Nama: Nabil Julian Syah
NRP: 5025231023
Github: https://github.com/Bibiing/progjar/tree/master/TugasETS

Untuk file generator saya menggunakan

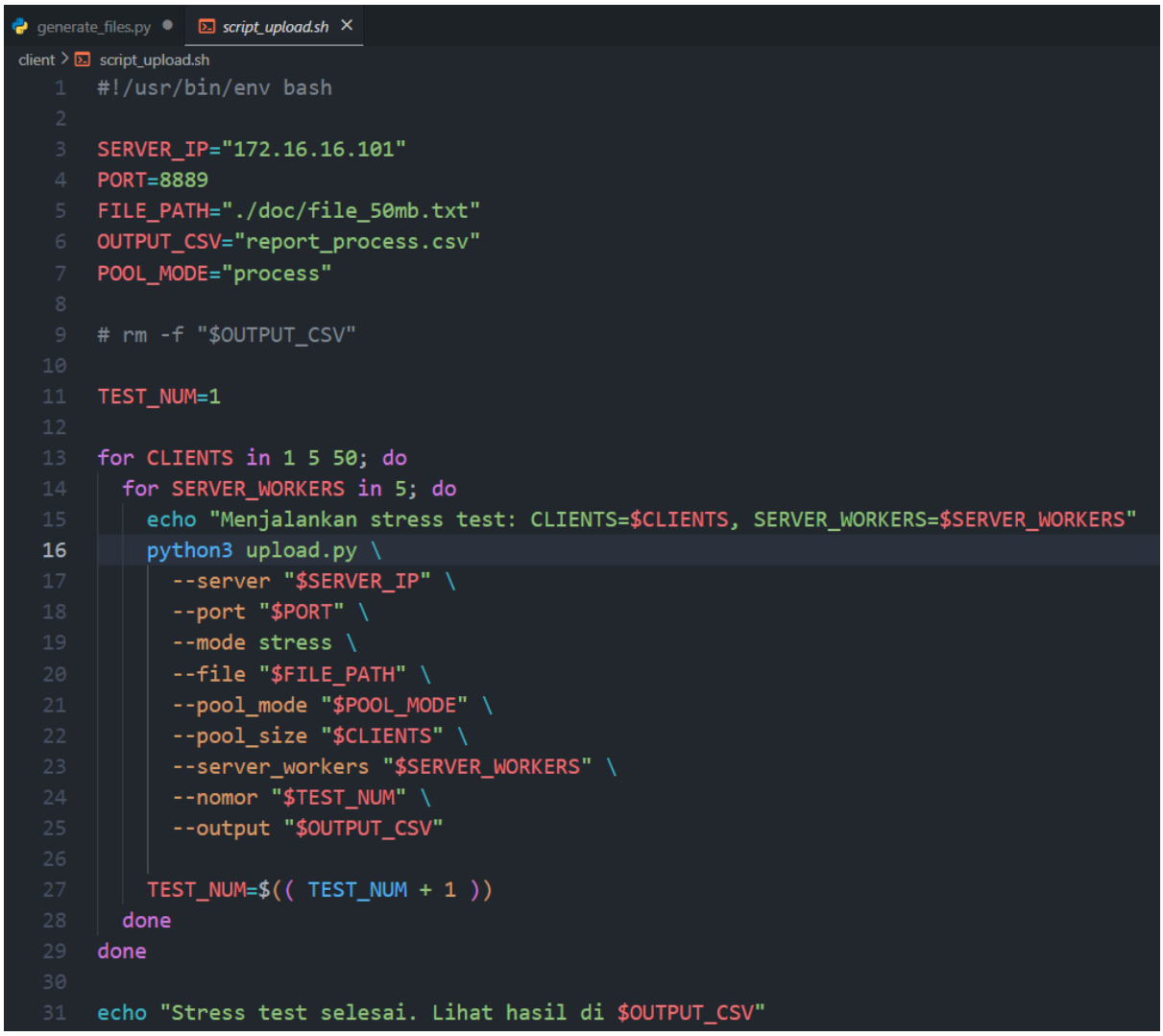
generate_files.py

client > generate_files.py > ...

```python
import os
import base64

def generate_b64_file(directory, filename, size_in_mb):
    os.makedirs(directory, exist_ok=True)
    filepath = os.path.join(directory, filename)

    size_in_bytes = size_in_mb * 1024 * 1024

    # 1. Buat data biner acak
    random_binary_data = os.urandom(size_in_bytes)

    # 2. Encode data biner tersebut ke Base64
    encoded_data = base64.b64encode(random_binary_data)

    # 3. Simpan data yang sudah di-encode ke dalam file
    with open(filepath, 'wb') as f:
        f.write(encoded_data)

    print(f"File Base64 '{filepath}' ({size_in_mb}MB data asli) berhasil dibuat.")

generate_b64_file("doc", "file_10mb.txt", 10)
generate_b64_file("doc", "file_50mb.txt", 50)
generate_b64_file("doc", "file_100mb.txt", 100)
```

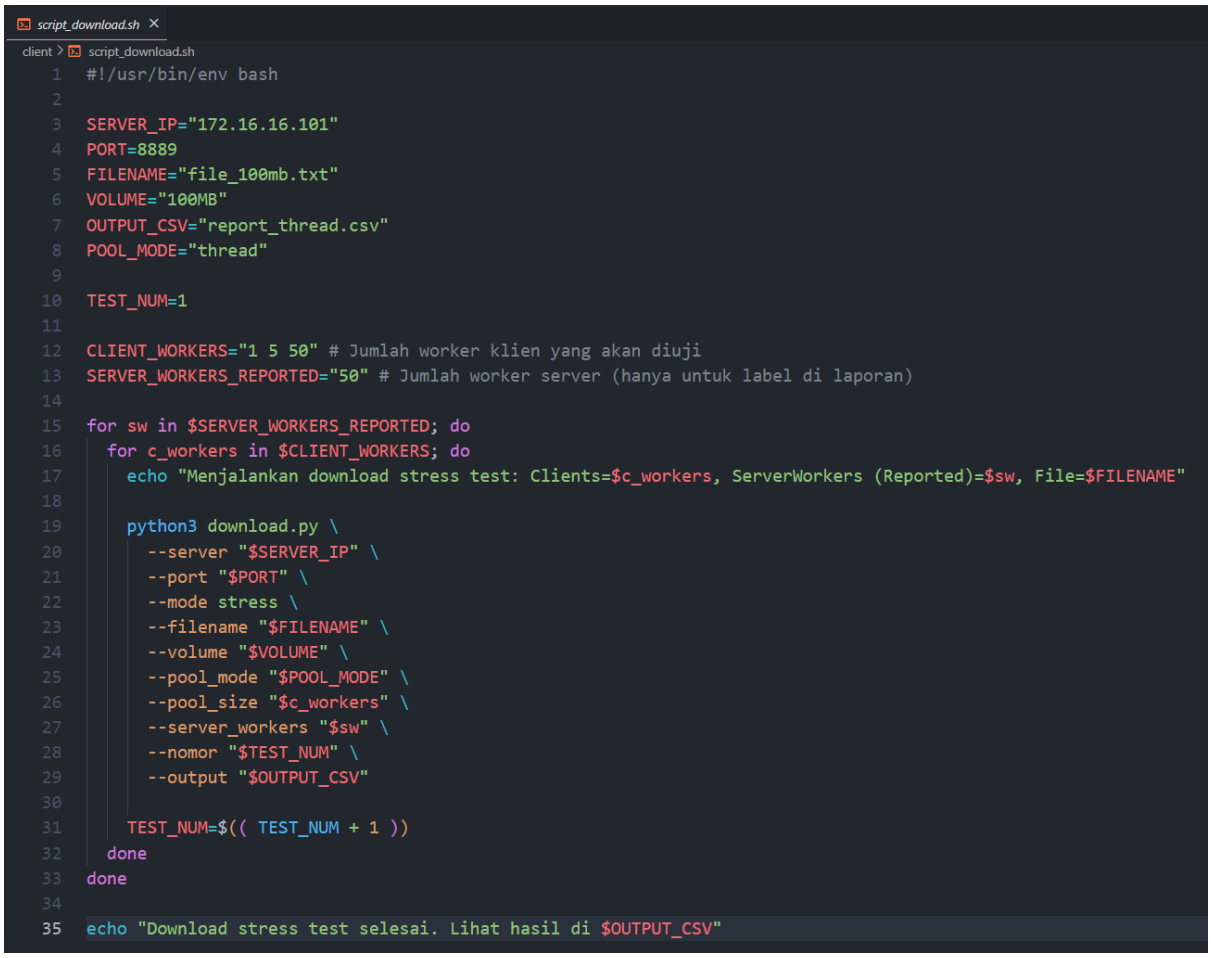
generate_file akan membuat file txt denga isi konten adalah base64 dan ukuran 10mb, 50mb, dan 100mb jika pada client akan di simpan dalam doc/

untuk menjalankan testing di client saya sudah menambahkan bash .sh agar mudah running (testing). Berikut merupakan contoh untuk upload



pada client dengan menjalankan sh script_upload.sh akan menjalankan code upload.py dengan parameter yang ditentukan. Untuk itu kita perlu mengubah secara manual seperti FILE_PATH, POOL_MODE (thread atau process), SERVER_WORKERS (jumlah worker yang berjalan di server side). Semua hasil output akan dicatat pada OUTPUT_CSV

Sedangkan untuk download



```bash
#!/usr/bin/env bash

SERVER_IP="172.16.16.101"
PORT=8889
FILENAME="file_100mb.txt"
VOLUME="100MB"
OUTPUT_CSV="report_thread.csv"
POOL_MODE="thread"

TEST_NUM=1

CLIENT_WORKERS="1 5 50" # Jumlah worker klien yang akan diuji
SERVER_WORKERS_REPORTED="50" # Jumlah worker server (hanya untuk label di laporan)

for sw in $SERVER_WORKERS_REPORTED; do
  for c_workers in $CLIENT_WORKERS; do
    echo "Menjalankan download stress test: Clients=$c_workers, ServerWorkers (Reported)=$sw, File=$FILENAME"

    python3 download.py \
      --server "$SERVER_IP" \
      --port "$PORT" \
      --mode stress \
      --filename "$FILENAME" \
      --volume "$VOLUME" \
      --pool_mode "$POOL_MODE" \
      --pool_size "$c_workers" \
      --server_workers "$sw" \
      --nomor "$TEST_NUM" \
      --output "$OUTPUT_CSV"

    TEST_NUM=$(( TEST_NUM + 1 ))
  done
done

echo "Download stress test selesai. Lihat hasil di $OUTPUT_CSV"
```

Sama saja seperti script sebelumnya, disini kita perlu mengubah FILENAME, VOLUME, POOL_MODE dan SERVER_WORKERS. Script ini akan menjalankan download.py dengan parameter-parameternya

Arsitektur server dirancang unttuk berjalan dalam dua mode konkurensi yang berbeda untuk perbandingan

1. Model Multitrhreading
   Saat server dijalankan, ia mem-parsing argumen seperti host, port, dan jumlah worker. Server kemudian menginisialisasi ThreadPoolExecutor dengan jumlah worker yang ditentukan.

   Fungsi handle_client menerima dua jenis perintah utama: UPLOAD dan GET.
   - **Upload**: Untuk mengatasi pengiriman file besar, server menggunakan metode *streaming* pada fungsi handle_upload_streaming. Metode ini bekerja dengan menerima data dalam potongan kecil, langsung men-decode-nya dari Base64, dan menulisnya ke disk. Hal ini mencegah penggunaan memori yang berlebihan di server.
   - **Download**: Fungsi handle_get menangani permintaan unduh dari klien.

2. Model Multiprocessing
   Model ini menggunakan ProcessPoolExecutor dari library multiprocessing Python untuk membuat proses-proses pekerja yang terisolasi

   Perbedaan Utama: Logika penanganan unduh dan unggah di sisi klien tetap sama. Perbedaan utamanya terletak pada bagaimana server mengelola konkurensi dibandingkan dengan versi threading

MultyThreadPool

Langkah pertama sebelum server dijalankan, args akan di parse menjadi beberapa bagian seperti host, port, jumlah worker, storage, dan server.log. Setelah itu akan melakukan proses login, yang diikuti setelahnya start server dengan args yang di parse sebelumnya.

```
def start_server(host, port, workers, storage_dir):
    os.makedirs(storage_dir, exist_ok=True)
    with ThreadPoolExecutor(max_workers=workers, thread_name_prefix='Worker') as executor:
        try:
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
                server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
                server_socket.bind((host, port))
                server_socket.listen(100)
                logging.info(f"Scalable Server listening on {host}:{port} with {workers} workers")
                while True:
                    conn, addr = server_socket.accept()
                    executor.submit(handle_client, conn, addr, storage_dir)
        except KeyboardInterrupt:
            logging.info("Shutdown signal received.")
        except Exception as e:
            logging.error(f"Server main loop error: {e}", exc_info=True)
        finally:
            logging.info("Server has been shut down.")
```

Pada fungsi handle client, server menerima 2 request antara UPLOAD atau GET

```
if command == "UPLOAD":
    handle_upload_streaming(conn, addr, parts, filepath, initial_payload)
elif command == "GET":
    handle_get(conn, addr, filepath)
else:
    logging.warning(f"Unknown command '{command}' from {addr}.")
    conn.sendall(b"ERROR Unknown command\r\n\r\n")
```

Untuk handle upload, karna menggunakan case file yang cukup besar, disini saya pada fungsi handle_upload_streaming akan mengatasi hal tersebut. Agar tidak memakan memori yang cukup besar. Fungsi tersebut menerima sebagian kecil data, decode data tersebut, dan tuliskan ke file. Sedangkan untuk fungsi handle_get masih sama seperti pada tugas 3.

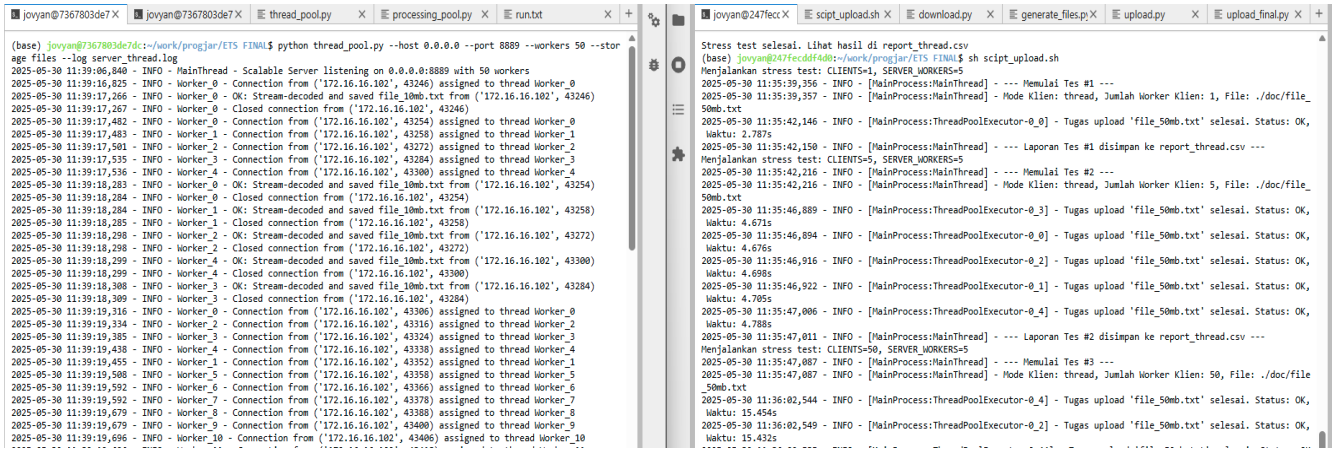


Figure 1Runing MultiThreadPool Upload

```
6w0wqWlsaf8Eeuijt2ylRkvMRJvKOWylNZvm9d7EwzohJpTU7Xzk/HKoCv52CcgxaPWx69QmuzQ8qAi
1774 2025-05-25 13:41:12,119 - INFO - ThreadPoolExecutor-0_21 - Closed connection from ('172.16.16.102', 60528)
1775 2025-05-25 13:41:12,248 - ERROR - ThreadPoolExecutor-0_13 - Error saving file_50mb.bin: Incorrect padding
1776 2025-05-25 13:41:12,328 - INFO - ThreadPoolExecutor-0_12 - Saved file file_50mb.bin
1777 2025-05-25 13:41:12,377 - DEBUG - ThreadPoolExecutor-0_16 - Received data: upload file_50mb.bin
     6w0wqWlsaf8Eeuijt2ylRkvMRJvKOWylNZvm9d7EwzohJpTU7Xzk/HKoCv52CcgxaPWx69QmuzQ8qAi
1778 2025-05-25 13:41:12,384 - DEBUG - ThreadPoolExecutor-0_4 - Received data: upload file_50mb.bin
     6w0wqWlsaf8Eeuijt2ylRkvMRJvKOWylNZvm9d7EwzohJpTU7Xzk/HKoCv52CcgxaPWx69QmuzQ8qAi
1779 2025-05-25 13:41:12,385 - DEBUG - ThreadPoolExecutor-0_6 - Received data: upload file_50mb.bin
     6w0wqWlsaf8Eeuijt2ylRkvMRJvKOWylNZvm9d7EwzohJpTU7Xzk/HKoCv52CcgxaPWx69QmuzQ8qAi
1780 2025-05-25 13:41:12,422 - ERROR - ThreadPoolExecutor-0_1 - Error saving file_50mb.bin: Incorrect padding
1781 2025-05-25 13:41:12,544 - INFO - ThreadPoolExecutor-0_10 - Connection from ('172.16.16.102', 60552)
1782 2025-05-25 13:41:12,560 - INFO - ThreadPoolExecutor-0_14 - Closed connection from ('172.16.16.102', 60436)
1783 2025-05-25 13:41:12,569 - INFO - ThreadPoolExecutor-0_0 - Saved file file_50mb.bin
1784 2025-05-25 13:41:12,577 - DEBUG - ThreadPoolExecutor-0_8 - Received data: upload file_50mb.bin
```

Figure 2 error upload server

```
(base) jovyan@247fecddf4d0:~/work/progjar/ETS FINAL$ sh scipt_upload.sh
Menjalankan stress test: CLIENTS=1, SERVER_WORKERS=5
2025-05-30 11:36:52,494 - INFO - [MainProcess:MainThread] - --- Memulai Tes #1 ---
2025-05-30 11:36:52,494 - INFO - [MainProcess:MainThread] - Mode Klien: thread, Jumlah Worker Klien: 1, File: ./doc/file_
100mb.txt
2025-05-30 11:36:58,765 - INFO - [MainProcess:ThreadPoolExecutor-0_0] - Tugas upload 'file_100mb.txt' selesai. Status: OK
, Waktu: 6.269s
2025-05-30 11:36:58,770 - INFO - [MainProcess:MainThread] - --- Laporan Tes #1 disimpan ke report_thread.csv ---
Menjalankan stress test: CLIENTS=5, SERVER_WORKERS=5
2025-05-30 11:36:58,841 - INFO - [MainProcess:MainThread] - --- Memulai Tes #2 ---
2025-05-30 11:36:58,841 - INFO - [MainProcess:MainThread] - Mode Klien: thread, Jumlah Worker Klien: 5, File: ./doc/file_
100mb.txt
2025-05-30 11:37:08,514 - INFO - [MainProcess:ThreadPoolExecutor-0_3] - Tugas upload 'file_100mb.txt' selesai. Status: OK
, Waktu: 9.669s
2025-05-30 11:37:08,546 - INFO - [MainProcess:ThreadPoolExecutor-0_0] - Tugas upload 'file_100mb.txt' selesai. Status: OK
, Waktu: 9.702s
2025-05-30 11:37:08,588 - INFO - [MainProcess:ThreadPoolExecutor-0_1] - Tugas upload 'file_100mb.txt' selesai. Status: OK
, Waktu: 9.744s
2025-05-30 11:37:08,656 - INFO - [MainProcess:ThreadPoolExecutor-0_2] - Tugas upload 'file_100mb.txt' selesai. Status: OK
, Waktu: 9.812s
2025-05-30 11:37:08,675 - INFO - [MainProcess:ThreadPoolExecutor-0_4] - Tugas upload 'file_100mb.txt' selesai. Status: OK
, Waktu: 9.830s
2025-05-30 11:37:08,683 - INFO - [MainProcess:MainThread] - --- Laporan Tes #2 disimpan ke report_thread.csv ---
Menjalankan stress test: CLIENTS=50, SERVER_WORKERS=5
2025-05-30 11:37:08,753 - INFO - [MainProcess:MainThread] - --- Memulai Tes #3 ---
2025-05-30 11:37:08,754 - INFO - [MainProcess:MainThread] - Mode Klien: thread, Jumlah Worker Klien: 50, File: ./doc/file
_100mb.txt
Killed
Stress test selesai. Lihat hasil di report_thread.csv
```

Figure 3 error upload client

Saat pengujian upload dilakukan dengan worker klien dalam jumlah besar (misalnya 50 klien), terjadi error di sisi klien dengan pesan Killed. Hal ini menunjukkan bahwa proses klien kehabisan memori atau sumber daya lain dan dihentikan paksa oleh sistem operasi. Kasus ini tidak tercatat dalam file CSV karena prosesnya berhenti secara abnormal.Jika server mendapatkan command GET dari client itu yang berarti client mencoba untuk mendowload file.

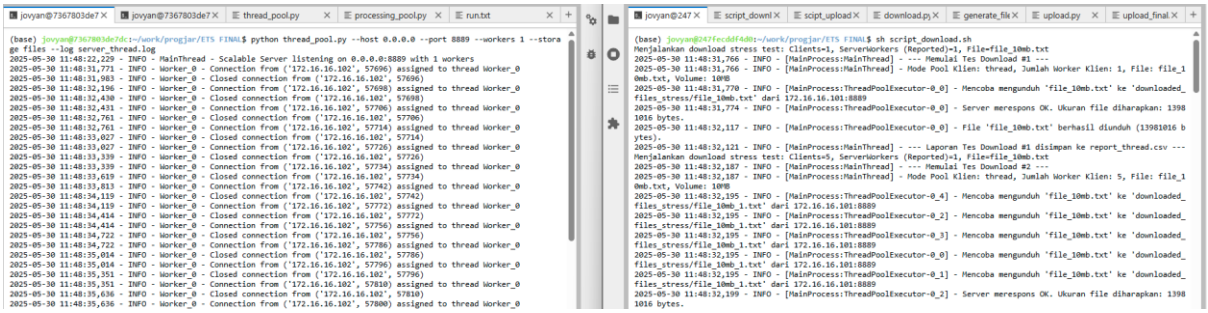


Figure 4ThreadPool Download

Dari gambar diatas adalah contoh saat client mencoba download dengan berbagai macam kondisi seperti jumlah worker di client/server dan filesize.

Berikut adalah hasil dari multithreading

Nomor	Operasi	Volume	Jumlah client worker	Jumlah server worker	Waktu total per client	Throughput per client	Jumlah worker client yang sukses dan gagal	Jumlah worker server yang sukses dan gagal
1	Upload	13.33MB	1	1	0.627s	21772.08 KB/s	1 sukses, 0 gagal	Periksa Log Server
2	Upload	13.33MB	5	1	1.585s	8612.26 KB/s	5 sukses, 0 gagal	Periksa Log Server
3	Upload	13.33MB	50	1	13.843s	986.3 KB/s	50 sukses, 0 gagal	Periksa Log Server
4	Upload	66.67MB	1	1	3.064s	22282.78 KB/s	1 sukses, 0 gagal	Periksa Log Server
5	Upload	66.67MB	5	1	7.700s	8866.21 KB/s	5 sukses, 0 gagal	Periksa Log Server
6	Upload	66.67MB	50	1	68.083s	1223.92 KB/s	27 sukses, 23 gagal	Periksa Log Server
7	Upload	133.33MB	1	1	5.788s	23588.4 KB/s	1 sukses, 0 gagal	Periksa Log Server
8	Upload	133.33MB	5	1	15.376s	8879.78 KB/s	5 sukses, 0 gagal	Periksa Log Server
9	Upload	13.33MB	1	5	0.524s	26079.65 KB/s	1 sukses, 0 gagal	Periksa Log Server
10	Upload	13.33MB	5	5	0.978s	13957.46 KB/s	5 sukses, 0 gagal	Periksa Log Server
11	Upload	13.33MB	50	5	6.228s	2192.22 KB/s	50 sukses, 0 gagal	Periksa Log Server
12	Upload	66.67MB	1	5	2.787s	24490.49 KB/s	1 sukses, 0 gagal	Periksa Log Server
13	Upload	66.67MB	5	5	4.708s	14501.66 KB/s	5 sukses, 0 gagal	Periksa Log Server
14	Upload	66.67MB	50	5	33.712s	2025.02 KB/s	50 sukses, 0 gagal	Periksa Log Server
15	Upload	133.33MB	1	5	6.269s	21780.56 KB/s	1 sukses, 0 gagal	Periksa Log Server
16	Upload	133.33MB	5	5	9.752s	14001.05 KB/s	5 sukses, 0 gagal	Periksa Log Server
17	Upload	13.33MB	1	50	0.525s	26007.63 KB/s	1 sukses, 0 gagal	Periksa Log Server
18	Upload	13.33MB	5	50	0.952s	14341.23 KB/s	5 sukses, 0 gagal	Periksa Log Server
19	Upload	13.33MB	50	50	8.047s	1696.76 KB/s	50 sukses, 0 gagal	Periksa Log Server
20	Upload	66.67MB	1	50	2.627s	25989.77 KB/s	1 sukses, 0 gagal	Periksa Log Server
21	Upload	66.67MB	5	50	4.884s	13978.3 KB/s	5 sukses, 0 gagal	Periksa Log Server
22	Upload	66.67MB	50	50	44.574s	1531.52 KB/s	50 sukses, 0 gagal	Periksa Log Server
23	Upload	133.33MB	1	50	5.785s	23601.03 KB/s	1 sukses, 0 gagal	Periksa Log Server
24	Upload	133.33MB	5	50	9.798s	13934.37 KB/s	5 sukses, 0 gagal	Periksa Log Server
25	Download	10MB	1	1	0,351	38907,32	1	0
26	Download	10MB	5	1	0,968	14100,63	5	0
27	Download	10MB	50	1	7,961	1714,94	50	0
28	Download	50MB	1	1	1,913	35679,36	1	0
29	Download	50MB	5	1	5,156	13240,97	5	0
30	Download	50MB	50	1	39,238	2138,1	37	13
31	Download	100MB	1	1	0,011	0	1	0
32	Download	100MB	5	1	0,022	0	5	0
33	Download	100MB	50	1	0,162	0	50	0
34	Download	10MB	1	5	0,35	39062,84	1	0
35	Download	10MB	5	5	0,73	18713,74	5	0
36	Download	10MB	50	5	4,078	3348,07	50	0
37	Download	50MB	1	5	1,81	37715,2	1	0
38	Download	50MB	5	5	3,647	18718,24	5	0
39	Download	50MB	50	5	20,038	3406,82	50	0
40	Download	100MB	1	5	0,014	0	1	0
41	Download	100MB	5	5	0,015	0	5	0
42	Download	100MB	50	5	0,048	0	50	0
43	Download	10MB	1	50	0,37	36931,05	1	0
44	Download	10MB	5	50	0,715	19091,03	5	0
45	Download	10MB	50	50	7,149	1909,84	50	0
46	Download	50MB	1	50	1,775	38470,14	1	0
47	Download	50MB	5	50	3,608	18923,15	5	0
48	Download	50MB	50	50	35,648	1915	50	0
49	Download	100MB	1	50	0,019	0	1	0
50	Download	100MB	5	50	0,018	0	5	0
51	Download	100MB	50	50	0,07	0	50	0

Kenapa tidak 54? Karna disini program yang saya buat, tidak mencatat Ketika error terjadi.

MultyProcessing

Pada multiprocessing pool ini akan menggunakan library python multiptocessing. Process untuk menjalan multiprocess. Disini akan dibuat queue (antrian) untuk worker, diambil dari multiprocessing.Queue()

Untuk Download logika yang digunakan sama seperti multithread karna untuk download filenya sama. Yang membedakan hanya deserver saja (processing_pool untuk process dan thread_pool untuk threading). Download process terletak di client pada fie download_files

Sedangkan untuk upload multiprocessing sama seperti logika sebelumnya karna filenya sama.

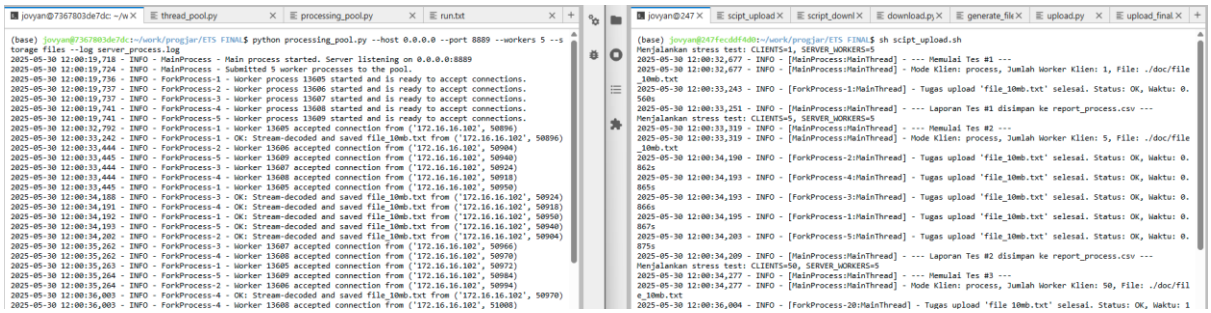


Figure 5ProcessingPool Upload

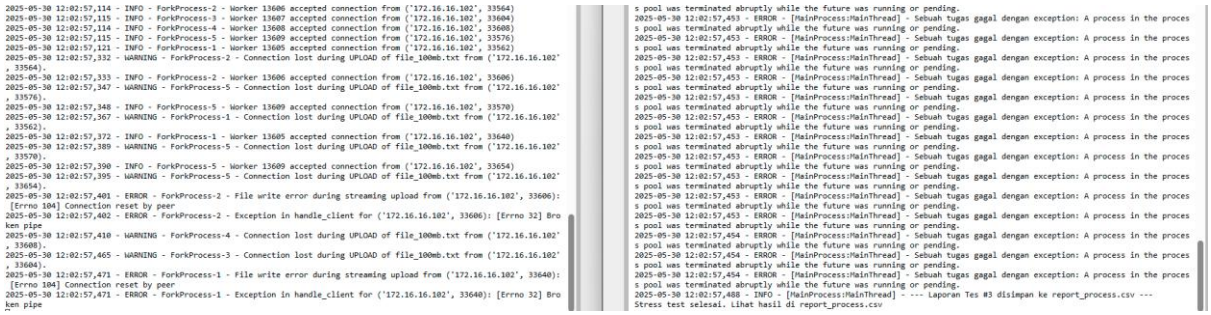


Figure 6Processing Upload Error

Saat pengujian upload menggunakan ProcessPoolExecutor, baik server maupun klien menunjukkan banyak error. Di server, error Connection lost dan Connection reset by peer sering terjadi. Di klien, error yang muncul adalah A process in the process pool was terminated abruptly.

Error-error ini mengindikasikan ketidakstabilan yang parah saat menggunakan model multi-proses untuk tugas yang sangat I/O-bound seperti ini, di mana manajemen koneksi antar-proses menjadi lebih kompleks dan rentan gagal.

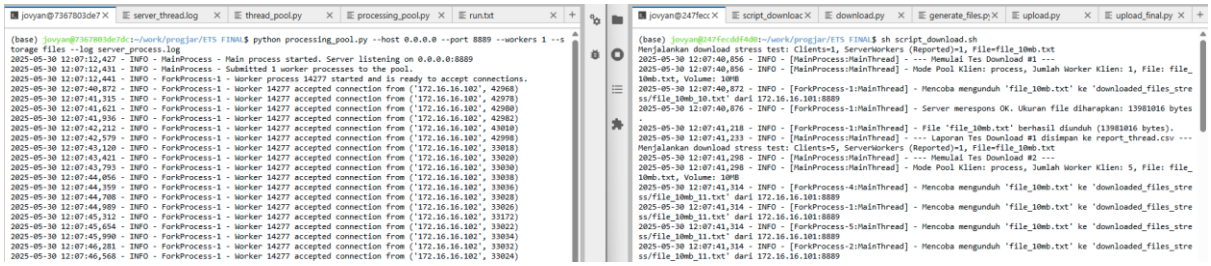


Figure 7ProcessingPool Download

Hasil report dari Multiprocess

Nomor	Operasi	Volume	Jumlah client worker	Jumlah server worker	Waktu total per client	Throughput per client	Jumlah worker client yang sukses dan gagal	Jumlah worker server yang sukses dan gagal
1	process	13.33MB	1	1	0.556s	24542.36 KB/s	1 sukses, 0 gagal	Periksa Log Server
2	process	13.33MB	5	1	1.476s	9248.78 KB/s	5 sukses, 0 gagal	Periksa Log Server
3	process	13.33MB	50	1	12.777s	1068.57 KB/s	50 sukses, 0 gagal	Periksa Log Server
4	process	66.67MB	1	1	2.800s	24384.97 KB/s	1 sukses, 0 gagal	Periksa Log Server
5	process	66.67MB	5	1	7.194s	9489.25 KB/s	5 sukses, 0 gagal	Periksa Log Server
6	process	66.67MB	50	1	53.649s	1676.12 KB/s	27 sukses, 23 gagal	Periksa Log Server
7	process	133.33MB	1	1	6.156s	22179.55 KB/s	1 sukses, 0 gagal	Periksa Log Server
8	process	133.33MB	5	1	14.522s	9402.15 KB/s	5 sukses, 0 gagal	Periksa Log Server
9	process	133.33MB	50	1	300.000s	0.0 KB/s	0 sukses, 50 gagal	Periksa Log Server
10	process	13.33MB	1	5	0.560s	24377.93 KB/s	1 sukses, 0 gagal	Periksa Log Server
11	process	13.33MB	5	5	0.867s	15745.29 KB/s	5 sukses, 0 gagal	Periksa Log Server
12	process	13.33MB	50	5	5.171s	2640.58 KB/s	50 sukses, 0 gagal	Periksa Log Server
13	process	66.67MB	1	5	2.879s	23710.38 KB/s	1 sukses, 0 gagal	Periksa Log Server
14	process	66.67MB	5	5	4.181s	16326.79 KB/s	5 sukses, 0 gagal	Periksa Log Server
15	process	66.67MB	50	5	26.713s	2555.55 KB/s	50 sukses, 0 gagal	Periksa Log Server
16	process	133.33MB	1	5	5.743s	23774.11 KB/s	1 sukses, 0 gagal	Periksa Log Server
17	process	133.33MB	5	5	8.692s	15708.3 KB/s	5 sukses, 0 gagal	Periksa Log Server
18	process	133.33MB	50	5	300.000s	0.0 KB/s	0 sukses, 50 gagal	Periksa Log Server
19	process	13.33MB	1	50	0.619s	22074.56 KB/s	1 sukses, 0 gagal	Periksa Log Server
20	process	13.33MB	5	50	0.848s	16094.01 KB/s	5 sukses, 0 gagal	Periksa Log Server
21	process	13.33MB	50	50	6.071s	2248.87 KB/s	50 sukses, 0 gagal	Periksa Log Server
22	process	66.67MB	1	50	2.744s	24874.71 KB/s	1 sukses, 0 gagal	Periksa Log Server
23	process	66.67MB	5	50	4.225s	16158.5 KB/s	5 sukses, 0 gagal	Periksa Log Server
24	process	66.67MB	50	50	32.666s	2089.86 KB/s	50 sukses, 0 gagal	Periksa Log Server
25	process	133.33MB	1	50	5.464s	24988.21 KB/s	1 sukses, 0 gagal	Periksa Log Server
26	process	133.33MB	5	50	8.677s	15735.93 KB/s	5 sukses, 0 gagal	Periksa Log Server
27	process	133.33MB	50	50	300.000s	0.0 KB/s	0 sukses, 50 gagal	Periksa Log Server
28	process	10MB	1	1	0,358	38088,11	1	0
29	process	10MB	5	1	1,022	13355,12	5	0
30	process	10MB	50	1	8,304	1644,17	50	0
31	process	50MB	1	1	1,711	39908,09	1	0
32	process	50MB	5	1	5,177	13185,4	5	0
33	process	50MB	50	1	38,311	2172,35	38	12
34	process	100MB	1	1	0,028	6976,28	1	0
35	process	100MB	5	1	0,094	2046,16	5	0
36	process	100MB	50	1	1,031	186,28	50	0
37	process	10MB	1	5	0,359	38050,41	1	0
38	process	10MB	5	5	0,699	19533,89	5	0
39	process	10MB	50	5	4,183	3264,28	50	0
40	process	50MB	1	5	1,665	40990,34	1	0
41	process	50MB	5	5	3,423	19944,26	5	0
42	process	50MB	50	5	20,197	3380,12	50	0
43	process	100MB	1	5	0,031	6140,58	1	0
44	process	100MB	5	5	0,044	4379,31	5	0
45	process	100MB	50	5	0,254	755,57	50	0
46	process	10MB	1	50	0,354	38575,26	1	0
47	process	10MB	5	50	0,668	20428,42	5	0
48	process	10MB	50	50	7,252	1882,66	50	0
49	process	50MB	1	50	1,671	40857,88	1	0
50	process	50MB	5	50	3,466	19693,68	5	0
51	process	50MB	50	50	35,441	1926,21	50	0