5Free F6SortBy F7Nice F8Nice +F9Kill F10Quit

```

Dari gambar `htop` yang disajikan, PHP worker bernama Lawine tampaknya mengalami beban kerja yang rendah, seperti yang ditunjukkan oleh indikator beban rata-rata sistem yang berada pada nilai yang rendah. Proses PHP-fpm yang terkait dengan Lawine menunjukkan penggunaan memori yang moderat dan penggunaan CPU yang rendah, yang menandakan bahwa permintaan yang dihandle oleh worker ini tidak memerlukan banyak pemrosesan komputasi atau bahwa permintaan tersebut terdistribusi merata sepanjang waktu.

Dalam konteks algoritma Generic Hash di load balancer, pengamatan ini bisa mengindikasikan bahwa fungsi hash yang digunakan berhasil mendistribusikan permintaan secara efisien di antara semua worker yang tersedia. Keefektifan distribusi ini penting untuk memastikan bahwa tidak ada worker yang mengalami kelebihan beban, yang bisa menurunkan kinerja atau mengakibatkan downtime. Jika beban tetap rendah secara konsisten, ini juga bisa menjadi indikasi bahwa ada ruang untuk optimasi sumber daya, yang mana sumber daya server bisa dikurangi untuk mengefisienkan biaya tanpa mengorbankan kinerja.

A.4.2.2. htop pada Linie

```

< te X Eisen X Lawine X Linie X > - X

1 [ 0.0%] Tasks: 11, 0 thr; 1 running
2 [ 0.0%] Load average: 0.01 0.05 0.10
Mem[|||||] 779M/3.82G Uptime: 01:28:31
Swp[ 0K/2.00G]

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---
9148 root         20    0  4788   3216   2788  R   0.0   0.1   0:00.00 htop
    1 root         20    0  4000   3224   2724  S   0.0   0.1   0:00.07 bash
   83 root         20    0  2456    136     80  S   0.0   0.0   0:00.01 /tmp/gns3/bin/u
   85 root         20    0  2456     4      0  S   0.0   0.0   0:00.02 /gns3/bin/busyb
   94 root         20    0  2456     4      0  S   0.0   0.0   0:00.00 /gns3/bin/busyb
 8600 root         20    0  188M   6672   3292  S   0.0   0.2   0:00.06 php-fpm: master
 8601 www-data     20    0  189M  10000   6436  S   0.0   0.2   0:00.00 php-fpm: pool w
 8602 www-data     20    0  189M  10764   7100  S   0.0   0.3   0:00.00 php-fpm: pool w
 9083 root         20    0  61592   1748     60  S   0.0   0.0   0:00.00 nginx: master p
 9084 www-data     20    0  62228   5712   3800  S   0.0   0.1   0:00.02 nginx: worker p
 9085 www-data     20    0  62228   5712   3800  S   0.0   0.1   0:00.00 nginx: worker p

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit

```

Dalam gambar `htop` untuk PHP worker bernama Linie, sistem terlihat memiliki beban yang sangat ringan, yang ditunjukkan oleh bar penggunaan CPU dan memori serta angka beban rata-rata yang rendah. Ini menunjukkan bahwa saat ini Linie tidak mengalami beban kerja yang berat dan dapat menangani lebih banyak permintaan jika perlu.

Proses yang terkait dengan Linie, khususnya proses PHP-fpm, memiliki penggunaan memori yang stabil dan penggunaan CPU yang minimal. Ini menunjukkan bahwa permintaan yang diarahkan ke worker ini mungkin tidak memerlukan pemrosesan intensif atau bahwa algoritma generic hash yang diimplementasikan pada load balancer telah berhasil menyeimbangkan beban permintaan secara merata di antara server yang tersedia. Dalam jangka panjang, pemantauan seperti ini penting untuk mengevaluasi kinerja dan efektivitas algoritma generic hash dalam menangani sesi klien dan permintaan, serta dalam menginformasikan keputusan terkait skalabilitas dan alokasi sumber daya.

A.4.2.3.htop pada Lugner

```

< te X Eisen X Lugner X > - X

1 [| 1.4%] Tasks: 11, 0 thr; 1 running
2 [| 2.8%] Load average: 0.00 0.04 0.09
Mem[|||||779M/3.82G] Uptime: 01:29:13
Swp[0K/2.00G]

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
 9084 www-data   20    0 62228  5616  3700  S   0.0   0.1   0:00.07 nginx: worker p
   1 root        20    0  4000   3324  2828  S   0.0   0.1   0:00.05 bash
 8606 root        20    0  188M   6416  3040  S   0.0   0.2   0:00.05 php-fpm: master
 8607 www-data   20    0  189M   9848  6284  S   0.0   0.2   0:00.02 php-fpm: pool w
 8608 www-data   20    0  189M  10416  6752  S   0.0   0.3   0:00.02 php-fpm: pool w
   83 root        20    0  2456   136    80  S   0.0   0.0   0:00.01 /tmp/gns3/bin/u
   86 root        20    0  2456    4      0  S   0.0   0.0   0:00.01 /gns3/bin/busyb
 9085 www-data   20    0 62228  5616  3700  S   0.0   0.1   0:00.01 nginx: worker p
   93 root        20    0  2456    4      0  S   0.0   0.0   0:00.00 /gns3/bin/busyb
 9083 root        20    0 61592  1736    52  S   0.0   0.0   0:00.00 nginx: master p
 9147 root        20    0  4788  3184  2760  R   0.0   0.1   0:00.00 htop

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit

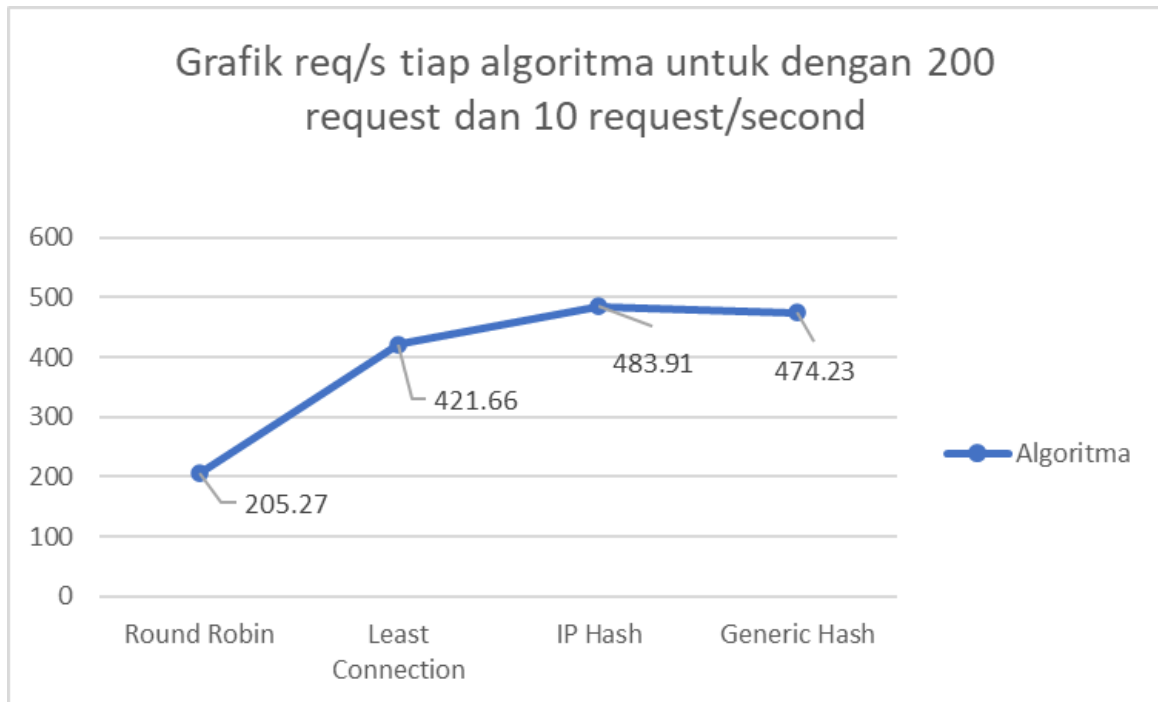
```

Pada tangkapan layar `htop` untuk PHP worker bernama Lugner, terlihat bahwa sistem tidak mengalami beban yang berat. Bar indikator CPU dan memori menunjukkan penggunaan yang relatif rendah, yang diperkuat oleh angka beban rata-rata sistem yang juga rendah. Proses PHP-fpm yang berkaitan dengan Lugner tampaknya memiliki penggunaan memori yang wajar dengan penggunaan CPU yang sangat rendah, mengindikasikan bahwa permintaan saat ini dihandle dengan lancar tanpa menyebabkan tekanan pada sumber daya sistem.

Dalam konteks algoritma generic hash pada load balancer, kondisi beban kerja yang ringan ini bisa menunjukkan bahwa distribusi permintaan telah dilakukan dengan baik, memanfaatkan kapasitas server tanpa membebani server tertentu secara tidak proporsional. Algoritma generic hash biasanya bertujuan untuk memastikan distribusi beban yang adil dan seragam di antara semua server yang tersedia, dan dari data yang terlihat, sepertinya algoritma ini berhasil menjalankan tugasnya dengan efektif untuk Lugner. Monitoring terus-menerus menggunakan `htop` penting untuk memastikan kinerja optimal dari load balancer dan untuk membuat penyesuaian jika terjadi perubahan dalam pola trafik atau beban kerja.

B. Grafik request per second untuk masing-masing algoritma

Berikut ini adalah grafik request per second untuk masing-masing algoritma berdasarkan Apache Benchmark yang telah dilakukan dengan 200 request dan 10 request/second.



C. Analisis

Dari grafik yang disajikan, terlihat bahwa algoritma **IP Hash** memiliki kinerja tertinggi dalam hal jumlah request per detik (req/s), dengan nilai puncak 483.91 req/s, yang menandakan efisiensi yang sangat baik dalam distribusi beban kerja. Ini diikuti oleh Generic Hash dengan 474.23 req/s, menunjukkan performa yang kuat. Di tempat ketiga adalah Least Connection dengan 421.66 req/s, dan terakhir Round Robin dengan performa terendah 205.27 req/s. Khususnya, IP Hash menunjukkan keunggulan karena algoritmanya menggunakan alamat IP klien sebagai kunci untuk mendistribusikan permintaan secara konsisten ke server tertentu, memfasilitasi manajemen sesi yang efisien dan caching yang efektif. Konsistensi ini mengurangi overhead dan meningkatkan throughput karena server dapat menyimpan dan mengakses data dari sesi klien yang sama dengan lebih cepat. Pendekatan ini juga mencegah overload pada server individu, menyebarluaskan beban kerja merata berdasarkan jumlah klien unik, yang ideal dalam lingkungan dengan banyak permintaan seragam, menghasilkan performa yang superior dibandingkan dengan algoritma lain dalam skenario ini.

Nomor 9

Dengan menggunakan algoritma Round Robin, lakukan testing dengan menggunakan 3 worker, 2 worker, dan 1 worker sebanyak 100 request dengan 10 request/second, kemudian tambahkan grafiknya pada grimoire.

Jawab:

A. Testing dengan 3 worker

Dalam rangka menilai kinerja dan keefektifan algoritma Round Robin pada load balancer, kami telah melakukan serangkaian pengujian menggunakan tiga worker PHP yang telah dikonfigurasi, yaitu Lawine, Linie, dan Lugner. Pengujian ini melibatkan distribusi seratus permintaan web secara merata dengan laju sepuluh permintaan per detik untuk memastikan bahwa algoritma Round Robin dapat dengan adil membagi beban kerja di antara ketiga worker tersebut. Hasil pengujian ini akan mengungkapkan bagaimana algoritma Round Robin mengelola permintaan yang datang dan seberapa baik setiap worker PHP memproses permintaan tersebut, memberikan data berharga mengenai latensi, throughput, dan kemampuan setiap worker dalam menangani beban secara simultan.

A.1. Apache Benchmark

Berikut ini adalah hasil testing Apache Benchmark untuk algoritma load balancer Round Robin dengan 3 worker:

```
root@Revolte:~# ab -n 100 -c 10 http://www.granz.channel.B04.com/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

Benchmarking www.granz.channel.B04.com (be patient).....done

```
Server Software:      nginx/1.14.2
Server Hostname:      www.granz.channel.B04.com
Server Port:          80
```

```
Document Path:        /
Document Length:      606 bytes
```

```
Concurrency Level:    10
Time taken for tests:  0.502 seconds
Complete requests:    100
Failed requests:       33
  (Connect: 0, Receive: 0, Length: 33, Exceptions: 0)
Total transferred:    74267 bytes
HTML transferred:     60567 bytes
Requests per second:  199.14 [#/sec] (mean)
Time per request:     50.217 [ms] (mean)
Time per request:     5.022 [ms] (mean, across all concurrent requests)
Transfer rate:        144.43 [Kbytes/sec] received
```

```
Connection Times (ms)
      min mean[+/-sd] median max
Connect:    3  11  5.7   10   35
```

Processing:	17	36	17.1	30	87
Waiting:	17	36	16.9	30	85
Total:	25	47	20.7	39	109

Percentage of the requests served within a certain time (ms)

50%	39
66%	42
75%	48
80%	63
90%	88
95%	98
98%	104
99%	109
100%	109 (longest request)

A.2. htop pada masing-masing PHP worker

Sebagai bagian dari evaluasi performa load balancing menggunakan algoritma Round Robin, kami telah mengamati penggunaan sumber daya pada setiap worker PHP—Lawine, Linie, dan Lugner—melalui tool `htop`. Pemantauan ini memungkinkan kita untuk secara visual memeriksa dan menganalisis penggunaan CPU dan memori secara real-time, yang merefleksikan seberapa efisien masing-masing worker dapat menangani permintaan yang diterima. Data ini penting untuk memastikan bahwa tidak ada worker yang kelebihan beban atau tidak cukup dimanfaatkan, yang dapat mengindikasikan perlunya penyesuaian pada strategi distribusi beban atau konfigurasi server untuk mencapai kinerja optimal dalam lingkungan produksi.

A.2.1. htop pada Lawine


```

1  [|||] 7.1%] Tasks: 11, 0 thr; 1 running
2  [||] 4.0%] Load average: 0.29 0.37 0.52
Mem[|||||]764M/3.82G] Uptime: 00:49:38
Swp[ 0K/2.00G]

  PID USER   PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---
 112 root    20   0  2456     4     0  S   0.0  0.0   0:00.00 /gns3/bin/busyb
 104 root    20   0  2456     4     0  S   0.0  0.0   0:00.01 /gns3/bin/busyb
 101 root    20   0  2456    132    80  S   0.0  0.0   0:00.01 /tmp/gns3/bin/u
   1 root    20   0  4000   3332  2824  S   0.0  0.1   0:00.10 bash
 9102 root    20   0  4788   3264  2840  R   0.7  0.1   0:00.03 htop
 8976 root    20   0 61592   1736    52  S   0.0  0.0   0:00.00 nginx: master p
 8977 www-data 20   0 62228   6784  4736  S   0.0  0.2   0:00.05 nginx: worker p
 8978 www-data 20   0 62228   6776  4736  S   0.0  0.2   0:00.33 nginx: worker p
 8498 root    20   0 188M    6472  3092  S   0.0  0.2   0:00.43 php-fpm: master
 8499 www-data 20   0 189M    9792  6216  S   0.0  0.2   0:00.12 php-fpm: pool w
 8500 www-data 20   0 189M   10512  6844  S   0.0  0.3   0:00.11 php-fpm: pool w

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit

```

Dalam gambar `htop` untuk worker PHP bernama Lawine, kita dapat melihat bahwa penggunaan sumber daya sistem cukup rendah. Proses dengan PID 8976, 8977, 8498, 8499, dan 8500, yang kemungkinan adalah instance dari PHP-FPM, menunjukkan penggunaan memori yang moderat namun dengan penggunaan CPU yang sangat rendah, mengindikasikan bahwa mereka tidak sedang menerima beban kerja yang berat saat ini. Khususnya, proses PHP-FPM dengan PID 8500 menunjukkan sedikit lebih banyak aktivitas dibandingkan dengan yang lain, yang mungkin sedang menangani permintaan yang lebih kompleks atau berjumlah lebih banyak. Namun, secara keseluruhan, beban sistem yang rendah ini menunjukkan bahwa Lawine berada dalam keadaan yang stabil dan mampu menangani permintaan tambahan yang mungkin diterima dari load balancer.

A.2.2. htop pada Linie

```

1  [|||||] 6.5%] Tasks: 11, 0 thr; 1 running
2  [|||||] 10.2%] Load average: 0.22 0.37 0.52
Mem[|||||] 764M/3.82G] Uptime: 00:49:09
Swp[|] 0K/2.00G]

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  9087 root        20    0  4788   3304  2880  R   1.3   0.1   0:00.03 htop
    1 root        20    0  4000   3188  2688  S   0.0   0.1   0:00.14 bash
   85 root        20    0  2456    140    80  S   0.0   0.0   0:00.02 /tmp/gns3/bin/u
   92 root        20    0  2456     4     0  S   0.0   0.0   0:00.02 /gns3/bin/busyb
   99 root        20    0  2456     4     0  S   0.0   0.0   0:00.00 /gns3/bin/busyb
  8497 root        20    0  188M   6664  3288  S   0.0   0.2   0:00.25 php-fpm: master
  8498 www-data    20    0  189M   9892  6320  S   0.0   0.2   0:00.11 php-fpm: pool w
  8499 www-data    20    0  189M  10712  7048  S   0.0   0.3   0:00.12 php-fpm: pool w
  8982 root        20    0  61592  1752    72  S   0.0   0.0   0:00.00 nginx: master p
  8983 www-data    20    0  62228  5680  3772  S   0.0   0.1   0:00.31 nginx: worker p
  8984 www-data    20    0  62228  6804  4756  S   0.0   0.2   0:00.02 nginx: worker p

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit

```

Pada gambar `htop` untuk worker PHP bernama Linie, terlihat bahwa beban kerja terhadap sumber daya sistem sedikit lebih tinggi dibandingkan dengan Lawine. Beberapa proses PHP-FPM, terutama dengan PID 8497 dan 8499, memperlihatkan aktivitas CPU yang kecil, yang menandakan adanya pemrosesan permintaan. Meski begitu, penggunaan memori masih pada tingkat yang wajar dan tidak ada indikasi penggunaan CPU yang berlebihan. Ini menunjukkan bahwa Linie beroperasi dengan efektif dalam kondisi saat ini dan mampu menangani beban kerja yang diberikan oleh load balancer dengan algoritma Round Robin. Ketersediaan sumber daya yang memadai ini mengindikasikan bahwa Linie memiliki kapasitas untuk menampung peningkatan beban kerja jika diperlukan.

A.2.3. htop pada Lugner

root@Revolte:~# ab -n 100 -c 10 http://www.granz.channel.B04.com/
This is ApacheBench, Version 2.3 <\$Revision: 1843412 \$>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking www.granz.channel.B04.com (be patient).....done

Server Software: nginx/1.14.2
Server Hostname: www.granz.channel.B04.com
Server Port: 80

Document Path: /
Document Length: 606 bytes

Concurrency Level: 10
Time taken for tests: 0.419 seconds
Complete requests: 100
Failed requests: 50
(Connect: 0, Receive: 0, Length: 50, Exceptions: 0)
Total transferred: 74250 bytes
HTML transferred: 60550 bytes
Requests per second: 238.62 [#/sec] (mean)
Time per request: 41.907 [ms] (mean)
Time per request: 4.191 [ms] (mean, across all concurrent requests)
Transfer rate: 173.02 [Kbytes/sec] received

Connection Times (ms)

	min	mean	+/-sd	median	max
Connect:	1	10	3.7	10	22
Processing:	10	30	5.8	30	45
Waiting:	9	29	5.8	29	43
Total:	11	40	6.8	40	54

Percentage of the requests served within a certain time (ms)

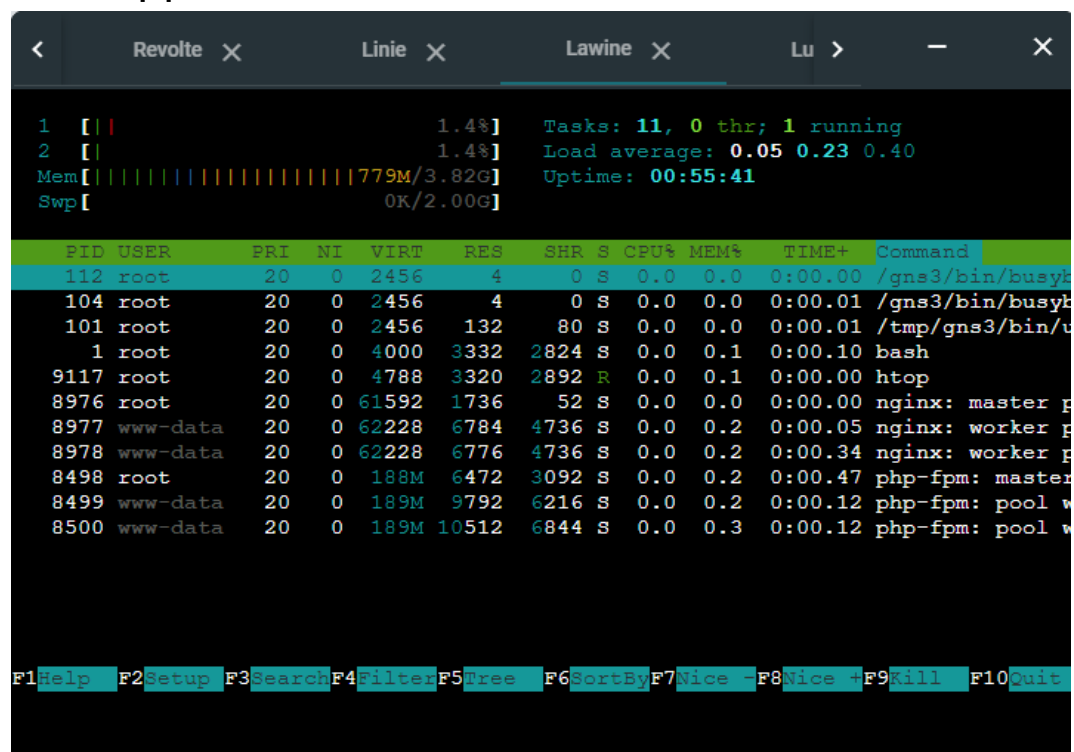
50%	40
66%	43
75%	44
80%	45
90%	50
95%	52
98%	54
99%	54

100% 54 (longest request)

B.2. htop pada masing-masing PHP worker

Dalam tahap ini, kita mengarahkan perhatian pada analisis penggunaan sumber daya oleh worker PHP Lawine dan Linie melalui `htop`, setelah mengonfigurasi load balancer untuk hanya menggunakan kedua worker ini dengan algoritma Round Robin. Dengan Lugner tidak lagi berpartisipasi, kita dapat mengukur dampak dari pengurangan kapasitas pada performa dan efisiensi sistem. Pengamatan ini penting untuk memahami bagaimana setiap worker menanggapi beban yang lebih besar dan bagaimana hal ini mempengaruhi distribusi permintaan antar worker, yang dapat menginformasikan keputusan skalabilitas dan kebijakan failover dalam arsitektur load balancing kami.

B.2.1. htop pada Lawine



```
<  Revolte  X  Linie  X  Lawine  X  Lu  >  -  X

1  [||  1.4%]  Tasks: 11, 0 thr; 1 running
2  [|  1.4%]  Load average: 0.05 0.23 0.40
Mem[||||| 779M/3.82G]  Uptime: 00:55:41
Swp[ 0K/2.00G]

PID USER  PRI  NI  VIRT  RES  SHR S  CPU% MEM%  TIME+  Command
112 root    20   0  2456   4    0 s  0.0  0.0  0:00.00 /gns3/bin/busyb
104 root    20   0  2456   4    0 s  0.0  0.0  0:00.01 /gns3/bin/busyb
101 root    20   0  2456  132   80 s  0.0  0.0  0:00.01 /tmp/gns3/bin/v
1 root    20   0  4000 3332 2824 s  0.0  0.1  0:00.10 bash
9117 root    20   0  4788 3320 2892 R  0.0  0.1  0:00.00 htop
8976 root    20   0 61592 1736  52 s  0.0  0.0  0:00.00 nginx: master p
8977 www-data 20   0 62228 6784 4736 s  0.0  0.2  0:00.05 nginx: worker p
8978 www-data 20   0 62228 6776 4736 s  0.0  0.2  0:00.34 nginx: worker p
8498 root    20   0 188M  6472 3092 s  0.0  0.2  0:00.47 php-fpm: master
8499 www-data 20   0 189M  9792 6216 s  0.0  0.2  0:00.12 php-fpm: pool w
8500 www-data 20   0 189M 10512 6844 s  0.0  0.3  0:00.12 php-fpm: pool w

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice + F9Kill F10Quit
```

Pada pemantauan penggunaan sumber daya oleh worker PHP Lawine melalui `htop`, kita dapat melihat bahwa ada beberapa proses PHP-FPM yang aktif, yang ditandai dengan PID 8976, 8977, 8498, 8499, dan 8500. Penggunaan CPU oleh proses ini relatif rendah, menunjukkan bahwa tidak ada permintaan yang sangat membebani pada saat ini, meskipun ada sedikit aktivitas yang terlihat, kemungkinan sebagai hasil dari permintaan web yang sedang diproses. Penggunaan memori stabil dan tidak mencapai batas yang mengkhawatirkan, menandakan bahwa Lawine dalam kondisi yang baik untuk menangani beban kerja yang diberikan oleh algoritma Round Robin di lingkungan dengan dua worker. Ini menunjukkan keandalan dan kemampuan Lawine untuk beroperasi secara efisien dalam kondisi pengurangan jumlah worker.

B.2.2. htop pada Linie

```
<  Revolte  X      Linie  X      Lawine  X      Lu  >  -  X

1  [          0.0%]  Tasks: 11, 0 thr; 1 running
2  [          0.0%]  Load average: 0.03 0.20 0.38
Mem[|||||779M/3.82G]  Uptime: 00:56:24
Swp[          0K/2.00G]

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---  ---      -  --  -
9109 root        20   0  4788   3300  2880  R   0.0   0.1   0:00.00  htop
   1 root        20   0  4000   3188  2688  S   0.0   0.1   0:00.14  bash
  85 root        20   0  2456    140    80  S   0.0   0.0   0:00.02  /tmp/gns3/bin/u
  92 root        20   0  2456     4     0  S   0.0   0.0   0:00.02  /gns3/bin/busyb
  99 root        20   0  2456     4     0  S   0.0   0.0   0:00.00  /gns3/bin/busyb
8497 root        20   0  188M   6664  3288  S   0.0   0.2   0:00.30  php-fpm: master
8498 www-data    20   0  189M   9892  6320  S   0.0   0.2   0:00.12  php-fpm: pool w
8499 www-data    20   0  189M  10712  7048  S   0.0   0.3   0:00.12  php-fpm: pool w
8982 root        20   0  61592  1752    72  S   0.0   0.0   0:00.00  nginx: master p
8983 www-data    20   0  62228   5680  3772  S   0.0   0.1   0:00.32  nginx: worker p
8984 www-data    20   0  62228   6804  4756  S   0.0   0.2   0:00.03  nginx: worker p

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice +F8Nice +F9Kill F10Quit
```

Snapshot `htop` untuk worker PHP Linie menunjukkan bahwa server ini juga sedang menjalankan beberapa proses PHP-FPM, khususnya proses dengan PID 8497, 8499, 8982, 8983, dan 8984, dengan penggunaan memori yang wajar dan beban CPU yang rendah. Tidak adanya tekanan yang signifikan pada CPU menunjukkan bahwa Linie mampu menangani permintaan saat ini dengan efisien, sesuai dengan ekspektasi dari konfigurasi load balancer Round Robin yang beroperasi hanya dengan dua worker. Kondisi ini menandakan bahwa Linie, bersama dengan Lawine, memberikan kontribusi yang seimbang dalam mengelola beban kerja yang diberikan, menunjukkan kapasitas dan kemampuan mereka untuk mempertahankan performa yang stabil meskipun dengan satu worker yang tidak aktif.

C. Testing dengan 1 worker

Dalam fase terakhir pengujian, kami mengisolasi worker PHP Lawine sebagai satu-satunya unit pemrosesan dalam konfigurasi load balancer Round Robin kami, dengan sengaja mematikan worker Linie dan Lugner untuk menilai kinerja sistem dalam skenario single-worker. Tujuan dari pengujian ini adalah untuk menguji batas kemampuan dan daya tahan Lawine ketika dituntut untuk menangani seluruh beban kerja sendirian. Ini akan memberikan wawasan penting tentang performa maksimal yang bisa dicapai oleh satu worker dan akan menunjukkan bagaimana Lawine mengelola permintaan yang bertambah tanpa dukungan dari worker lain dalam infrastruktur kami.

C.1. Apache Benchmark

Berikut ini adalah hasil testing Apache Benchmark untuk algoritma load balancer Round Robin dengan 1 worker:

```
root@Revolte:~# ab -n 100 -c 10 http://www.granz.channel.B04.com/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

Benchmarking www.granz.channel.B04.com (be patient).....done

Server Software: nginx/1.14.2
Server Hostname: www.granz.channel.B04.com
Server Port: 80

Document Path: /
Document Length: 606 bytes

Concurrency Level: 10
Time taken for tests: 0.305 seconds
Complete requests: 100
Failed requests: 0
Total transferred: 74300 bytes
HTML transferred: 60600 bytes
Requests per second: 328.33 [#/sec] (mean)
Time per request: 30.457 [ms] (mean)
Time per request: 3.046 [ms] (mean, across all concurrent requests)
Transfer rate: 238.23 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	1	7 5.2	6	30
Processing:	4	22 7.1	21	54
Waiting:	3	22 7.1	20	54
Total:	5	29 9.6	27	58

Percentage of the requests served within a certain time (ms)

50%	27
66%	29
75%	31
80%	31
90%	49
95%	54
98%	58
99%	58
100%	58 (longest request)

C.2. htop pada PHP worker

Dalam evaluasi lebih lanjut terhadap kinerja load balancing dengan hanya satu worker PHP, kita memfokuskan pada pengamatan penggunaan sumber daya oleh Lawine melalui `htop`. Analisis ini akan memberikan wawasan tentang bagaimana Lawine, sebagai satu-satunya penerima beban kerja, mengelola sumber daya yang dialokasikan dan mempertahankan kinerja sistem ketika Linie dan Lugner tidak aktif. Kondisi ini penting untuk menilai kapasitas maksimum dan efisiensi Lawine dalam menghadapi permintaan secara independen dalam skenario load balancer Round Robin.

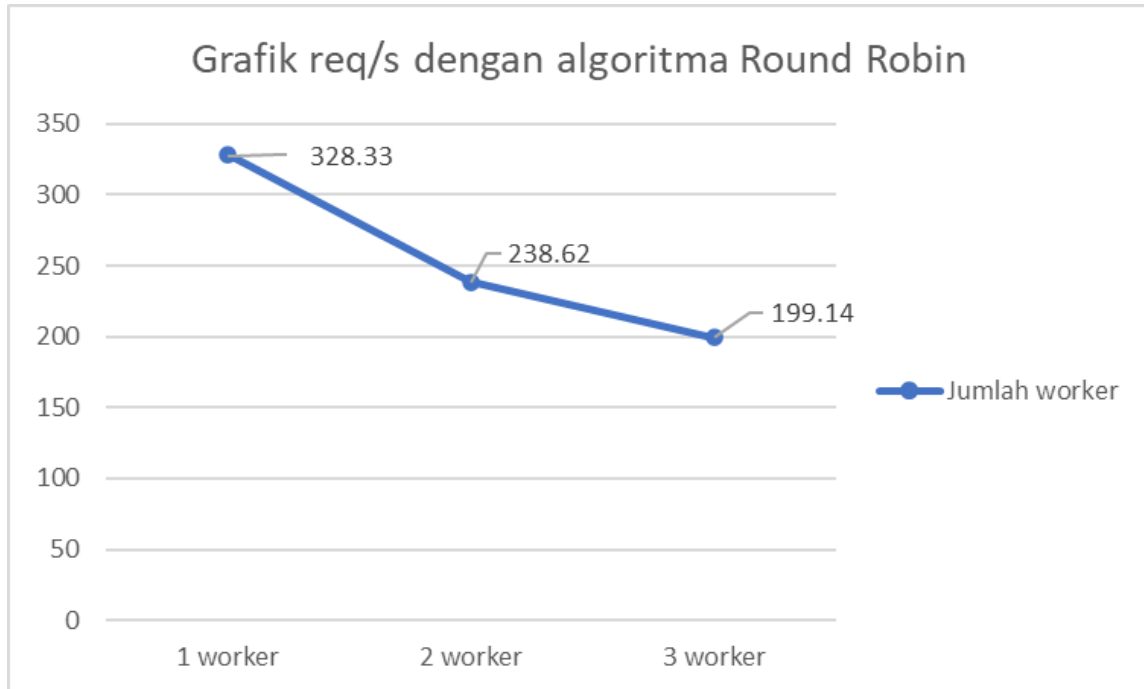
C.2.1. htop pada Lawine

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
112	root	20	0	2456	4	0	S	0.0	0.0	0:00.00	/gns3/bin/busybox
104	root	20	0	2456	4	0	S	0.0	0.0	0:00.01	/gns3/bin/busybox
101	root	20	0	2456	132	80	S	0.0	0.0	0:00.02	/tmp/gns3/bin/udev
1	root	20	0	4000	3332	2824	S	0.0	0.1	0:00.10	bash
9132	root	20	0	4788	3224	2796	R	0.0	0.1	0:00.00	htop
8976	root	20	0	61592	1736	52	S	0.0	0.0	0:00.00	nginx: master process
8977	www-data	20	0	62228	6784	4736	S	0.0	0.2	0:00.05	nginx: worker process
8978	www-data	20	0	62228	6776	4736	S	0.0	0.2	0:00.38	nginx: worker process
8498	root	20	0	188M	6472	3092	S	0.0	0.2	0:00.50	php-fpm: master process
8499	www-data	20	0	189M	9792	6216	S	0.0	0.2	0:00.13	php-fpm: pool worker
8500	www-data	20	0	189M	10512	6844	S	0.0	0.3	0:00.13	php-fpm: pool worker

Dari tangkapan layar, kita dapat melihat bahwa Lawine sedang mengelola beberapa proses PHP-FPM, dengan proses-proses tersebut menunjukkan penggunaan memori yang stabil dan beban CPU yang relatif rendah. Hal ini menunjukkan bahwa meskipun Linie dan Lugner telah dinonaktifkan, Lawine tampak mampu menangani beban kerja secara mandiri tanpa tanda-tanda stres berlebihan pada sumber daya sistem. Pengamatan ini krusial untuk memahami ketahanan dan efisiensi satu worker dalam infrastruktur load balancing kami ketika menghadapi seluruh beban permintaan.

D. Grafik request/second untuk algoritma round robin dengan variasi jumlah worker

Berikut ini adalah grafik request per second untuk masing-masing variasi jumlah worker berdasarkan Apache Benchmark yang telah dilakukan dengan 100 request dan 10 request/second



E. Analisis

Dari data yang disajikan, dapat disimpulkan bahwa ada penurunan performa seiring dengan penambahan jumlah worker. Dengan satu worker, sistem dapat menangani 328.33 req/s, yang merupakan performa terbaik yang tercatat. Namun, ketika jumlah worker meningkat menjadi dua, terjadi penurunan performa menjadi 238.62 req/s. Penurunan performa ini berlanjut dengan penambahan worker ketiga, di mana sistem hanya mampu menangani 199.14 req/s.

Penurunan ini dapat disebabkan oleh beberapa faktor. Pertama, Round Robin mendistribusikan beban secara merata tanpa mempertimbangkan beban saat ini dari setiap worker. Oleh karena itu, seiring dengan penambahan worker, mungkin terjadi ketidakseimbangan beban di mana beberapa worker mungkin mengalami underutilization sementara yang lain mungkin overwhelmed. Kedua, overhead komunikasi antar-worker dapat meningkat dengan penambahan worker, yang menyebabkan penurunan efisiensi keseluruhan. Ketiga, konteks switching antara worker juga bisa menyebabkan latensi tambahan, terutama jika sistem tidak dirancang untuk skala horizontal yang efektif. Hasil ini menggarisbawahi pentingnya penyesuaian kapasitas infrastruktur dengan beban kerja yang diharapkan untuk mencapai optimalisasi beban yang efisien.

Nomor 15

Riegel Channel memiliki beberapa endpoint yang harus ditesting sebanyak 100 request dengan 10 request/second. Tambahkan response dan hasil testing pada grimoire POST /auth/register.

Jawab:

Dalam rangka meningkatkan kualitas layanan dan keandalan sistem pada Riegel Channel, kami telah melakukan serangkaian pengujian menggunakan Apache Benchmark dari klien bernama Sein. Pengujian ini difokuskan pada endpoint /auth/register untuk memastikan bahwa sistem dapat menangani 100 request dengan kecepatan 10 request per detik. Pengujian bertujuan untuk mengukur performa dan kestabilan sistem dalam menangani permintaan pendaftaran pengguna, yang merupakan aspek kritis dari pengalaman pengguna. Hasil dari pengujian ini akan memberikan data penting mengenai latensi, throughput, dan kemampuan server untuk memproses request POST secara efektif di bawah beban yang simulasi.

A. Apache Benchmark

Berikut ini adalah hasil testing Apache Benchmark untuk 100 request dengan 10 request/second pada POST /auth/register:

```
root@Sein:~# ab -n 100 -c 10 -p register.json -T application/json
http://192.180.4.2:8002/api/auth/register
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

Benchmarking 192.180.4.2 (be patient).....done

```
Server Software:      nginx/1.14.2
Server Hostname:      192.180.4.2
Server Port:          8002
```

```
Document Path:        /api/auth/register
Document Length:       95 bytes
```

```
Concurrency Level:     10
Time taken for tests:   0.861 seconds
Complete requests:     100
Failed requests:        30
  (Connect: 0, Receive: 0, Length: 30, Exceptions: 0)
Non-2xx responses:     100
Total transferred:     236030 bytes
Total body sent:        21700
HTML transferred:      205220 bytes
Requests per second:   116.13 [#/sec] (mean)
```

Time per request: 86.112 [ms] (mean)
Time per request: 8.611 [ms] (mean, across all concurrent requests)
Transfer rate: 267.67 [Kbytes/sec] received
24.61 kb/s sent
292.28 kb/s total

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	0 0.4	0	2
Processing:	23	78 20.1	74	148
Waiting:	23	78 19.9	74	148
Total:	25	79 20.0	74	149

Percentage of the requests served within a certain time (ms)

50%	74
66%	82
75%	85
80%	88
90%	106
95%	122
98%	140
99%	149
100%	149 (longest request)

B. htop pada Laravel worker (Flamme)

The screenshot shows the htop interface with the following system statistics at the top:

- Tasks: 12, 0 thr; 1 running
- Load average: 0.11 0.30 0.32
- Uptime: 01:43:50
- Mem: 810M/3.82G
- Swp: 268K/2.00G

Below the statistics is a table of running processes:

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1	root	20	0	4000	3212	2708	S	0.0	0.1	0:00.08	bash
521	root	20	0	2456	4	0	S	0.0	0.0	0:00.01	/gns3/bin/busybo
549	root	20	0	2456	4	0	S	0.0	0.0	0:00.00	/gns3/bin/busybo
970	root	20	0	2456	140	80	S	0.0	0.0	0:00.01	/tmp/gns3/bin/u
14434	root	20	0	238M	10728	5108	S	0.0	0.3	0:00.03	php-fpm: master
14435	www-data	20	0	241M	36212	26592	S	0.0	0.9	0:00.61	php-fpm: pool w
14436	www-data	20	0	241M	37140	27028	S	0.0	0.9	0:00.93	php-fpm: pool w
14471	root	20	0	61592	1740	56	S	0.0	0.0	0:00.00	nginx: master p
14472	www-data	20	0	62300	6732	4596	S	0.0	0.2	0:00.01	nginx: worker p
14473	www-data	20	0	62360	6636	4596	S	0.0	0.2	0:00.01	nginx: worker p
14506	www-data	20	0	240M	33908	24684	S	0.0	0.8	0:00.11	php-fpm: pool w
14507	root	20	0	4788	3160	2728	R	0.0	0.1	0:00.00	htop

At the bottom, there is a menu bar with function keys: F1 Help, F2 Setup, F3 Search, F4 Filter, F5 Tree, F6 SortBy, F7 Nice -, F8 Nice +, F9 Kill, F10 Quit.

Pada tangkapan layar `htop` yang ditampilkan, Flamme, yang merupakan worker berbasis Laravel, tampak sedang memproses permintaan setelah serangkaian tes dilakukan dari Sein. Dari informasi penggunaan sumber daya, kita bisa melihat bahwa ada beberapa proses PHP-FPM yang aktif, khususnya dengan PID 14435 dan 14436, yang menggunakan sebagian besar CPU, menandakan bahwa worker ini sedang sibuk menangani beban kerja yang dihasilkan dari tes. Namun, penggunaan memori masih pada level yang dapat dikelola, menunjukkan bahwa meskipun sistem sedang diuji, tidak ada kelebihan beban yang berarti. Kedua sistem, Sein dan Flamme, berada pada switch yang sama, menunjukkan bahwa tes dilakukan dalam jaringan lokal yang sama, yang memungkinkan komunikasi yang cepat dan efisien antar layanan.

Nomor 16

Riegel Channel memiliki beberapa endpoint yang harus ditesting sebanyak 100 request dengan 10 request/second. Tambahkan response dan hasil testing pada grimoire POST /auth/login.

Jawab:

Untuk memastikan bahwa endpoint login Riegel Channel dapat menangani beban yang intensif, kami telah melakukan serangkaian uji coba menggunakan Apache Benchmark. Pengujian ini dirancang untuk mensimulasikan kondisi lalu lintas yang tinggi dengan mengirim 100 permintaan POST ke /auth/login dengan rate 10 permintaan per detik. Hasil dari pengujian ini

akan menentukan stabilitas dan keandalan endpoint saat menghadapi lonjakan permintaan yang secara realistis dapat terjadi dalam produksi. Data yang dikumpulkan akan memberikan wawasan berharga untuk optimisasi lebih lanjut dan peningkatan infrastruktur.

A. Apache Benchmark

Berikut ini adalah hasil testing Apache Benchmark untuk 100 request dengan 10 request/second pada POST /auth/login:

```
root@Sein:~# ab -n 100 -c 10 -p login.json -T application/json
http://192.180.4.2:8002/api/auth/login
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

Benchmarking 192.180.4.2 (be patient).....done

Server Software: nginx/1.14.2
Server Hostname: 192.180.4.2
Server Port: 8002

Document Path: /api/auth/login
Document Length: 95 bytes

Concurrency Level: 10
Time taken for tests: 0.740 seconds
Complete requests: 100
Failed requests: 38
(Connect: 0, Receive: 0, Length: 38, Exceptions: 0)
Non-2xx responses: 100
Total transferred: 288652 bytes
Total body sent: 21400
HTML transferred: 257412 bytes
Requests per second: 135.14 [#/sec] (mean)
Time per request: 73.999 [ms] (mean)
Time per request: 7.400 [ms] (mean, across all concurrent requests)
Transfer rate: 380.93 [Kbytes/sec] received
28.24 kb/s sent
409.18 kb/s total

Connection Times (ms)

	min	mean	mean[+/-sd]	median	max
Connect:	0	1	0.6	0	4
Processing:	11	69	20.4	69	124
Waiting:	11	68	20.2	69	124

Total: 13 69 20.2 69 124

WARNING: The median and mean for the initial connection time are not within a normal deviation

These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)

50% 69
66% 77
75% 81
80% 83
90% 93
95% 105
98% 118
99% 124
100% 124 (longest request)

B. htop pada Laravel worker (Flamme)

```
< X Denken X Flamme X Sein X > - X

1 [| 0.7%] Tasks: 12, 0 thr; 1 running
2 [| 0.0%] Load average: 0.37 0.30 0.30
Mem[|||||||810M/3.82G] Uptime: 01:47:42
Swp[| 268K/2.00G]

  PID USER   PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---
    1 root     20    0  4000   3212  2708  S   0.0  0.1   0:00.08 bash
    521 root     20    0  2456     4     0  S   0.0  0.0   0:00.01 /gns3/bin/busyb
    549 root     20    0  2456     4     0  S   0.0  0.0   0:00.00 /gns3/bin/busyb
    970 root     20    0  2456    140    80  S   0.0  0.0   0:00.01 /tmp/gns3/bin/u
  14434 root     20    0  238M 10728   5108  S   0.0  0.3   0:00.04 php-fpm: master
  14435 www-data  20    0  241M 36224  26604  S   0.0  0.9   0:01.07 php-fpm: pool w
  14436 www-data  20    0  241M 37152  27040  S   0.0  0.9   0:01.33 php-fpm: pool w
  14471 root     20    0  61592  1740     56  S   0.0  0.0   0:00.00 nginx: master p
  14472 www-data  20    0  62300  6732   4596  S   0.0  0.2   0:00.02 nginx: worker p
  14473 www-data  20    0  62360  6636   4596  S   0.0  0.2   0:00.05 nginx: worker p
  14506 www-data  20    0  241M 35256  25788  S   0.0  0.9   0:00.48 php-fpm: pool w
  14515 root     20    0  4788   3264   2840  R   0.0  0.1   0:00.00 htop

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice -F9Kill F10Quit
```

Gambar `htop` yang disajikan menunjukkan Flamme, yang merupakan worker berbasis Laravel, setelah menjalankan serangkaian tes POST ke endpoint /auth/login dari Sein. Dari output `htop`, terlihat bahwa ada beberapa proses PHP-FPM yang aktif, dengan PID 14435 dan 14436, menunjukkan tingkat penggunaan CPU yang signifikan, yang mengindikasikan bahwa Flamme sedang memproses permintaan yang dihasilkan oleh tes. Meskipun ada kenaikan penggunaan CPU, tidak ada satu proses pun yang tampaknya mengalami kelebihan beban, dan memori sistem masih dalam kondisi yang

baik, menandakan bahwa worker mampu mengatasi beban tes tersebut. Sein dan Flamme berada di switch yang sama, menandakan bahwa tes dilakukan dalam jaringan lokal yang sama, yang memungkinkan pemeriksaan kinerja yang cepat dan tepat dalam skenario yang mendekati produksi riil.

Nomor 17

Riegel Channel memiliki beberapa endpoint yang harus ditesting sebanyak 100 request dengan 10 request/second. Tambahkan response dan hasil testing pada grimoire GET /me.

Jawab:

A. Apache Benchmark

Berikut ini adalah hasil testing Apache Benchmark untuk 100 request dengan 10 request/second pada GET /me:

```
root@Sein:~# ab -n 100 -c 10 -H "Authorization: Bearer $token"
http://192.180.4.2:8002/api/me
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

Benchmarking 192.180.4.2 (be patient).....done

```
Server Software:      nginx/1.14.2
Server Hostname:      192.180.4.2
Server Port:          8002
```

```
Document Path:        /api/me
Document Length:       42 bytes
```

```
Concurrency Level:     10
Time taken for tests:   0.605 seconds
Complete requests:     100
Failed requests:        36
  (Connect: 0, Receive: 0, Length: 36, Exceptions: 0)
Non-2xx responses:     100
Total transferred:     272104 bytes
HTML transferred:      240972 bytes
Requests per second:   165.32 [#/sec] (mean)
Time per request:      60.489 [ms] (mean)
Time per request:      6.049 [ms] (mean, across all concurrent requests)
```

Transfer rate: 439.30 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	0 0.2	0	1
Processing:	12	56 11.2	57	76
Waiting:	12	56 11.3	57	75
Total:	13	56 11.2	58	77

Percentage of the requests served within a certain time (ms)

50%	58
66%	61
75%	65
80%	66
90%	68
95%	72
98%	72
99%	77
100%	77 (longest request)

B. htop pada Laravel worker (Flamme)

The screenshot shows the htop terminal window with the following content:

```
< X Denken X Flamme X Sein X > - X

1  [|] 0.7% Tasks: 12, 0 thr; 1 running
2  [|] 2.1% Load average: 0.00 0.08 0.19
Mem[|||||||809M/3.82G] Uptime: 01:53:53
Swp[|] 268K/2.00G

  PID USER   PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---  ---
    1 root     20    0  4000   3212  2708  S   0.0  0.1   0:00.08 bash
  521 root     20    0  2456     4     0  S   0.0  0.0   0:00.01 /gns3/bin/busyb
  549 root     20    0  2456     4     0  S   0.0  0.0   0:00.00 /gns3/bin/busyb
  970 root     20    0  2456   140    80  S   0.0  0.0   0:00.01 /tmp/gns3/bin/u
14434 root     20    0  238M 10728  5108  S   0.0  0.3   0:00.05 php-fpm: master
14435 www-data 20    0  241M 36248 26628  S   0.0  0.9   0:01.42 php-fpm: pool w
14436 www-data 20    0  241M 37176 27064  S   0.0  0.9   0:01.70 php-fpm: pool w
14471 root     20    0 61592  1740    56  S   0.0  0.0   0:00.00 nginx: master p
14472 www-data 20    0 62300   6732  4596  S   0.0  0.2   0:00.02 nginx: worker p
14473 www-data 20    0 62360   6636  4596  S   0.0  0.2   0:00.08 nginx: worker p
14506 www-data 20    0  241M 35344 25812  S   0.0  0.9   0:00.79 php-fpm: pool w
14523 root     20    0  4788   3232  2812  R   0.0  0.1   0:00.00 htop

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit
```


Gambar htop menunjukkan aktivitas sistem dari server yang menjalankan Laravel, diidentifikasi sebagai "Flamme," setelah menjalankan tes beban dengan 100 request GET ke endpoint /me dengan laju 10 request per detik. Terlihat bahwa CPU dan memori memiliki beban yang signifikan, dengan beberapa proses 'php-fpm' seperti yang diindikasikan oleh PID 14435 dan 14436 menunjukkan penggunaan CPU yang meningkat. Meskipun demikian, penggunaan memori masih terjaga, menunjukkan bahwa server masih stabil meskipun di bawah beban tes. Penggunaan htop ini membantu dalam memonitor sumber daya sistem dan memastikan bahwa aplikasi Laravel dapat mengelola lalu lintas yang padat tanpa overloading, yang krusial untuk memastikan kinerja yang optimal dalam lingkungan produksi.

Nomor 18

Untuk memastikan ketiganya bekerja sama secara adil untuk mengatur Riegel Channel maka implementasikan Proxy Bind pada Eisen untuk mengaitkan IP dari Frieren, Flamme, dan Fern.

Jawab:

Berikut ini adalah hasil Apache Benchmark setelah melakukan implementasi Proxy Bind pada Eisen yang dilakukan testing pada client Sein:

```
root@Sein:~# ab -n 100 -c 10 -p login.json -T application/json
http://www.riegel.canyon.B04.com/api/auth/login
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

Benchmarking www.riegel.canyon.B04.com (be patient).....done

Server Software: nginx/1.14.2
Server Hostname: www.riegel.canyon.B04.com
Server Port: 80

Document Path: /api/auth/login
Document Length: 95 bytes

Concurrency Level: 10
Time taken for tests: 0.834 seconds
Complete requests: 100
Failed requests: 36
(Connect: 0, Receive: 0, Length: 36, Exceptions: 0)
Non-2xx responses: 100
Total transferred: 275497 bytes
Total body sent: 22300

HTML transferred: 244364 bytes
Requests per second: 119.95 [#/sec] (mean)
Time per request: 83.370 [ms] (mean)
Time per request: 8.337 [ms] (mean, across all concurrent requests)
Transfer rate: 322.70 [Kbytes/sec] received
26.12 kb/s sent
348.83 kb/s total

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	1 0.6	1	3
Processing:	21	77 16.9	77	150
Waiting:	21	77 16.6	76	149
Total:	23	78 16.8	78	150

Percentage of the requests served within a certain time (ms)

50%	78
66%	82
75%	86
80%	88
90%	98
95%	107
98%	111
99%	150
100%	150 (longest request)

Nomor 19

Untuk meningkatkan performa dari Worker, coba implementasikan PHP-FPM pada Frieren, Flamme, dan Fern. Untuk testing kinerja naikan

- pm.max_children
- pm.start_servers
- pm.min_spare_servers
- pm.max_spare_servers

sebanyak tiga percobaan dan lakukan testing sebanyak 100 request dengan 10 request/second kemudian berikan hasil analisisnya pada Grimoire.

Jawab:

A. Testing 1

```

om = dynamic
om.max_children = 10
om.start_servers = 2
om.min_spare_servers = 1
om.max_spare_servers = 3
om.process_idle_timeout = 5s

```

```

root@Sein:~# ab -n 100 -c 10 -p login.json -T application/json
http://www.riegel.canyon.B04.com/api/auth/login
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

```

Benchmarking www.riegel.canyon.B04.com (be patient).....done

```

Server Software:      nginx/1.14.2
Server Hostname:      www.riegel.canyon.B04.com
Server Port:          80

```

```

Document Path:        /api/auth/login
Document Length:       95 bytes

```

```

Concurrency Level:     10
Time taken for tests:   0.821 seconds
Complete requests:     100
Failed requests:        36
  (Connect: 0, Receive: 0, Length: 36, Exceptions: 0)
Non-2xx responses:     100
Total transferred:     275498 bytes
Total body sent:        22300
HTML transferred:      244364 bytes
Requests per second:   121.84 [#/sec] (mean)
Time per request:      82.075 [ms] (mean)
Time per request:      8.207 [ms] (mean, across all concurrent requests)
Transfer rate:          327.80 [Kbytes/sec] received
                        26.53 kb/s sent
                        354.33 kb/s total

```

```

Connection Times (ms)
      min mean[+/-sd] median  max
Connect:    0   1   1.4    1    5

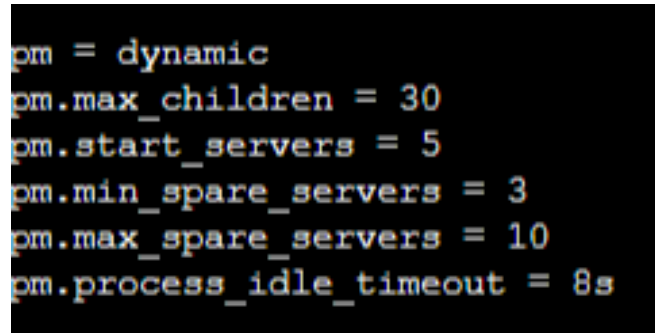
```

Processing: 24 76 14.5 76 102
Waiting: 23 74 14.3 75 101
Total: 24 77 14.1 78 102

Percentage of the requests served within a certain time (ms)

50% 78
66% 84
75% 87
80% 89
90% 94
95% 94
98% 97
99% 102
100% 102 (longest request)

B. Testing 2



```
pm = dynamic  
pm.max_children = 30  
pm.start_servers = 5  
pm.min_spare_servers = 3  
pm.max_spare_servers = 10  
pm.process_idle_timeout = 8s
```

```
root@Sein:~# ab -n 100 -c 10 -p login.json -T application/json  
http://www.riegel.canyon.B04.com/api/auth/login  
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>  
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/  
Licensed to The Apache Software Foundation, http://www.apache.org/
```

Benchmarking www.riegel.canyon.B04.com (be patient).....done

Server Software: nginx/1.14.2
Server Hostname: www.riegel.canyon.B04.com
Server Port: 80

Document Path: /api/auth/login
Document Length: 95 bytes

Concurrency Level: 10
Time taken for tests: 1.137 seconds
Complete requests: 100

Failed requests: 40
(Connect: 0, Receive: 0, Length: 40, Exceptions: 0)
Non-2xx responses: 100
Total transferred: 301810 bytes
Total body sent: 22300
HTML transferred: 270460 bytes
Requests per second: 87.98 [#/sec] (mean)
Time per request: 113.660 [ms] (mean)
Time per request: 11.366 [ms] (mean, across all concurrent requests)
Transfer rate: 259.31 [Kbytes/sec] received
19.16 kb/s sent
278.47 kb/s total

Connection Times (ms)

	min	mean	mean[+/-sd]	median	max
Connect:	0	1	0.9	1	5
Processing:	39	88	28.8	83	251
Waiting:	39	88	28.6	83	246
Total:	44	89	28.8	84	252

Percentage of the requests served within a certain time (ms)

50%	84
66%	93
75%	99
80%	105
90%	112
95%	145
98%	193
99%	252
100%	252 (longest request)

C. Testing 3

```
pm = dynamic
pm.max_children = 75
pm.start_servers = 10
pm.min_spare_servers = 5
pm.max_spare_servers = 20
pm.process_idle_timeout = 10s
```

```
root@Sein:~# ab -n 100 -c 10 -p login.json -T application/json
http://www.riegel.canyon.B04.com/api/auth/login
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
```

Licensed to The Apache Software Foundation, <http://www.apache.org/>

Benchmarking www.riegel.canyon.B04.com (be patient).....done

Server Software: nginx/1.14.2
Server Hostname: www.riegel.canyon.B04.com
Server Port: 80

Document Path: /api/auth/login
Document Length: 95 bytes

Concurrency Level: 10
Time taken for tests: 1.457 seconds
Complete requests: 100
Failed requests: 37
(Connect: 0, Receive: 0, Length: 37, Exceptions: 0)
Non-2xx responses: 100
Total transferred: 282074 bytes
Total body sent: 22300
HTML transferred: 250888 bytes
Requests per second: 68.65 [#/sec] (mean)
Time per request: 145.660 [ms] (mean)
Time per request: 14.566 [ms] (mean, across all concurrent requests)
Transfer rate: 189.11 [Kbytes/sec] received
14.95 kb/s sent
204.06 kb/s total

Connection Times (ms)

	min	mean	+/-sd	median	max
Connect:	1	1	0.9	1	6
Processing:	18	113	49.8	111	316
Waiting:	18	112	49.6	109	315
Total:	19	115	49.8	111	316

Percentage of the requests served within a certain time (ms)

50%	111
66%	125
75%	143
80%	154
90%	176
95%	199
98%	232
99%	316

100% 316 (longest request)

D. Analisis

Dalam konfigurasi PHP-FPM (FastCGI Process Manager), yang merupakan alternatif PHP FastCGI dengan beberapa fitur tambahan yang berguna untuk situs web dengan lalu lintas tinggi, berikut adalah penjelasan dari pengaturan yang digunakan:

1. `pm.max_children`
 - a. Ini menentukan jumlah maksimum child process yang diperbolehkan untuk melayani permintaan PHP. Menetapkan nilai yang terlalu rendah dapat menyebabkan server tidak mampu menangani permintaan tambahan jika semuanya sedang sibuk, sedangkan nilai yang terlalu tinggi dapat menyebabkan penggunaan memori yang berlebihan dan membebani server.
 - b. Meningkatkan ``pm.max_children`` dapat memungkinkan server untuk menangani lebih banyak permintaan secara bersamaan, tetapi juga dapat meningkatkan penggunaan memori dan membebani server jika angkanya terlalu tinggi untuk sumber daya server yang tersedia.
2. `pm.start_servers`
 - a. Ini adalah jumlah child process yang dijalankan saat PHP-FPM dimulai. Nilai ini harus diatur ke nilai rata-rata yang diharapkan untuk child process berdasarkan beban normal situs.
 - b. Meningkatkan ``pm.start_servers`` dapat mengurangi waktu tunggu untuk pembuatan proses baru ketika ada lonjakan lalu lintas, tetapi juga dapat meningkatkan penggunaan sumber daya saat startup jika banyak proses ini tidak diperlukan.
3. `pm.min_spare_servers`
 - a. Ini menentukan jumlah minimum child process idle yang PHP-FPM akan menjaga. Jika jumlah idle process jatuh di bawah nilai ini, PHP-FPM akan membuat child process baru untuk menjaga keseimbangan. Ini memastikan bahwa ada cukup process yang siap untuk menangani permintaan masuk tanpa penundaan.
 - b. Meningkatkan ``pm.min_spare_servers`` memastikan bahwa lebih banyak proses siap menunggu permintaan masuk, yang dapat meningkatkan responsivitas tetapi juga meningkatkan penggunaan sumber daya secara tidak perlu selama waktu-waktu ketika permintaan rendah.
4. `pm.max_spare_servers`
 - a. Ini adalah kebalikan dari ``pm.min_spare_servers``. Ini menentukan jumlah maksimum child process idle yang PHP-FPM akan mempertahankan. Jika jumlah idle process melebihi nilai ini, PHP-FPM akan membunuh beberapa process yang tidak dibutuhkan untuk menghemat sumber daya.
 - b. Meningkatkan ``pm.max_spare_servers`` memungkinkan untuk lebih banyak proses idle tanpa mematikannya, yang dapat bermanfaat selama lonjakan lalu lintas mendadak tetapi juga dapat membuang-buang sumber daya selama waktu-waktu ketika situs tidak sibuk.'

Berdasarkan ketiga testing yang telah dilakukan dalam percobaan pertama dengan konfigurasi pm.max_children yang rendah pada 10, server menghasilkan performa yang cukup baik dengan 121,84 permintaan per detik, namun terjadi 36 gagal permintaan yang menunjukkan konfigurasi ini tidak cukup untuk beban yang diberikan. Dalam percobaan kedua, peningkatan signifikan pada pm.max_children menjadi 30 dan parameter lainnya meningkatkan kapasitas untuk menangani permintaan, tetapi mengejutkannya, performa menurun menjadi 87,98 permintaan per detik dengan 40 gagal permintaan, yang mungkin menunjukkan bahwa kenaikan pm.max_children membebani server lebih dari yang dapat ditanganinya. Percobaan ketiga dengan pm.max_children sebanyak 75 menunjukkan penurunan performa lebih lanjut menjadi 68,65 permintaan per detik dengan 37 gagal permintaan, menegaskan bahwa peningkatan sumber daya tidak selalu berbanding lurus dengan peningkatan kinerja. Waktu proses rata-rata yang bertambah dan tingkat kegagalan yang konsisten menunjukkan bahwa server mungkin mengalami pembatasan pada faktor lain seperti I/O disk, bandwidth jaringan, atau konfigurasi database yang tidak optimal. Optimasi lebih lanjut diperlukan untuk mengidentifikasi dan mengatasi bottleneck ini untuk memanfaatkan sepenuhnya sumber daya yang dialokasikan.

Nomor 20

Nampaknya hanya menggunakan PHP-FPM tidak cukup untuk meningkatkan performa dari worker maka implementasikan Least-Conn pada Eisen. Untuk testing kinerja dari worker tersebut dilakukan sebanyak 100 request dengan 10 request/second.

Jawab:

A. Apache Benchark

```
root@Sein:~# ab -n 100 -c 10 -p login.json -T application/json
http://www.riegel.canyon.B04.com/api/auth/login
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

Benchmarking www.riegel.canyon.B04.com (be patient).....done

```
Server Software:      nginx/1.14.2
Server Hostname:      www.riegel.canyon.B04.com
Server Port:          80
```

```
Document Path:        /api/auth/login
Document Length:       95 bytes
```

```
Concurrency Level:     10
Time taken for tests:   0.797 seconds
Complete requests:     100
```


Failed requests: 36
 (Connect: 0, Receive: 0, Length: 36, Exceptions: 0)
 Non-2xx responses: 100
 Total transferred: 275497 bytes
 Total body sent: 22300
 HTML transferred: 244364 bytes
 Requests per second: 125.41 [#sec] (mean)
 Time per request: 79.741 [ms] (mean)
 Time per request: 7.974 [ms] (mean, across all concurrent requests)
 Transfer rate: 337.39 [Kbytes/sec] received
 27.31 kb/s sent
 364.70 kb/s total

Connection Times (ms)
 min mean[+/-sd] median max
 Connect: 0 1 1.7 0 7
 Processing: 11 74 34.4 77 134
 Waiting: 10 72 33.1 77 134
 Total: 12 75 34.6 78 137

Percentage of the requests served within a certain time (ms)
 50% 78
 66% 102
 75% 106
 80% 108
 90% 120
 95% 124
 98% 130
 99% 137
 100% 137 (longest request)

B. Analisis

Analisis pengujian ApacheBench setelah penerapan algoritma Least-Conn pada Eisen, yang berfungsi sebagai Load Balancer, menunjukkan peningkatan kinerja yang signifikan. Algoritma Least-Conn dirancang untuk mendistribusikan permintaan yang masuk ke worker yang memiliki jumlah koneksi terendah, membantu untuk menjaga beban yang lebih merata di antara worker dan mengurangi waktu tunggu ketika beban kerja yang tinggi. Hasilnya menunjukkan peningkatan jumlah permintaan per detik menjadi 125,41, dari hasil tes sebelumnya yang terbaik yaitu 121,84. Waktu rata-rata yang dibutuhkan untuk setiap permintaan menurun menjadi 79,741 ms, yang menandakan responsivitas sistem yang lebih baik.

Namun, masih ada 36 permintaan yang gagal, yang menunjukkan bahwa meskipun distribusi beban lebih efisien, ada faktor lain yang membatasi kinerja. Kegagalan ini

mungkin disebabkan oleh konfigurasi PHP-FPM yang tidak optimal atau keterbatasan lain dalam arsitektur aplikasi. Dalam hal ini, walaupun Least-Conn sebagai strategi load balancing memberikan distribusi beban yang lebih adil dan efektif, perbaikan pada konfigurasi PHP-FPM masih perlu untuk mencapai keandalan yang lebih tinggi dan mengurangi jumlah permintaan yang gagal.

Kesimpulannya, penerapan algoritma Least-Conn pada Eisen sebagai Load Balancer memang menunjukkan peningkatan performa secara keseluruhan. Ini memberikan justifikasi bahwa strategi ini lebih baik dalam mengelola koneksi yang masuk dibandingkan hanya mengandalkan peningkatan sumber daya PHP-FPM. Namun, untuk mencapai kinerja yang optimal, perlu ada keseimbangan antara konfigurasi load balancing yang cerdas dan konfigurasi PHP-FPM yang tepat. Algoritma Least-Conn membantu mendistribusikan beban kerja secara merata tetapi juga memerlukan worker backend yang dikonfigurasi dengan baik untuk menangani beban yang didistribusikan tersebut.