Nama : Ivan Fairuz Adinata

NRP : 5025221167

Kelas : Progjar D

LAPORAN TUGAS ETS PROJAR D

1. Link GitHub: https://github.com/mlikiwe/progjar-d-2025-tugasets
2. Model Concurrency
    a. Multithreading menggunakan Pool

    Dari file_server.py pada tugas sebelumnya, kita akan membuat file_server baru dengan memanfatkan ThreadPoolExecutor.

```
def __init__(self, ipaddress='0.0.0.0', port=45000, max_workers=5):
        self.ipinfo = (ipaddress, port)
        self.my_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.my_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.executor = ThreadPoolExecutor(max_workers=max_workers)
        self.success_count = 0
        self.fail_count = 0
```

    Di dalam method init, kita akan menambahkan executor yang disesuaikan dengan metode awalnya, yakni ThreadPoolExecutor, dengan parameter jumlah worker nya. Untuk default, kita akan set 5, namun dalam penerapannya nanti, kita dapat mengatur jumlah dari worker saat kita jalankan server-nya.

    b. Multiprocessing menggunakan Pool

```
def __init__(self, ipaddress='0.0.0.0', port=45000, max_workers=5):
        self.ipinfo = (ipaddress, port)
        self.my_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.my_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.executor = ProcessPoolExecutor(max_workers=max_workers)
        self.success_count = multiprocessing.Value('i', 0)
        self.fail_count = multiprocessing.Value('i', 0)
```

    Sementara itu, di bagian ini kita akan memodifikasi file_server dengan menggunakan ProcessPoolExecutor. Hampir sama dengan pendekatan multithread menggunakan pool, di bagian method init, akan ditambahkan executor berupa ProcessPoolExecutor dengan parameter jumlah worker nya. Sama juga dengan pendekatan multithreading menggunakan pool, di saat nanti kita menjalankan server nya, kita dapat mengatur jumlah worker sesuai ketentuan dalam soal.

3. Konfigurasi Arsitektur Stress Test pada Client

Secara garis besar, saya di sini menambahkan script python untuk melakukan stress test dengan memanfaatkan utility yang ada pada file client sebelumnya, yakni UPLOAD dan GET dengan melakukan import file client_cli pada tugas 3. Terdapat beberapa method yang masing-masing memiliki fungsionalitas nya sendiri.

```
def generate_test_file(filename, size_bytes):
    with open(filename, 'wb') as f:
        f.write(os.urandom(size_bytes))
    return filename
```

Method generate_test_file tersebut berfungsi untuk membuat file dengan ukuran yang sudah ditentukan yang berisi data acak. Selanjutnya, terdapat method client_task

```
def client_task(operation, filename, concurrency_type="thread",
worker_id=0):
```

Di dalam method ini, kita akan mendefinisikan tugas dari satu client, di mana dia harus melakukan UPLOAD dan GET atau download. Kedua operasi tersebut akan ditentukan dengan parameter operation. Seperti yang telah dijelaskan, script ini akan memanfaatkan fungsi atau method yang ada di file client sebelumnya yakni fungsi send_command(). Di akhir method, akan di return value untuk waktu eksekusi dan throughput.

```
def run_stress_test(operation, file_size_mb, client_workers,
concurrency_type="thread"):
```

Berikutnya, terdapat method run_stress_test. Method ini merupakan inti dari stress test yang akan dijalankan oleh client. Di dalamnya, kita mendifinisikan file_size dan akan memanggil method generate_file_test. Di sini, kita juga akan menambahkan executor, di mana nantinya akan disesuaikan dengan server yang dijalankan, baik itu Multithread maupun Multiprocess. Hasil dari stress test akan disimpan ke dalam variabel result yang nantinya dapat ditampilkan di terminal ketika seluruh proses selesai.

Di method main, stress test akan dijalankan dengan memanggil method run_stress_test. Ketika client menjalankan script stress test ini, client dapat melengkapi beberapa argumen, seperti operation (UPLOAD dan DOWNLOAD), method (thread dan process), volume untuk mendefinisikan besar file (10, 50, 100) mb, dan worker untuk mengatur berapa banyak worker yang akan digunakan (1, 5, 50).

4. Capture Pengerjaan

   a. Tampilan client dan server ketika proses stress test

```
(base) jovyan@7dfa2c6f048b:~/work/progjar/progjar4a$ python file_server_threadpool.py 5
WARNING:root:Server running on ('0.0.0.0', 45000) with 5 workers
WARNING:root:Connection from ('172.16.16.102', 51934)
WARNING:root:Connection from ('172.16.16.102', 51944)
WARNING:root:Connection from ('172.16.16.102', 51956)
WARNING:root:Connection from ('172.16.16.102', 51966)
WARNING:root:Connection from ('172.16.16.102', 51974)
```

```
(base) jovyan@0d55ecc14602:~/work/progjar/progjar4a$ python stress_test_client.py --operation UPLOAD --method thread --volume 10 --worker 5
Running test: UPLOAD, 10MB, 5 workers, thread
WARNING:root:connecting to ('172.16.16.101', 45000)
WARNING:root:sending message
WARNING:root:connecting to ('172.16.16.101', 45000)
WARNING:root:sending message
WARNING:root:connecting to ('172.16.16.101', 45000)
WARNING:root:sending message
WARNING:root:connecting to ('172.16.16.101', 45000)
WARNING:root:connecting to ('172.16.16.101', 45000)
WARNING:root:sending message
WARNING:root:sending message
WARNING:root:data received from server:
WARNING:root:data received from server:
WARNING:root:data received from server:
WARNING:root:data received from server:
WARNING:root:data received from server:

Stress Test Results:
No    Op        Size(MB)   Client Workers  Concurrency  Avg Time(s)  Throughput(B/s) Client Success  Client Fail
1     UPLOAD    10         5               thread       29.11        361899.13       5              0
```

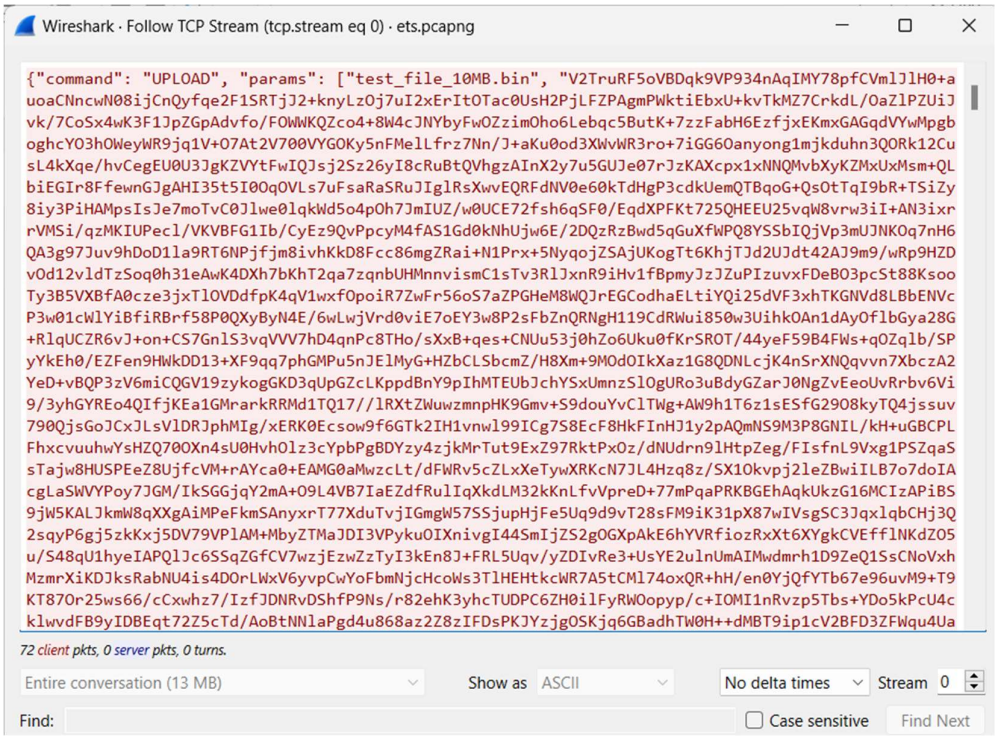
Di gambar yang atas merupakan tampilan server ketika dijalankan. Terlihat bahwa server berhasil berjalan dan berhasil menerima koneksi dari client dengan 5 worker, ditandai dengan Connection from (‘ip client’). Hal ini menunjukkan bahwa konfigurasi dari server telah benar. Di lain sisi, client mula-mula akan mencoba koneksi ke server, setelah berhasil terkoneksi, client akan memulai pengiriman data satu per satu. Dapat dilihat bahwa stress test dari client juga berhasil, ditandai dengan pesan data received from server sebanyak lima kali, dan diakhiri dengan result dari stress test ketika seluruh proses telah selesai.

   b. Tampilan wireshark ketika proses stress test

Source	Destination	Protocol	Lengtl	Info
172.16.16.102	172.16.16.101	TCP	74	58622 → 45000 [SYN] Seq=0 Win
172.16.16.101	172.16.16.102	TCP	74	45000 → 58622 [SYN, ACK] Seq=
172.16.16.102	172.16.16.101	TCP	66	58622 → 45000 [ACK] Seq=1 Ack
172.16.16.102	172.16.16.101	TCP	7306	58622 → 45000 [PSH, ACK] Seq=
172.16.16.101	172.16.16.102	TCP	66	45000 → 58622 [ACK] Seq=1 Ack
172.16.16.102	172.16.16.101	TCP	7306	58622 → 45000 [PSH, ACK] Seq=
172.16.16.102	172.16.16.101	TCP	10202	58622 → 45000 [PSH, ACK] Seq=
172.16.16.101	172.16.16.102	TCP	66	45000 → 58622 [ACK] Seq=1 Ack

Dapat dilihat bahwa ketika stress test dijalankan, di paket paling atas terdapat client dengan ip 172.16.16.102 yang meminta connect ke server, ketika server sudah melayani request koneksi tersebut dengan mengirimkan ACK, maka client akan mulai mengirimkan file ke server, ditandai dengan dikirimkannya PSH, ACK berulang-ulang

mengingat di sini, file yang akan di upload itu cukup besar, jadi tidak mungkin akan langsung dikirim sekaligus. Setelah seluruh proses selesai, maka seperti koneksi TCP pada umumnya, keduanya akan mengakiri koneksi dengan FIN. Berikut merupakan tampilan TCP stream yang benar menandakan bahwa proses yang di-capture merupakan proses UPLOAD dengan besar file 10mb.



Kedua tampilan tersebut merupakan sebuah contoh yang diambil dari satu scenario, yakni UPLOAD file 10mb dengan worker di server dan client berjumlah 5 dengan metode multithreading menggunakan pool. Dengan kombinasi yang ada, akan terdapat 108 skenario yang berbeda yang akan diuraikan pada table di bawah ini.

5. Tabel Kombinasi Percobaan Stress Test

   a. Multithread menggunakan pool

No	Operation	Size(MB)	Client Worker	Server worker	Avg Time(s)	Thoughput (B/s)	Client Success	Client Fail
1	UPLOAD	10	1	1	02.09	5013825.51	1	0
2	UPLOAD	10	5	1	6.39	2210786.94	5	0
3	UPLOAD	10	50	1	55.59	388790.37	50	0
4	UPLOAD	50	1	1	50.67	1034744.82	1	0
5	UPLOAD	50	5	1	151.74	464905.49	5	0
6	UPLOAD	50	50	1	1246.36	93553.22	50	0
7	UPLOAD	100	1	1	186.55	562090.90	1	0

8	UPLOAD	100	5	1	556.11	257663.70	5	0
9	UPLOAD	100	50	1	4720.53	50842.74	50	0
10	DOWNLOAD	10	1	1	1.85	5676490.34	1	0
11	DOWNLOAD	10	5	1	5.48	2596743.88	5	0
12	DOWNLOAD	10	50	1	45.88	516997.77	50	0
13	DOWNLOAD	50	1	1	45.92	1141692.99	1	0
14	DOWNLOAD	50	5	1	137.04	523348.03	5	0
15	DOWNLOAD	50	50	1	1159.00	103393.18	50	0
16	DOWNLOAD	100	1	1	183.62	571047.76	1	0
17	DOWNLOAD	100	5	1	542.50	265561.75	5	0
18	DOWNLOAD	100	50	1	4916.21	48951.11	50	0
19	UPLOAD	10	1	5	0.55	18957979.56	1	0
20	UPLOAD	10	5	5	1.57	6812291.31	5	0
21	UPLOAD	10	50	5	8.91	2038987.26	50	0
22	UPLOAD	50	1	5	2.57	20381339.78	1	0
23	UPLOAD	50	5	5	7.46	8444758.99	5	0
24	UPLOAD	50	50	5	95.81	1201389.19	50	0
25	UPLOAD	100	1	5	5.27	19909892.67	1	0
26	UPLOAD	100	5	5	17.21	7046066.67	5	0
27	UPLOAD	100	50	5	143.94	1527645.34	50	0
28	DOWNLOAD	10	1	5	0.09	121472192.91	1	0
29	DOWNLOAD	10	5	5	0.25	47219088.13	5	0
30	DOWNLOAD	10	50	5	1.62	10404414.98	50	0
31	DOWNLOAD	50	1	5	0.80	65572821.46	1	0
32	DOWNLOAD	50	5	5	3.74	14120494.91	5	0
33	DOWNLOAD	50	50	5	17.31	5162003.91	50	0
34	DOWNLOAD	100	1	5	2.32	45249035.39	1	0
35	DOWNLOAD	100	5	5	09.01	12102865.10	5	0
36	DOWNLOAD	100	50	5	52.52	3326290.76	50	0
37	UPLOAD	10	1	50	13.94	752412.30	1	0
38	UPLOAD	10	5	50	1.17	9993012.03	5	0
39	UPLOAD	10	50	50	44.99	278035.29	46	4
40	UPLOAD	50	1	50	2.23	23516382.27	1	0
41	UPLOAD	50	5	50	8.57	6190098.89	5	0
42	UPLOAD	50	50	50	38.47	2031769.01	42	8

43	UPLOAD	100	1	50	5.18	20262254.34	1	0
44	UPLOAD	100	5	50	17.32	6640091.18	5	0
45	UPLOAD	100	50	50	211.17	883667.14	48	2
46	DOWNLOAD	10	1	50	0.10	109792238.51	1	0
47	DOWNLOAD	10	5	50	0.45	26963484.64	5	0
48	DOWNLOAD	10	50	50	2.24	8880630.71	50	0
49	DOWNLOAD	50	1	50	0.74	70415635.36	1	0
50	DOWNLOAD	50	5	50	2.90	19755962.85	5	0
51	DOWNLOAD	50	50	50	23.79	4603803.05	50	0
52	DOWNLOAD	100	1	50	2.22	47151411.69	1	0
53	DOWNLOAD	100	5	50	8.61	14812669.51	5	0
54	DOWNLOAD	100	50	50	77.93	2474775.17	50	0

Stress test pada server multithreading pool menunjukkan bahwa waktu per cient meningkat drastis dengan bertambahnya jumlah worker dan ukuran file.Hal ini menunjukkan keterbatasan server dalam menangani beban concurrency yang tinggi. Download umumnya lebih cepat daripada upload karena overhead I/O disk yang lebih rendah. Throughput tertinggi dicapai saat client worker sedikit (1 atau 5) dan server worker 5, tetapi menurun drastis pada 50 client worker, yang menunjukkan bottleneck pada pengelolaan thread atau sumber daya server. Sebagian besar skenario berjalan tanpa kegagalan, kecuali pada 50 client worker dan 50 server worker untuk upload, di mana 4-8 klien gagal karena server mencapai titik jenuh. Denagn meningkatkan server worker dari 1 ke 5, kita akan mendapatkan peningkatkan performa, tetapi ketika ditambah hingga 50 worker, performa justru menurun akibat overhead thread.

b. Multiprocess menggunakan pool

No	Operation	Size(MB)	Client Worker	Server worker	Avg Time(s)	Thoughput (B/s)	Client Success	Client Fail
1	UPLOAD	10	1	1	0.47	22400703.04	1	0
2	UPLOAD	10	5	1	1.22	11646687.73	5	0
3	UPLOAD	10	50	1	9.75	2110956.70	50	0
4	UPLOAD	50	1	1	2.99	17548633.14	1	0
5	UPLOAD	50	5	1	6.86	10210181.64	5	0
6	UPLOAD	50	50	1	89.21	1749200.00	50	0
7	UPLOAD	100	1	1	8.78	11942749.57	1	0

8	UPLOAD	100	5	1	33.11	3326957.75	5	0
9	UPLOAD	100	50	1	240.24	1064982.00	50	0
10	DOWNLOAD	10	1	1	0.14	73296621.71	1	0
11	DOWNLOAD	10	5	1	0.32	43051715.38	5	0
12	DOWNLOAD	10	50	1	2.57	8634086.76	50	0
13	DOWNLOAD	50	1	1	0.92	57230684.99	1	0
14	DOWNLOAD	50	5	1	2.42	28506726.80	5	0
15	DOWNLOAD	50	50	1	19.73	5834835.57	50	0
16	DOWNLOAD	100	1	1	2.27	46204597.33	1	0
17	DOWNLOAD	100	5	1	6.89	20364204.71	5	0
18	DOWNLOAD	100	50	1	57.11	4096706.98	50	0
19	UPLOAD	10	1	5	0.50	20875273.75	1	0
20	UPLOAD	10	5	5	6.55	2180870.36	5	0
21	UPLOAD	10	50	5	55.32	424451.22	50	0
22	UPLOAD	50	1	5	2.21	23735488.81	1	0
23	UPLOAD	50	5	5	6.96	10346423.68	5	0
24	UPLOAD	50	50	5	105.61	707465.09	50	0
25	UPLOAD	100	1	5	5.10	20572731.26	1	0
26	UPLOAD	100	5	5	17.67	7354552.03	5	0
27	UPLOAD	100	50	5	216.05	1348853.69	50	0
28	DOWNLOAD	10	1	5	0.12	85489929.31	1	0
29	DOWNLOAD	10	5	5	0.32	42284023.28	5	0
30	DOWNLOAD	10	50	5	2.54	8391539.51	50	0
31	DOWNLOAD	50	1	5	0.90	58312190.34	1	0
32	DOWNLOAD	50	5	5	2.43	28224368.70	5	0
33	DOWNLOAD	50	50	5	19.76	5841232.14	50	0
34	DOWNLOAD	100	1	5	2.44	42970637.93	1	0
35	DOWNLOAD	100	5	5	6.89	20284029.37	5	0
36	DOWNLOAD	100	50	5	57.89	3926868.14	50	0
37	UPLOAD	10	1	50	0.37	28113671.30	1	0
38	UPLOAD	10	5	50	1.17	12348377.38	5	0
39	UPLOAD	10	50	50	12.78	1344164.78	50	0
40	UPLOAD	50	1	50	35.06	1495340.23	1	0
41	UPLOAD	50	5	50	7.45	9796052.23	5	0
42	UPLOAD	50	50	50	69.98	1554848.67	50	0

43	UPLOAD	100	1	50	5.52	18979161.94	1	0
44	UPLOAD	100	5	50	53.30	2005655.48	5	0
45	UPLOAD	100	50	50	228.01	962580.35	50	0
46	DOWNLOAD	10	1	50	0.14	74227505.59	1	0
47	DOWNLOAD	10	5	50	0.32	41922256.62	5	0
48	DOWNLOAD	10	50	50	2.57	8862071.66	50	0
49	DOWNLOAD	50	1	50	0.87	60035214.16	1	0
50	DOWNLOAD	50	5	50	2.43	28479893.81	5	0
51	DOWNLOAD	50	50	50	19.90	5778995.61	50	0
52	DOWNLOAD	100	1	50	2.44	42991795.68	1	0
53	DOWNLOAD	100	5	50	6.96	20160308.00	5	0
54	DOWNLOAD	100	50	50	57.76	4041897.70	50	0

Stress test pada server multiprocessing dengan pool menunjukkan bahwa waktu rata-rata umumnya lebih rendah dibandingkan server multithreading, terutama pada jumlah client worker rendah hingga sedang (1-5 clients), tetapi meningkat signifikan dengan bertambahnya jumlah client worker (50 clients) dan ukuran file (100 MB). Throughput tertinggi dicapai pada jumlah client worker rendah (1-5) dengan 5 server worker, tetapi menurun drastis pada 50 client worker, menunjukkan adanya bottleneck pada sumber daya server atau overhead pengelolaan proses. Tidak ada kegagalan klien (Client Fail = 0) di semua skenario, termasuk pada beban tinggi (50 client worker dan 50 server worker), menunjukkan stabilitas yang lebih baik dibandingkan multithreading. Peningkatan server worker dari 1 ke 5 meningkatkan performa secara signifikan, tetapi menggunakan 50 server worker sering kali menurunkan performa, terutama untuk upload, karena overhead pembuatan dan pengelolaan proses yang besar.