Tugas Kelompok Final Project

Studi Kasus Reverse Proxy dan Load Balancing Reverse Proxy

Oleh Kelompok D05:

1)	Rida Adila	05111840000002
2)	Clement Prolifel Priyatama	05111840000013
3)	Irsyadhani Dwi Shubhi	05111840000022

Dosen Pengampu:
Royyana Muslim Ijtihadie, S.Kom.,M.Kom., Ph.D.

S1 TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS

INSTITUT TEKNOLOGI SEPULUH NOPEMBER SURABAYA

2021

PENDAHULUAN

Pada tugas *Final Project* mata kuliah Pemrograman Jaringan ini, terdapat 2 studi kasus yang harus diimplementasikan. Yang pertama ialah studi kasus mengenai *reverse proxy*. Dimana pada studi kasus ini akan terdapat *client*, *proxy*, dan sebuah *backend server*. Yang mana nantinya *reverse proxy* akan menerjemahkan *path* pada sebuah URL untuk diteruskan ke *backend server* yang sesuai. Jika *reverse proxy* menerima *request* dalam *path* /images, maka objek akan di *retrieve* dari *backend server* yang melayani /images. Untuk *HTTP client* yang digunakan dalam uji coba studi kasus pertama ini berupa *web browser* dan curl *command*.

Lalu pada studi kasus kedua, mengenai *load balancing reverse proxy*, yang mana akan diimplementasikan 2 jenis model *load balancing reverse proxy*, yakni model *threaded* dan model *asynchronous*. *Request* yang diterima *reverse proxy*, akan diteruskan ke *cluster backend* dengan cara Round Robin. Maksud Round Robin sendiri ialah request diteruskan secara bergantian dalam proporsi yang sama. Kemudian akan dilakukan uji *testing performance* dengan menggunakan *Apache Benchmark* dengan target *server reverse proxy*. Uji *testing performance* sendiri dilakukan masing-masing 15 kali percobaan pada *threaded* dan *asynchronous* dengan menggunakan 10000 *request* dimana nantinya masing-masing tiap jumlah *backend server* mulai 1-5 akan menggunakan *concurrency level* sebesar: 2,5,10.

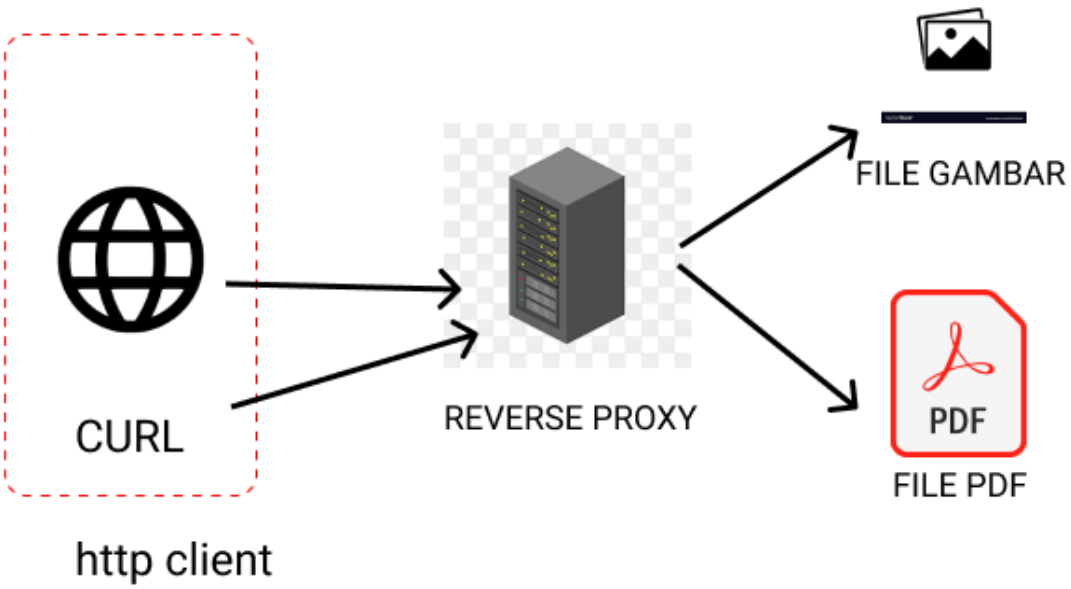
PEMBAHASAN

Tugas Anggota Kelompok:

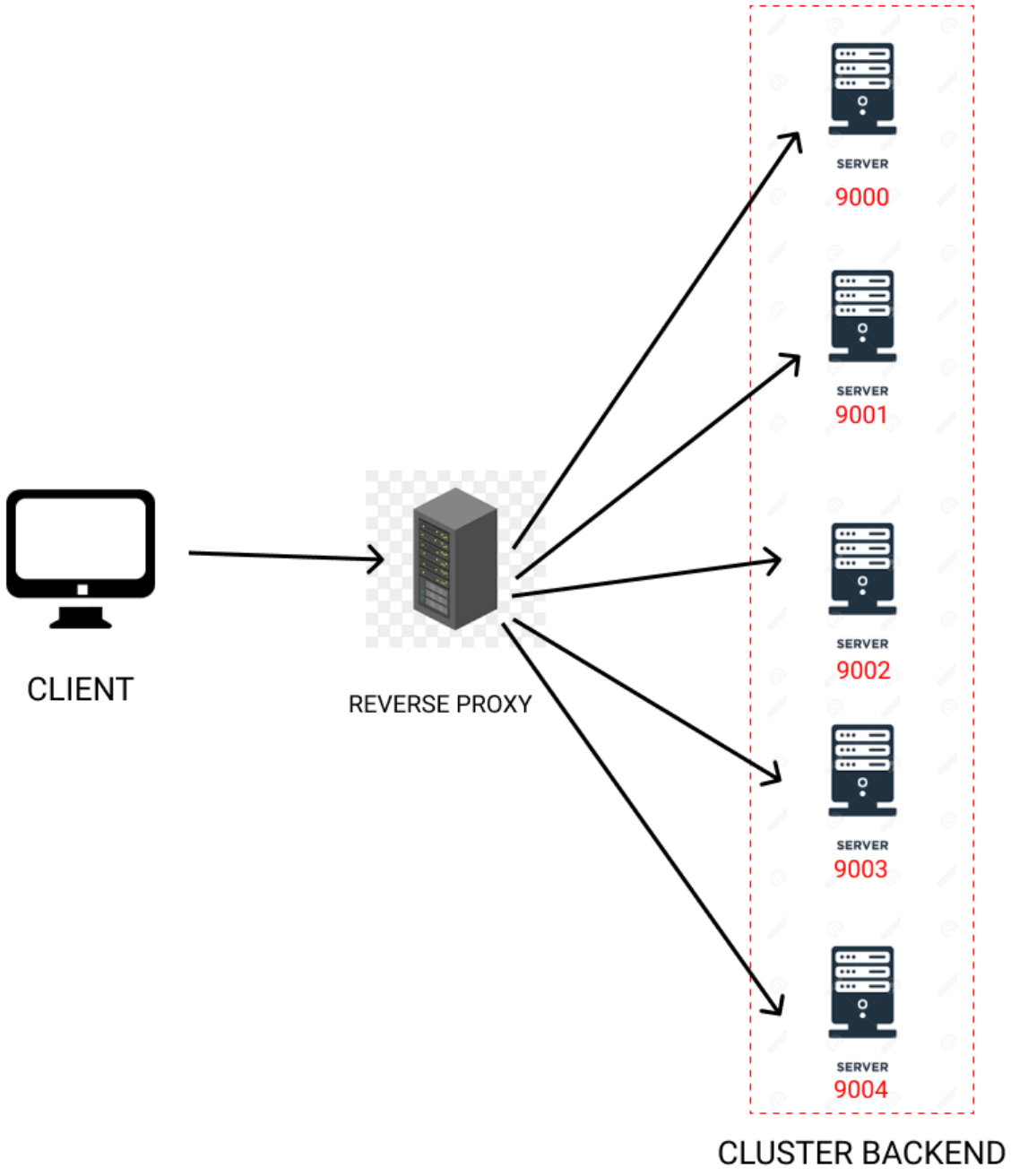
1) Rida Adila 05111840000002
    - Mengerjakan Load Balancing Threaded
2) Irsyadhani Dwi Shubhi 05111840000022
    - Mengerjakan Load Balancing Asynchronous
3) Clement Prolifel Priyatama 05111840000013
    - Mengerjakan Reverse Proxy

Arsitektur dan Konfigurasi:

1) Arsitektur pada studi kasus 1 reverse proxy



2) Arsitektur pada studi kasus 2 load balancer, dengan konfigurasi server 1 dimulai dari port 9000 dst



Pengujian:

1) Berikut merupakan cara melakukan pengujian *reverse proxy* pada studi kasus 1:
   a) Pada folder server, menjalankan perintah python3 -m http.server sebagai *backend server* yang dapat diakses pada http://localhost:8080.
   b) Untuk menjalankan *reverse proxy*, maka menjalankan python3 reverseProxy.py. Sehingga *reverse proxy* dapat diakses pada http://localhost:9097.

c) Menggunakan curl dengan perintah curl localhost:9097/images/2.png -o 2.png untuk *download* file 2.png dengan nama *output* file 2.png. Sedangkan untuk file pdf, menggunakan perintah curl curl localhost:9097/pdf/rfc2616.pdf -o test-pdf-reverse.pdf dengan nama *output* file test-pdf-reverse.pdf.

d) Pada *browser,* mengakses http://localhost:9097/images/4%20hokya%20-yeyeye.png untuk mendapatkan file png, sedangkan http://localhost:9097/pdf/rfc2616.pdf untuk file pdf.

2) Berikut merupakan cara melakukan pengujian *load balancing reverse proxy* pada studi kasus 2:

a) Pada model *Threaded*:

i) Membuat beberapa terminal sesuai dengan kebutuhan *cluster backend server*. Jika dibutuhkan 1 *backend server*, maka membuat hanya 1 terminal.

ii) Pada setiap terminal menjalankan python3 server_thread_http.py <port>. Port menyesuaikan jumlah *backend server* yang dimulai dari port 9000. Sehingga apabila jumlah *backend server* ada 5, maka port yang akan dibuka antara lain: port 9000, port 9001, port 9002, port 9003, port 9004.

iii) Untuk *reverse proxy,* membuat satu terminal dan menjalankan python3 socket_proxy_thread.py. Maka *reverse proxy* dapat diakses pada http://localhost:18000/.

iv) Untuk penerapan *load balancing* secara Round Robin, maka menggunakan *library* lb.py dengan membuat array yang berisi alamat masing-masing *backend server* berdasarkan poin 2.

v) Untuk menjalankan *performance test* dengan Apache Benchmark, maka membuat satu terminal baru, kemudian menjalankan ab -n 10000 -c <jumlah concurrency> http://localhost:18000/. Kemudian menunggu hasil setelah proses Apache Benchmark selesai.

b) Pada model *Asynchronous*:

i) Membuat beberapa terminal sesuai dengan kebutuhan *cluster backend server*. Jika dibutuhkan 1 *backend server*, maka membuat hanya 1 terminal.

ii) Pada setiap terminal menjalankan python3 async_server.py <port>. Port menyesuaikan jumlah *backend server* yang dimulai dari port 9000. Sehingga apabila jumlah *backend server* ada 5, maka port yang akan dibuka antara lain: port 9000, port 9001, port 9002, port 9003, port 9004.

iii) Untuk *reverse proxy,* membuat satu terminal dan menjalankan python3 socket_proxy_async.py. Maka *reverse proxy* dapat diakses pada http://localhost:8890/.

iv) Untuk penerapan *load balancing* secara Round Robin, maka menggunakan *library* lb.py dengan membuat array yang berisi alamat masing-masing *backend server* berdasarkan poin 2.

v) Untuk menjalankan *performance test* dengan Apache Benchmark, maka membuat satu terminal baru, kemudian menjalankan ab -n 10000 -c <jumlah concurrency> http://localhost:8890/. Kemudian menunggu hasil setelah proses Apache Benchmark selesai.

Screenshot Hasil: Screenshot hasil pengujian terdapat pada Github kelompok kami, yaitu di: https://github.com/prolifel/Pemrograman_Jaringan_D_Kelompok_5/tree/master/fp/Hasil%20Screenshot

Tabel Hasil Apache Benchmark

Tabel Apache Benchmark Asynchronous :

No	Jumlah Http Server	Concurrency	Jumlah complete request	Non 2x response	Jumlah request per second	Time per request mean (ms)
1	1 backend server	2	2285	1	-	-
2	1 backend server	5	10000	0	791.65	6.316
3	1 backend server	10	10000	0	836.06	11.961
4	2 backend server	2	2056	1	-	-
5	2 backend server	5	10000	0	750.21	6.665
6	2 backend server	10	10000	0	842.54	11.869
7	3 backend server	2	1434	1	-	-
8	3 backend server	5	10000	0	752.49	6.645
9	3 backend server	10	10000	0	869.06	11.506
10	4 backend server	2	5607	1	-	-
11	4 backend server	5	10000	0	711.83	7.024
12	4 backend server	10	10000	0	903.78	11.065
13	5 backend server	2	4156	1	-	-
14	5 backend server	5	10000	0	729.48	6.854
15	5 backend server	10	10000	0	869.67	11.499

Tabel Apache Benchmark Threaded :

No	Jumlah Http Server	Concurrency	Jumlah complete request	Non 2x response	Jumlah request per second	Time per request mean (ms)
1	1 backend server	2	10000	0	14.84	134.767
2	1 backend server	5	10000	0	33.92	147.394
3	1 backend server	10	9999	1	14.52	688.485
4	2 backend server	2	10000	0	31.07	64.363
5	2 backend server	5	10000	0	37.38	133.774
6	2 backend server	10	10000	0	42.83	233.457
7	3 backend server	2	10000	0	36.62	54.612
8	3 backend server	5	10000	0	44.06	113.473
9	3 backend server	10	10000	0	52.06	192.090
10	4 backend server	2	10000	0	35.32	56.626
11	4 backend server	5	10000	0	38.30	130.552
12	4 backend server	10	10000	0	40.74	245.476
13	5 backend server	2	10000	0	29.55	67.686
14	5 backend server	5	10000	0	31.15	160.522
15	5 backend server	10	10000	0	41.58	240.513

KESIMPULAN

Berdasarkan hasil percobaan yang dilakukan pada bagian *reverse proxy*, *reverse proxy* dapat berjalan dengan sebagaimana mestinya, dapat menghubungkan pengguna dengan *backend server*, dan dapat melakukan *download* file.

Dari uji testing yang dilakukan pada bagian load balancing reverse proxy asynchronous jika jumlah backend server yang digunakan semakin banyak, maka waktu yang dibutuhkan akan relatif semakin lama. Sedangkan pada bagian threaded waktu yang kami dapatkan dalam melakukan pemrosesan *requests* tidak menentu antara perbedaan jumlah backend server yang digunakan. Kemudian pada semua uji coba, jika *concurrency* yang digunakan semakin besar, maka waktu yang dibutuhkan untuk dapat menyelesaikan pemrosesan *request* akan semakin cepat. Antara asynchronous dan threaded, terlihat bahwa pada asynchronous waktu yang diperlukan untuk dapat menyelesaikan *requests* lebih cepat dibandingkan *threaded*, karena pada *asynchronous processing* eksekusi dijalankan ketika instruksi diberikan, sedangkan pada proses *threaded* yang bukan *asynchronous thread* akan berjalan normal sesuai *flow control*.

LINK REPOSITORY KELOMPOK :
https://github.com/prolifel/Pemrograman_Jaringan_D_Kelompok_5/tree/master/fp

LINK REPOSITORY INDIVIDU :

1) Rida Adila 05111840000002 :
   https://github.com/ridaadila/Pemrograman_Jaringan_D/tree/master/Pemrograman_Jaringan_D_Kelompok_5-master/fp
2) Clement Prolifel P. 05111840000013 :
   https://github.com/prolifel/Pemrograman_Jaringan_D/tree/master/fp
3) Irsyadhani Dwi S. 05111840000022 :
   https://github.com/irsyadhani/Pemrograman_Jaringan_D/tree/master/Tugas%20Kelompok%20Final%20Project