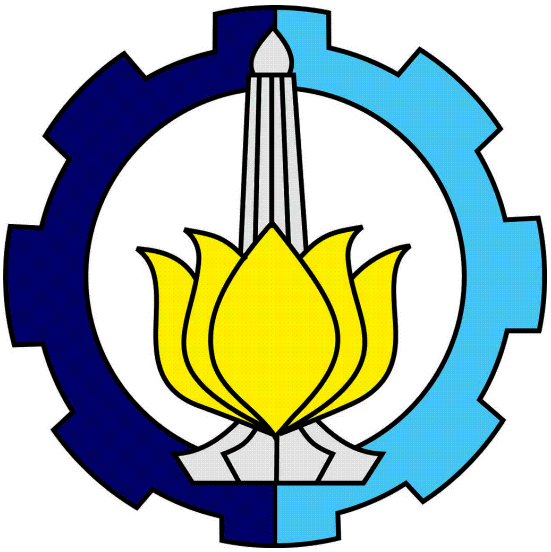
Final Project

Reverse Proxy dan Load Balancing Reverse Proxy



Pemrograman Jaringan

Kelas E

Rosa Valentine Lammora	05111840000035
Afia Hana Yusriya	05111840000111
Salsabila Harlen	05111840000127

Institut Teknologi Sepuluh Nopember

Surabaya

2021

A. Pendahuluan

a. Reverse proxy

Reverse proxy membantu penanganan *request* pada web server dengan menjadi jembatan atau perantara web server dan pengguna. Request diteruskan ke web server yang sesuai, kemudian hasil atau responnya dikembalikan ke client pemberi *request*. Keberadaan reverse proxy meningkatkan keamanan sistem karena *request* tidak langsung ditangani web server.

Secara lebih spesifik, reverse proxy memiliki kegunaan-kegunaan sebagai berikut:

1) Sebagai load balancer yang mendistribusikan banyak *request* dalam satu waktu agar server tidak *overloaded*.
2) Sebagai pelindung dari serangan terhadap server, karena informasi seperti private IP address milik server tidak dapat terlihat. Yang terlihat hanyalah IP address milik reverse proxy.
3) Melakukan *caching* terhadap konten, sehingga performa sistem meningkat.
4) Enkripsi SSL

b. Load Balancing Reverse Proxy

Load Balancing Reverse Proxy memiliki fungsionalitas khusus, yaitu sebagai *load balancer*. Biasanya, *load balancer* digunakan untuk lingkungan yang memiliki lebih dari satu server, untuk menjaga reliabilitas sistem dalam menangani *request* dari client-client yang terhubung. Penanganan *request* dengan *load balancer* dilakukan melalui distribusi *request* ke server-server sesuai kapasitasnya masing-masing. Dengan begitu, server tidak akan kelebihan kapasitas dan client bisa memperoleh *response* dengan lebih cepat.

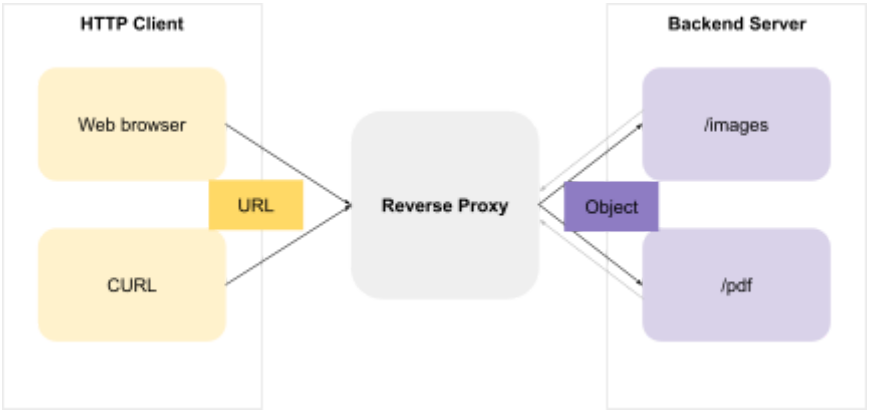
B. Pembagian Tugas

Rosa Valentine 05111840000035	**Afia Hana Yusriya 05111840000111**	**Salsabila Harlen 05111840000127**
Reverse Proxy	Load Balancing Reverse Proxy Asynchronous	Load Balancing Reverse Proxy Threaded

C. Arsitektur dan Konfigurasi

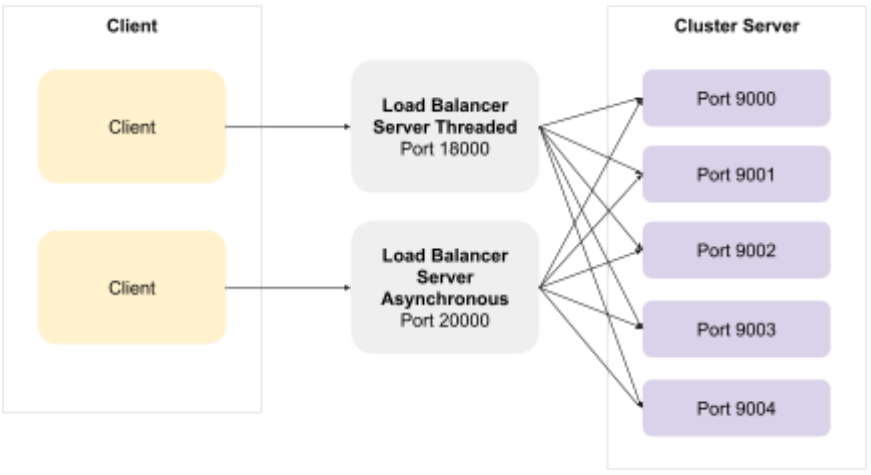
a. Study Case 1: Reverse Proxy

Diberikan arsitektur sebuah jaringan client-server sebagai berikut:



Pada arsitektur, terdapat reverse proxy yang bertugas untuk menerjemahkan path request berupa URL untuk backend server. Terjemahan ini dibutuhkan backend server untuk memahami lokasi request client agar bisa mengembalikannya, di kasus ini sebagai object file image atau pdf.

b. Study Case 2: Load Balancing Reverse Proxy



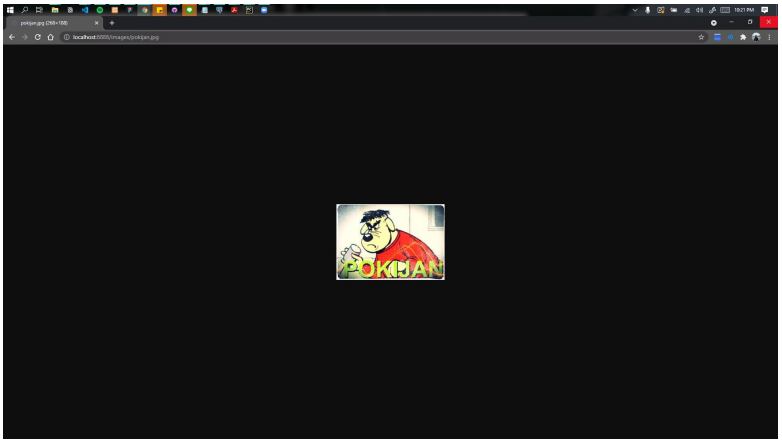
Pada arsitektur ini, terdapat 5 cluster server yang bertugas menangani request dari client. Request sebelumnya melewati load balancer (asynchronous atau threaded). Load balancer akan membagi request ke cluster server yang memiliki kapasitas memadai menggunakan metode Round Robin, yaitu pendistribusian request berdasarkan urutan masuknya ke server.

D. Pengujian & Screenshot Hasil

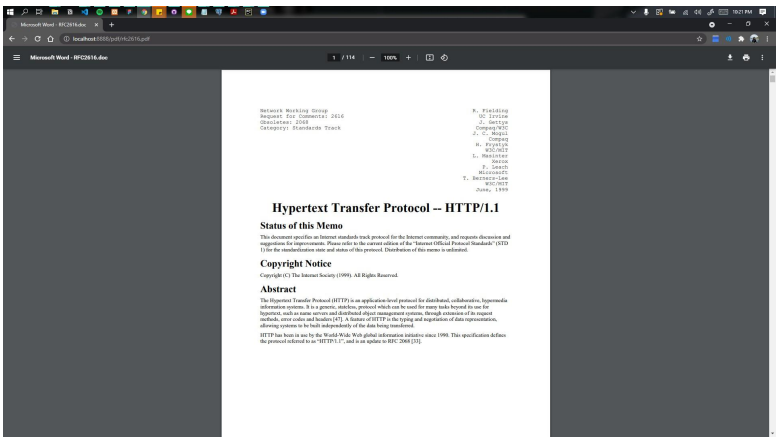
a. Study Case 1: Reverse Proxy

1. Implementasikan arsitektur berikut ini dengan memodifikasi program yang berkaitan pembahasan http server

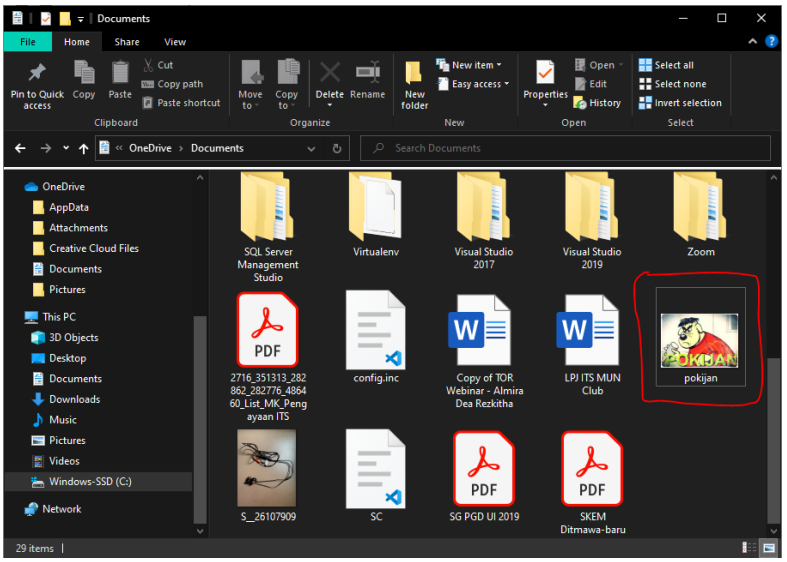
2. Reverse proxy dapat menerjemahkan path pada sebuah URL untuk diteruskan ke backend server yang sesuai
3. Pada gambar disamping, jika reverse proxy menerima request dalam path /images , maka object akan di retrieve dari backend server yang melayani /images
4. Gunakan http client berupa
    a. **Web browser (chrome/firefox)**
    Melakukan pengujian dengan mengakses suatu file bertipe *image* “pokijan.jpg” di web browser



Melakukan pengujian dengan mengakses suatu file ber-tipe pdf “rfc2616.pdf” di web browser



    b. **Curl**
    Melakukan pengujian dengan mengunduh file ber-tipe *image* “pokijan.jpg”, kemudian file yang telah terunduh akan masuk ke dalam file explorer.



b. Study Case 2: Load Balancing Reverse Proxy

1. Implementasikan load balancing reverse proxy berikut ini dengan model
   - Threaded
   - Asynchronous (sudah ada dalam contoh)
2. Request yang diterima reverse proxy akan diteruskan ke cluster backend dengan cara round robin.

Round robin -> bergantian dalam porsi yang sama

```
class LoadBalancer:
    def __init__(self):
        self.server_list = []
        self.server_list.append(("localhost", 9000))
        self.server_list.append(("localhost", 9001))
        self.server_list.append(("localhost", 9002))
        self.server_list.append(("localhost", 9003))
        self.server_list.append(("localhost", 9004))
        self.counter = 0

    def get_server(self):
        server = self.server_list[self.counter]
        self.counter += 1
        if self.counter >= len(self.server_list):
            self.counter = 0
        return server
```

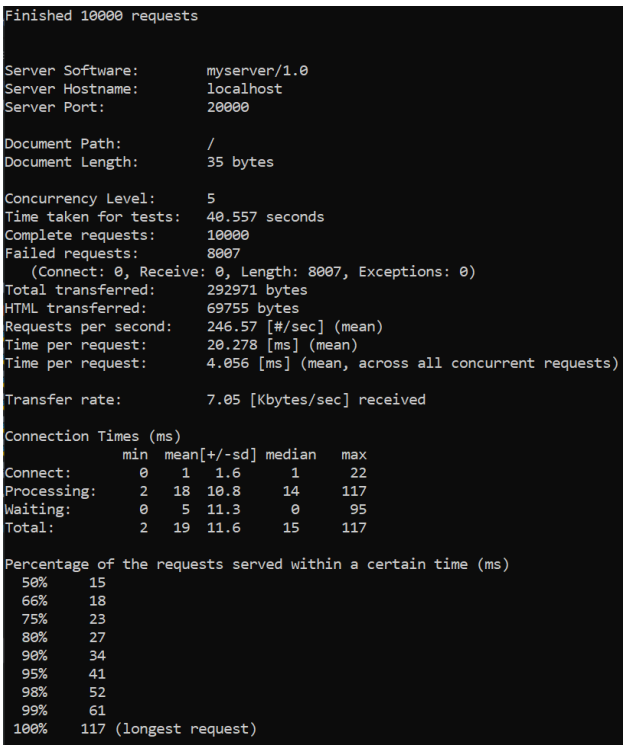
3. Jalankan performance test dengan menggunakan apache benchmark (ab) dengan target server pada reverse proxy
    - Jalankan file sesuai kebutuhan pengujian load balancing reverse proxy
    - Buka command prompt
    - Masuk ke folder xampp->apache->bin
    - Jalankan command untuk melakukan performance test
    - Lihat dan catat hasil performance test menggunakan apache benchmark (ab)

    Command untuk performance test
    `ab -n **jumlah_request** -c **concurrency** link`

    ab -n 10000 -c 2 http://localhost:20000/
    ab -n 10000 -c 5 http://localhost:20000/
    ab -n 10000 -c 10 http://localhost:20000/
    ab -n 10000 -c 2 http://localhost:18000/
    ab -n 10000 -c 5 http://localhost:18000/
    ab -n 10000 -c 10 http://localhost:18000/

4. Bandingkan performa
    - 1 HTTP server dengan multi http server
    - Concurrency 2, 5, 10
    - Jumlah backend server 1, 2, 3, 4, 5
    - Gunakan jumlah request 10000

    Salah satu hasil apache benchmark dengan jumlah request 10000, jumlah backend server 5, dan concurrency 5.



Screenshot hasil selengkapnya dapat dilihat di:

https://github.com/arommal/Pemrograman_Jaringan_E_Kelompok_5/tree/FinalProject/Soal%202/Screenshot%20Hasil

E. Tabel Hasil Apache Benchmark

ASYNCHRONOUS LOAD BALANCER

Jumlah http server	Concurrency	Jumlah Complete Request	Non-2xx Response	Jumlah Request per Second	Time per Request (mean across) [ms]
1	2	10000	-	36.55	27.650
1	5	10000	-	329.26	3.037
1	10	10000	-	19.84	50.406
2	2	10000	-	28.38	35.230
2	5	10000	-	183.84	5.439
2	10	10000	-	15.32	65.285
3	2	10000	-	85.32	11.270
3	5	10000	-	382.78	2.612

3	10	10000	-	16.48	60.678
4	2	10000	-	67.50	14.815
4	5	10000	-	393.98	2.538
4	10	10000	-	18.18	55.005
5	2	10000	-	232.70	4.297
5	5	10000	-	246.57	4.056
5	10	10000	-	18.20	54.951

THREADED LOAD BALANCER

Jumlah http server	Concurrency	Jumlah Complete Request	Non-2xx Response	Jumlah Request per Second	Time per Request (mean across) [ms]
1	2	10000	-	387.00	2.584
1	5	10000	-	400.05	2.500
1	10	10000	-	438.52	2.280
2	2	10000	-	406.37	2.461
2	5	10000	-	594.27	1.683
2	10	10000	-	16.34	61.198
3	2	10000	-	15.79	63.333
3	5	10000	-	658.87	1.518
3	10	10000	-	16.05	62.354
4	2	10000	-	24.77	40.364
4	5	10000	-	20.72	48.259
4	10	10000	-	10.70	93.497
5	2	10000	-	441.48	2.265
5	5	10000	-	355.33	2.814
5	10	10000	-	15.32	65.267

F. Kesimpulan

Berdasarkan hasil percobaan yang dilakukan pada *Load Balancing Reverse Proxy Asynchronous* dan *Threaded.* Pada tabel diatas, dapat dilihat bahwa pada *Asynchronous Server,* waktu yang dibutuhkan untuk dapat menyelesaikan requests relatif lebih cepat dibandingkan *Threaded*. Hal ini dikarenakan pada *asynchronous processing*, eksekusi dijalankan ketika instruksi diberikan, sedangkan pada *threaded* akan berjalan normal sesuai *flow control*. Maka, dapat disimpulkan bahwa performa pada *Asynchronous Server* lebih baik dibandingkan dengan *Threaded Server*.

G. Referensi

https://www.nginx.com/resources/glossary/reverse-proxy-vs-load-balancer/