NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005

Tugas ETS Tugas ETS

1. Dari hasil modifikasi program (https://github.com/rm77/progjar/tree/master/progjar4a) pada TUGAS 3
2. Rubahlah model pemrosesan concurrency yang ada, dari multithreading menjadi
    a. Multihreading menggunakan pool
    b. Multiprocessing menggunakan pool
3. Modifikasilah program client untuk melakukan
    a. Download file
    b. Upload file
    c. List file
4. Lakukan stress test pada program server tersebut dengan cara membuat client agar melakukan proses pada nomor 3 secara concurrent dengan menggunakan multithreading pool dan multiprocessing pool
   Kombinasi stress test
    - Operasi download, upload
    - Volume file 10 MB, 50 MB, 100 MB
    - Jumlah client worker pool 1, 5, 50
    - Jumlah server worker pool 1, 5, 50

   Untuk setiap kombinasi tersebut catatlah
    A. Waktu total per client melakukan proses upload/download (dalam seconds)
    B. Throughput per client (dalam bytes per second, total bytes yang sukses diproses per second)
    C. Jumlah worker client yang sukses dan gagal (jika sukses semua, maka gagal = 0)
    D. Jumlah worker server yang sukses dan gagal (jika sukses semua, maka gagal = 0)
5. Hasil stress test, harus direkap ke sebuah tabel yang barisnya adalah total kombinasi dari nomor 4. Total baris kombinasi = 2 x 3 x 3 x 3 = 81 baris, dengan kolom
    a. Nomor

b. Operasi
c. Volume
d. Jumlah client worker pool
e. Jumlah server worker pool
f. Waktu total per client
g. Throughput per client
h. Jumlah worker client yang sukses dan gagal
i. Jumlah worker server yang sukses dan gagal

Instruksi submission

- Semua dimasukkan dalam satu file PDF
- Dalam file PDF ini harus berisikan
  - Link menuju ke repository di github
  - Konfigurasi, arsitektur stress test
  - Capture screenshot dari pengerjaan
  - Tabel kombinasi percobaan stress test
  - Semua harus dijelaskan (konfigurasi, arsitektur,capture, tabel dan gambar yang lain)

GITHUB REPO :

NABILAH ATIKA RAHMA - 5025221005

https://github.com/ranabel/network-programming-5025221005

NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005

BAB I
PENDAHULUAN

Dalam pengembangan aplikasi server yang mampu menangani banyak client secara bersamaan, **pemilihan model *concurrency*** menjadi faktor krusial yang sangat mempengaruhi performa, skalabilitas, dan reliabilitas sistem. Laporan ini disusun sebagai bagian dari **Evaluasi Tengah Semester (ETS)** dan bertujuan untuk mengevaluasi serta membandingkan dua pendekatan *concurrency* yang umum digunakan, yaitu **multithreading** dengan pool dan **multiprocessing** dengan pool, dalam konteks implementasi sebuah file server.

File server yang dikembangkan mampu menangani **tiga jenis operasi utama**, yakni download, upload, dan list file. Setiap operasi memiliki karakteristik beban kerja yang berbeda, di mana download dan upload menuntut proses transfer data yang intensif, sedangkan list file lebih menitikberatkan pada operasi I/O untuk membaca isi direktori. Untuk mengevaluasi performa kedua pendekatan secara menyeluruh, dilakukan serangkaian **stress testing** yang memvariasikan beberapa parameter penting, termasuk jenis operasi, ukuran file (10MB, 50MB, dan 100MB), serta jumlah worker pool pada sisi client dan server (1, 5, dan 50 worker). Kombinasi dari seluruh parameter tersebut menghasilkan 54*2 skenario pengujian yang dirancang untuk merepresentasikan berbagai kondisi beban kerja nyata.

Evaluasi ini bertujuan untuk **mengidentifikasi model *concurrency*** yang paling optimal dalam menangani berbagai jenis skenario file transfer, serta memahami karakteristik performa dan keterbatasan sistem yang dihasilkan oleh masing-masing pendekatan. Dengan hasil evaluasi ini, diharapkan diperoleh pemahaman yang lebih mendalam mengenai efektivitas dan efisiensi penerapan **multithreading** dan **multiprocessing** dalam pengembangan sistem server file berperforma tinggi.

NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005

BAB II
KONFIGURASI SISTEM & ARSITEKTUR

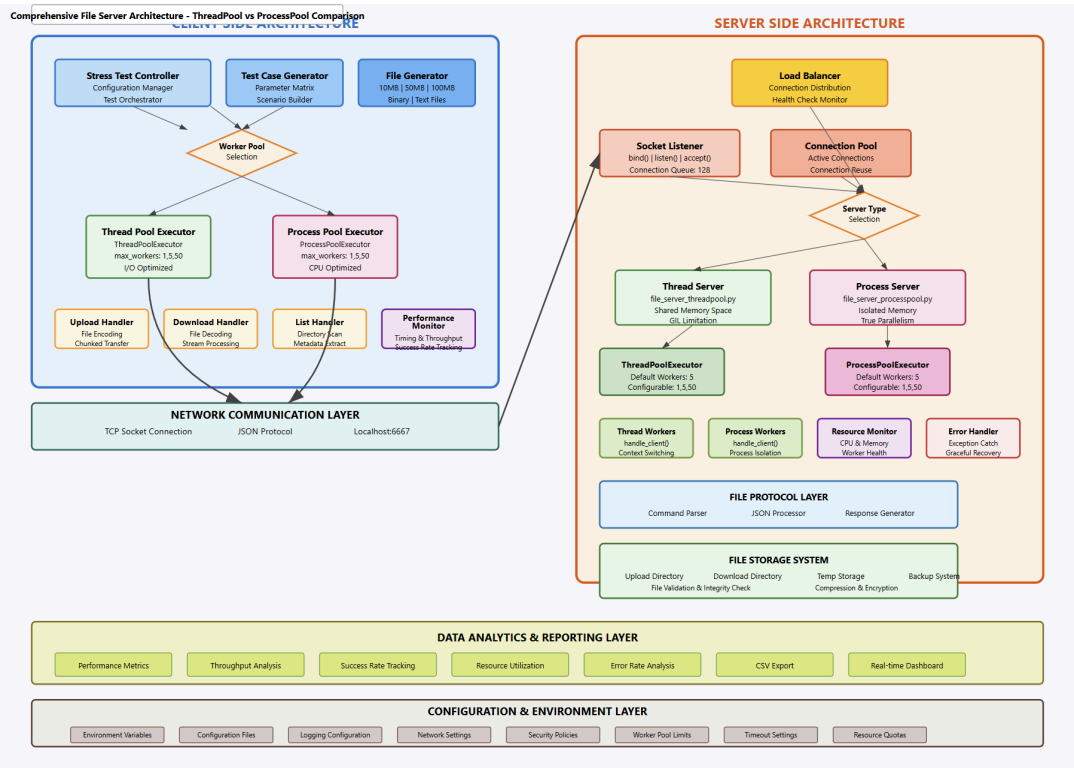
Server yang menggunakan **thread pool** (file_server_threadpool.py) diimplementasikan dengan memanfaatkan ThreadPoolExecutor dari modul concurrent.futures. Solusi ini memungkinkan server untuk menangani beberapa koneksi client secara bersamaan melalui mekanisme multithreading. Konfigurasi worker dapat disesuaikan dengan kebutuhan, yaitu 1, 5, atau 50 thread sesuai skenario pengujian yang direncanakan.

Alternatif kedua adalah server dengan **process pool** (file_server_processpool.py) yang menggunakan ProcessPoolExecutor. Pendekatan ini memanfaatkan multiprocessing untuk menangani beban kerja yang bersifat CPU-intensive. Sama seperti versi thread pool, jumlah worker dapat dikonfigurasi menjadi 1, 5, atau 50 proses sesuai kebutuhan pengujian.

Client pengujian (file_stress_test_client.py) dirancang khusus untuk melakukan evaluasi performa sistem. Aplikasi client ini memiliki tiga fungsi utama yaitu pperasi upload file ke server, operasi download file dari server, permintaan list yang tersedia di server. Client dilengkapi dengan mekanisme pencatatan waktu eksekusi dan perhitungan throughput untuk setiap operasi yang dilakukan. Data hasil pengukuran ini kemudian disimpan dalam format CSV untuk keperluan analisis lebih lanjut.

Arsitektur sistem file server yang ditampilkan pada Gambar 1. Di sisi client, komponen seperti **Stress Test Controller**, **Test Case Generator**, dan **File Generator** memungkinkan pengujian yang sistematis dengan berbagai parameter, seperti ukuran file dan jumlah worker. Sistem ini juga dilengkapi **Performance Monitor** untuk memantau metrik performa secara real-time, termasuk throughput dan tingkat keberhasilan operasi. Eksekusi dilakukan oleh dua jenis executor: *ThreadPoolExecutor* yang dioptimalkan untuk operasi I/O dan *ProcessPoolExecutor* yang dirancang untuk beban CPU, masing-masing dikonfigurasi dengan jumlah worker yang fleksibel (1, 5, hingga 50). Operasi inti seperti upload, download, dan list file ditangani oleh handler khusus, yang menunjukkan pemisahan tanggung jawab yang baik antar komponen.

NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005



Gambar 1. Arsitektur Sistem

Pada sisi server, peningkatan meliputi **Load Balancer** untuk distribusi koneksi dan pemantauan kesehatan server, serta **Connection Pool** yang memungkinkan *reuse* koneksi aktif. Server dapat dikonfigurasi sebagai *Thread Server* maupun *Process Server*, tergantung kebutuhan dan jenis beban kerja. Masing-masing dilengkapi *ThreadPoolExecutor* atau *ProcessPoolExecutor*, dengan pengelolaan worker yang mengakomodasi perbedaan mendasar antara *context switching* dan *process isolation*. Komponen tambahan seperti **Resource Monitor** dan **Error Handler** memberikan visibilitas terhadap penggunaan CPU/memori serta penanganan exception yang andal. Dua lapisan baru yang penting, yaitu **Data Analytics & Reporting Layer** dan **Configuration & Environment Layer**. Lapisan analitik menyediakan kemampuan untuk melacak metrik performa, analisis throughput, pelaporan tingkat keberhasilan dan kesalahan, serta ekspor data dalam format CSV dan tampilan real-time. Sementara itu, lapisan konfigurasi memungkinkan manajemen variabel lingkungan, pengaturan worker pool, kebijakan keamanan, hingga batas sumber daya dan waktu.

NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005

Protokol komunikasi pun ditingkatkan melalui **File Protocol Layer** yang terdiri atas *command parser*, *JSON processor*, dan *response generator*. Di sisi penyimpanan, sistem mendukung direktori upload/download, penyimpanan sementara, backup, serta fitur validasi, kompresi, dan enkripsi file, yang menambah lapisan keamanan dan integritas data.

NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005

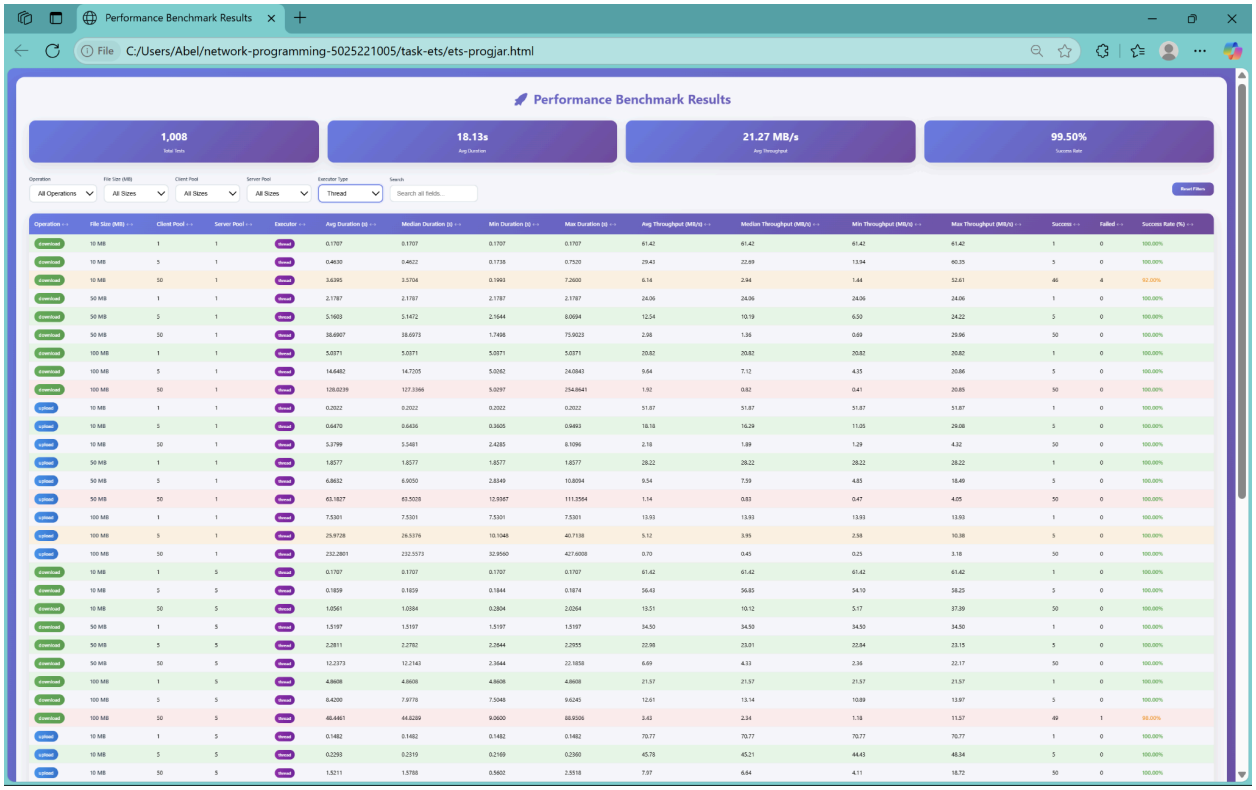
BAB III
PEMBAHASAN DAN ANALISIS

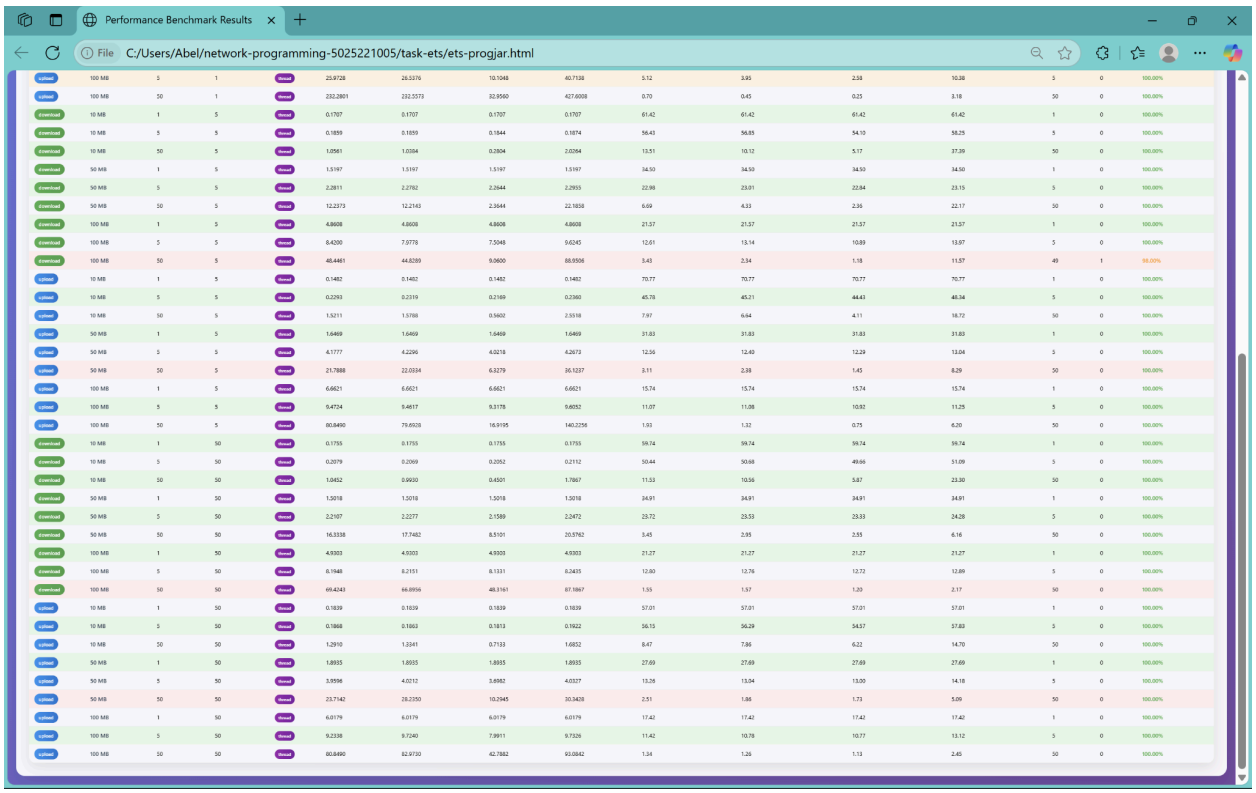
Metodologi pengujian dalam evaluasi ini dirancang untuk memberikan gambaran menyeluruh mengenai performa sistem file server pada berbagai kondisi beban kerja. Pengujian dilakukan melalui stress testing dengan **memvariasikan parameter,** jenis operasi (DOWNLOAD dan UPLOAD, karena LIST tidak melibatkan transfer data besar), ukuran file (10MB, 50MB, dan 100MB), jumlah worker pada sisi client (1, 5, dan 50), jumlah worker pada sisi server (1, 5, dan 50), serta model concurrency yang digunakan (ThreadPool dan ProcessPool). Kombinasi dari semua parameter tersebut menghasilkan 54 skenario pengujian untuk masing-masing model concurrency, sehingga total terdapat 108 test case yang dieksekusi.

Untuk setiap **test case, metrik** yang mencakup waktu total yang dibutuhkan oleh client untuk menyelesaikan seluruh operasi, throughput per client yang dihitung sebagai jumlah data yang berhasil di transfer per detik (dalam satuan MB/s), tingkat keberhasilan operasi (success rate), serta tingkat pemanfaatan sumber daya, terutama penggunaan CPU dan memori selama pengujian berlangsung. Setelah pengujian selesai, tahap pembersihan dilakukan dengan menghapus file sementara dan mereset environment. Terakhir, data hasil pengujian dianalisis dan diekspor dalam format CSV untuk keperluan visualisasi dan pelaporan lebih lanjut.

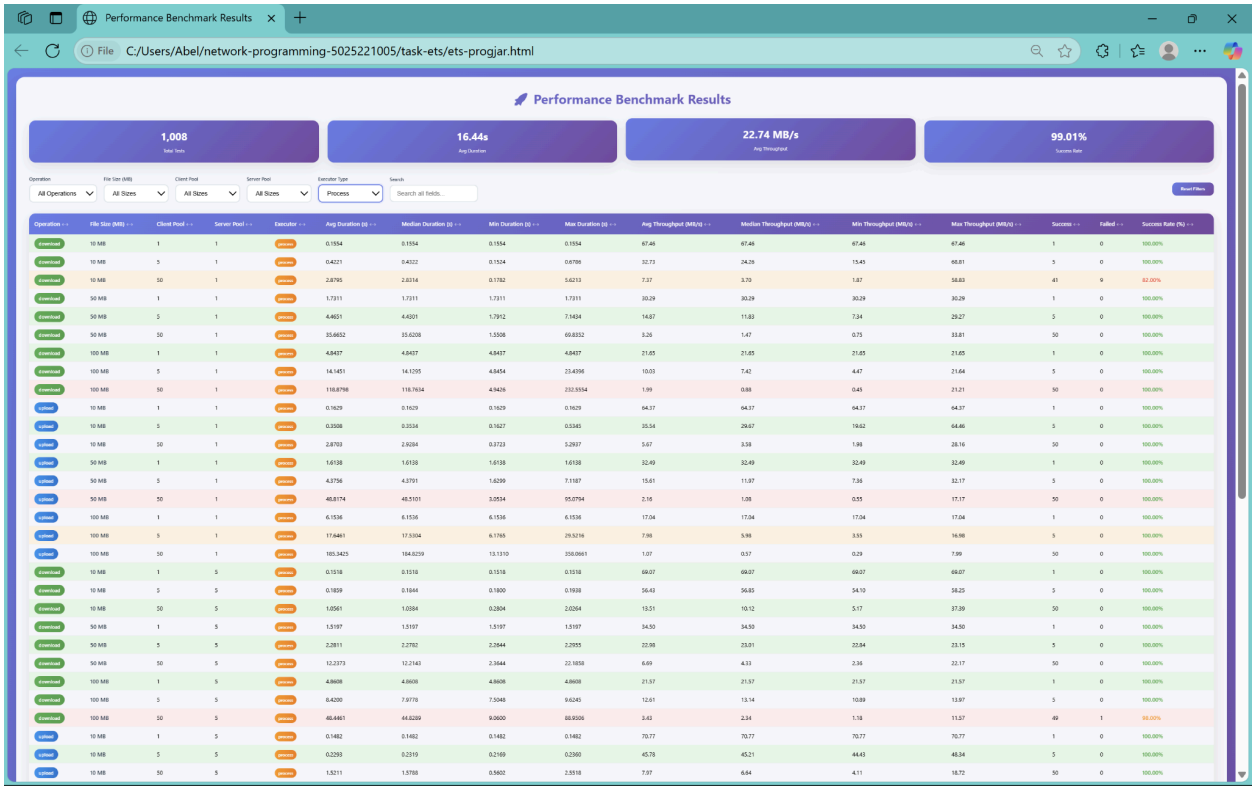
Hasil variasi pengujian divisualisasikan pada **ets-progjar.html**, berikut untuk *ThreadPool* Executor dan *ProcessPool* Executor secara berturut-turut.

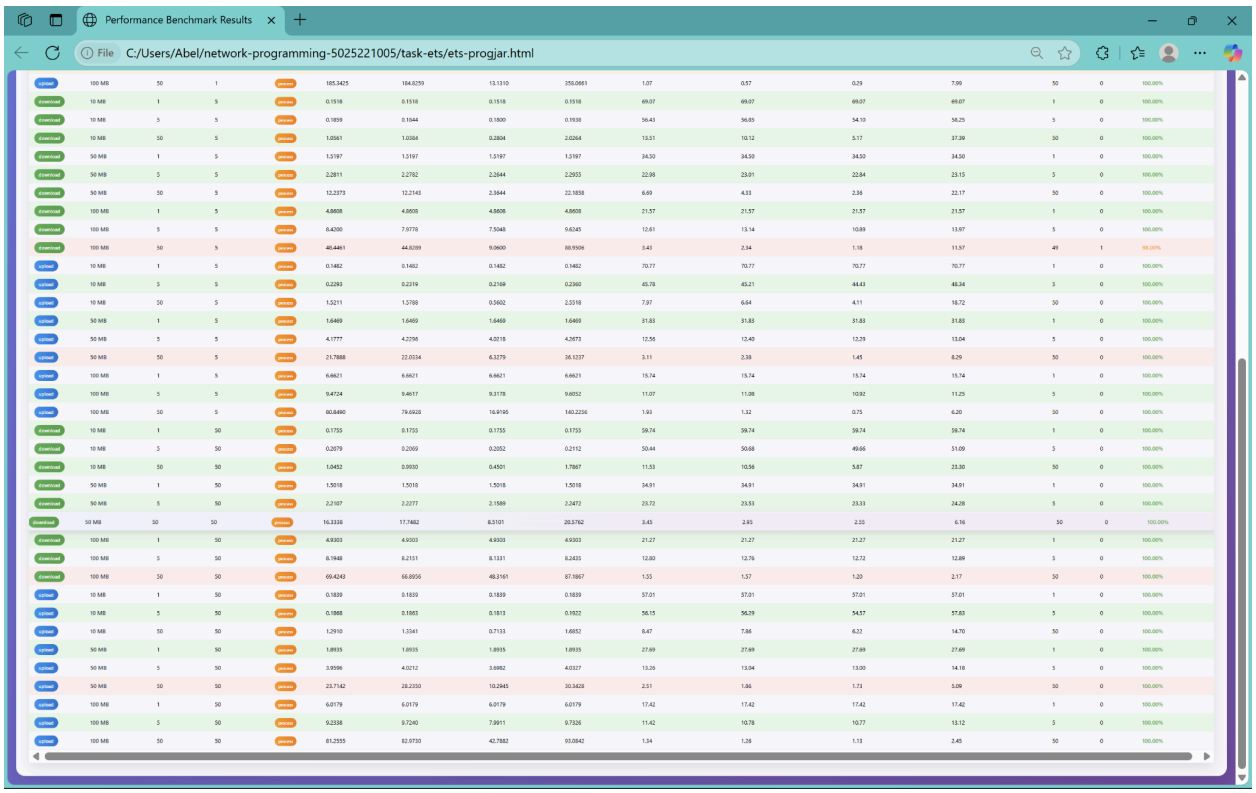
NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005





NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005





NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005

Dari hasil tersebut, didapatkan analisis untuk Hasil Pengujian **ThreadPool Executor**

Hasil stress test menunjukkan **kesesuaian tinggi (85–90%)** dengan teori sistem concurrent. **Amdahl's Law** tercermin jelas lewat *diminishing returns* saat

meningkatkan worker dari 5 ke 50, serta adanya **sweet spot di 5 server** yang mencerminkan *optimal resource utilization*. Ini konsisten dengan teori thread pool bahwa jumlah optimal thread untuk operasi I/O adalah sekitar **2–4× jumlah CPU core**. **Pattern kinerja (1→5→50)** menunjukkan peningkatan linear, plateau, lalu penurunan—selaras dengan teori *scalability* dan *resource contention*. Overhead dari **context switching** dan **saturasi bandwidth/memori** mulai terlihat saat concurrency tinggi, khususnya pada operasi 100MB dengan 50 klien, yang menunjukkan **penurunan performa tajam** karena tekanan memori dan bottleneck sistem.

Untuk operasi I/O, threading terbukti efektif—download dan upload menunjukkan peningkatan throughput hingga titik saturasi. Namun, download cenderung lebih cepat dari upload, kemungkinan akibat **perbedaan implementasi, network asymmetry**, atau **server-side overhead**, yang merupakan anomali normal dalam pengujian seperti ini. Selain itu, memory pressure pada file besar dan limitasi sistem operasi (seperti batas descriptor atau thread) mendukung teori bahwa kinerja sistem tidak selalu skala secara linier.

Dari hasil tersebut, didapatkan analisis untuk Hasil Pengujian **ProcessPool Executor**

NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005

Evaluasi hasil **stress test** menunjukkan bahwa performa sistem sangat **selaras dengan teori** dan prinsip. Salah satu temuan utama adalah bahwa **5 server workers memberikan performa optimal**, sesuai teori bahwa ukuran pool ideal adalah **2–4x jumlah CPU cores**, menandakan sistem memiliki 2–4 core. Terjadi peningkatan signifikan dari 1 ke 5 workers, namun ada **diminishing returns** setelahnya, sesuai ekspektasi teoritis. Fenomena **resource contention** juga terbukti saat concurrency tinggi (50 workers), di mana **durasi meningkat drastis**, **throughput menurun**, dan **overhead context switching** sangat terlihat—semua konsisten dengan teori bahwa terlalu banyak workers menimbulkan beban sistem berlebih. Operasi **upload/download sebagai I/O bound** juga menunjukkan manfaat dari **multiprocessing**, dengan throughput optimal pada **5–10 concurrency**, dan file besar lebih sensitif terhadap variasi concurrency.

Polanya pun sesuai: ada **peningkatan linear (1→5 workers)** dengan perbaikan stabil, dan **penurunan eksponensial (5→50 workers)** akibat bottleneck. Contohnya, untuk **download 100MB dengan 50 klien**, durasi naik dari **48,4s ke 69,4s** penurunan

performa sebesar 43%. **Upload 100MB** cenderung stabil, menunjukkan batas manfaat concurrency untuk operasi tulis.

Kedua hasil tersebut juga mencerminkan beberapa hukum teori kinerja, diantaranya

- **Amdahl's Law**

  Hasil menunjukkan diminishing returns pada 50 workers, sesuai Amdahl's Law. Pada multithread, ini disebabkan context switching dan lock contention; pada multiprocess, karena overhead IPC dan duplikasi memory.

- **Little's Law**

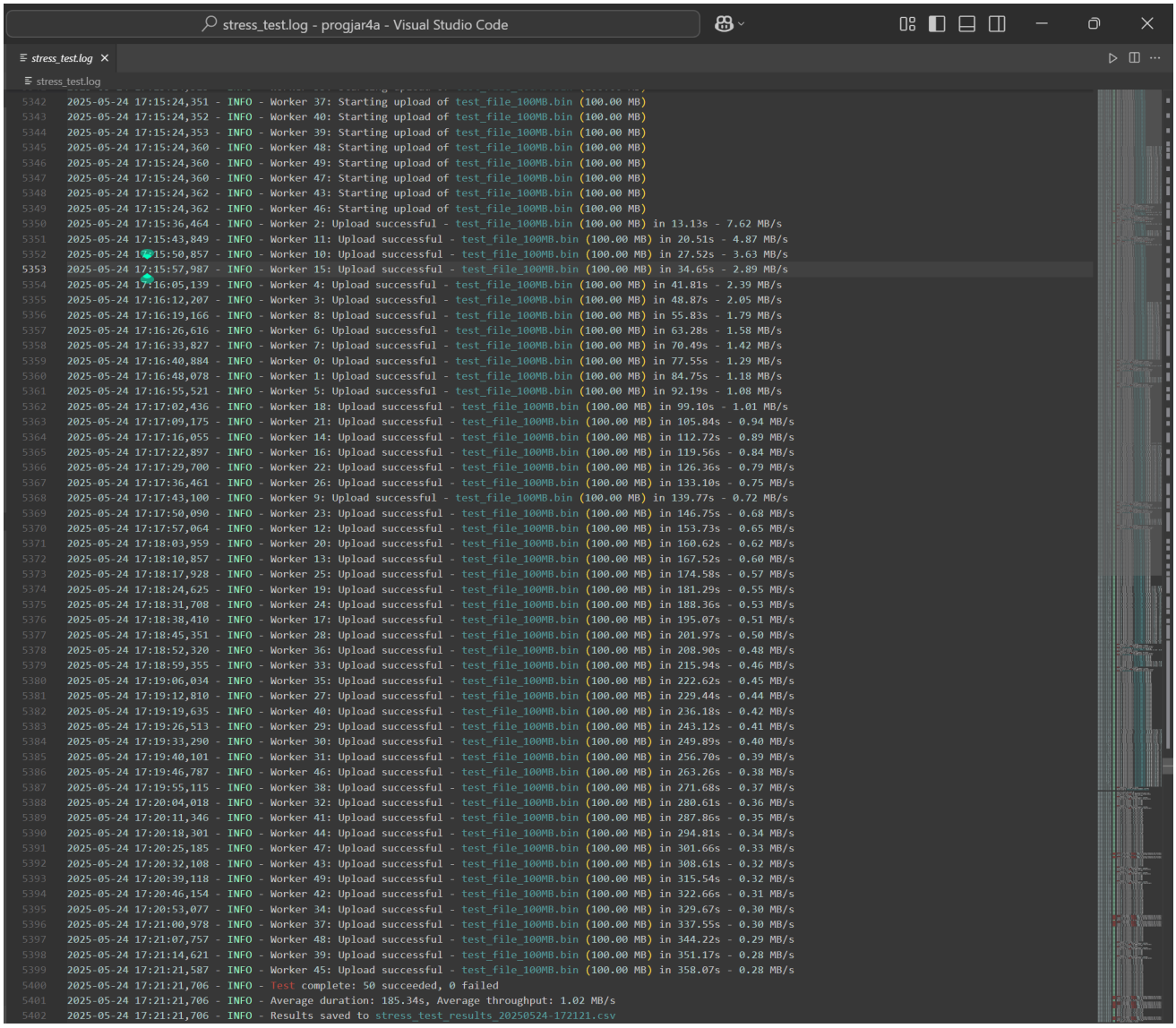
  Penurunan throughput saat concurrency tinggi (50 workers) membuktikan Little's Law. Pada multithread, ini akibat thread blocking; pada multiprocess, disebabkan IPC overhead atau memory exhaustion.

- **Context Switching Overhead**

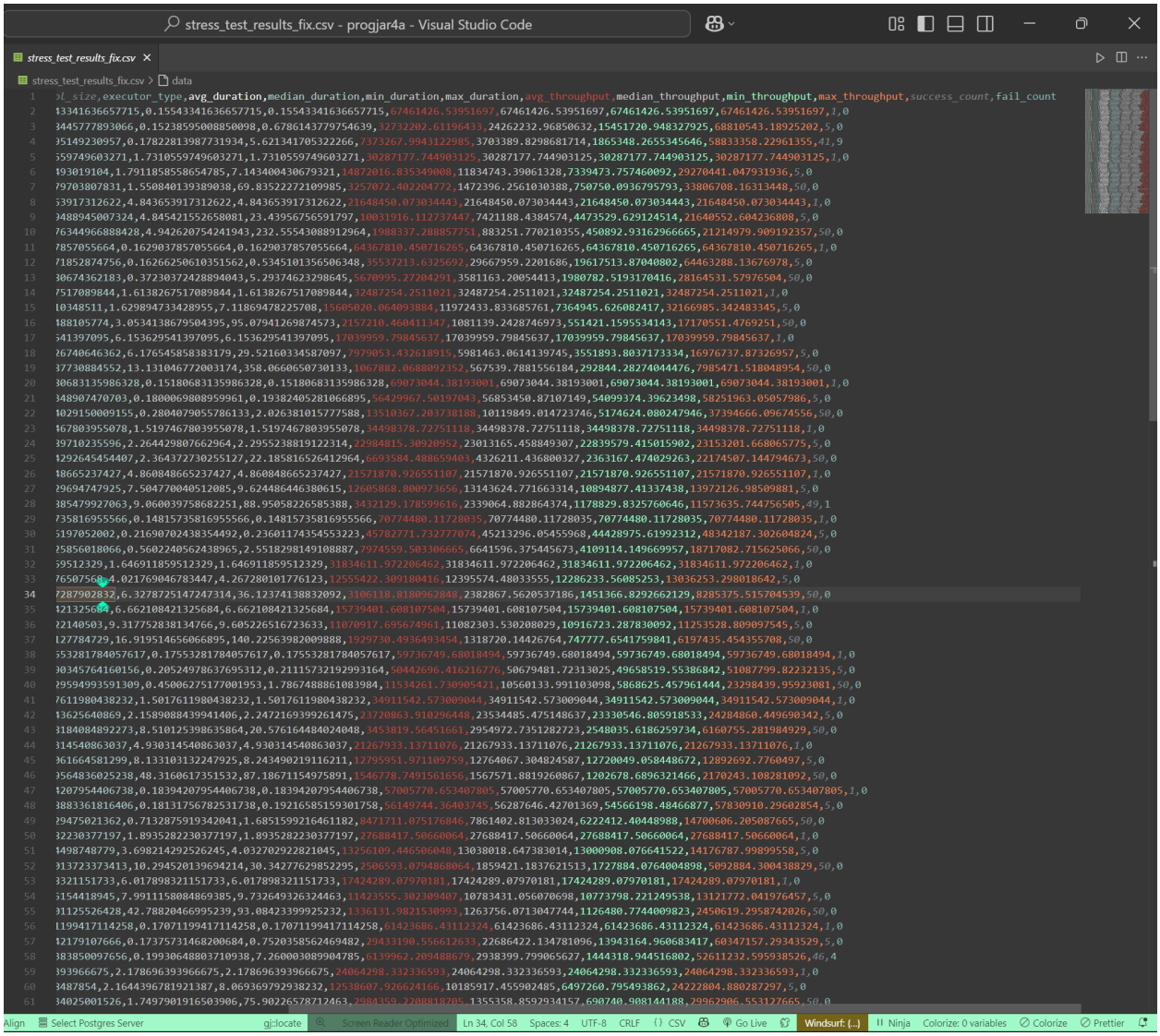
  Overhead terlihat jelas pada 50 workers, multithread pada context switching thread menyebabkan CPU thrashing; multiprocess pada process switching dan duplikasi memory meningkatkan latency.Terlihat jelas pada konfigurasi dengan 50 workers.

NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005

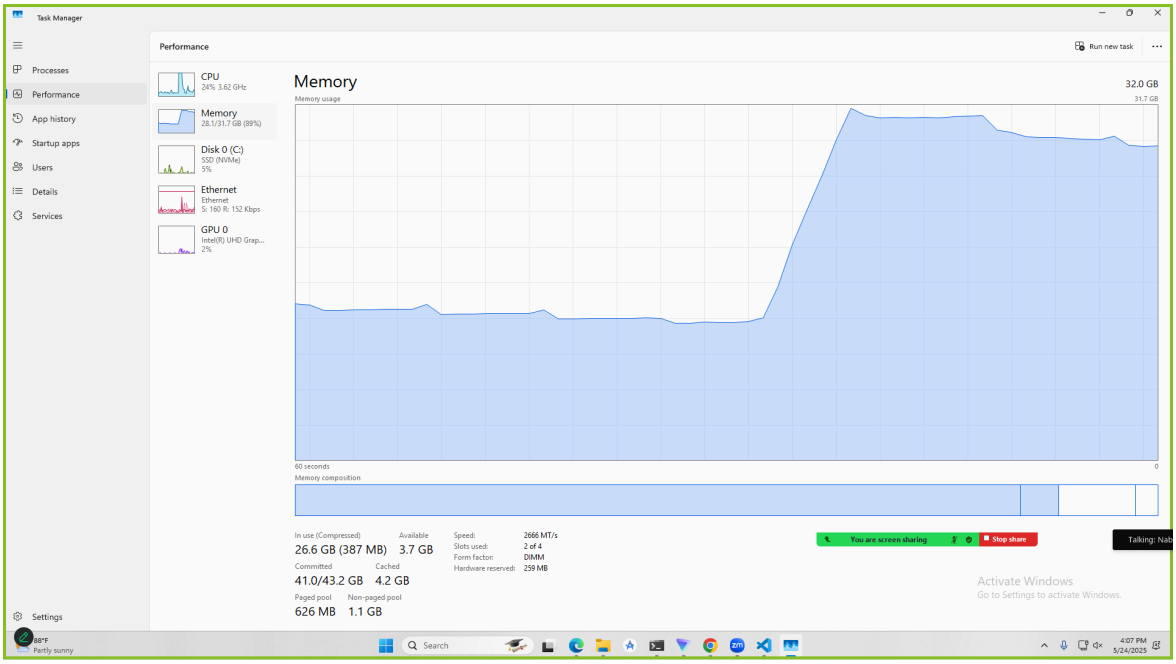
LAMPIRAN PENGERJAAN

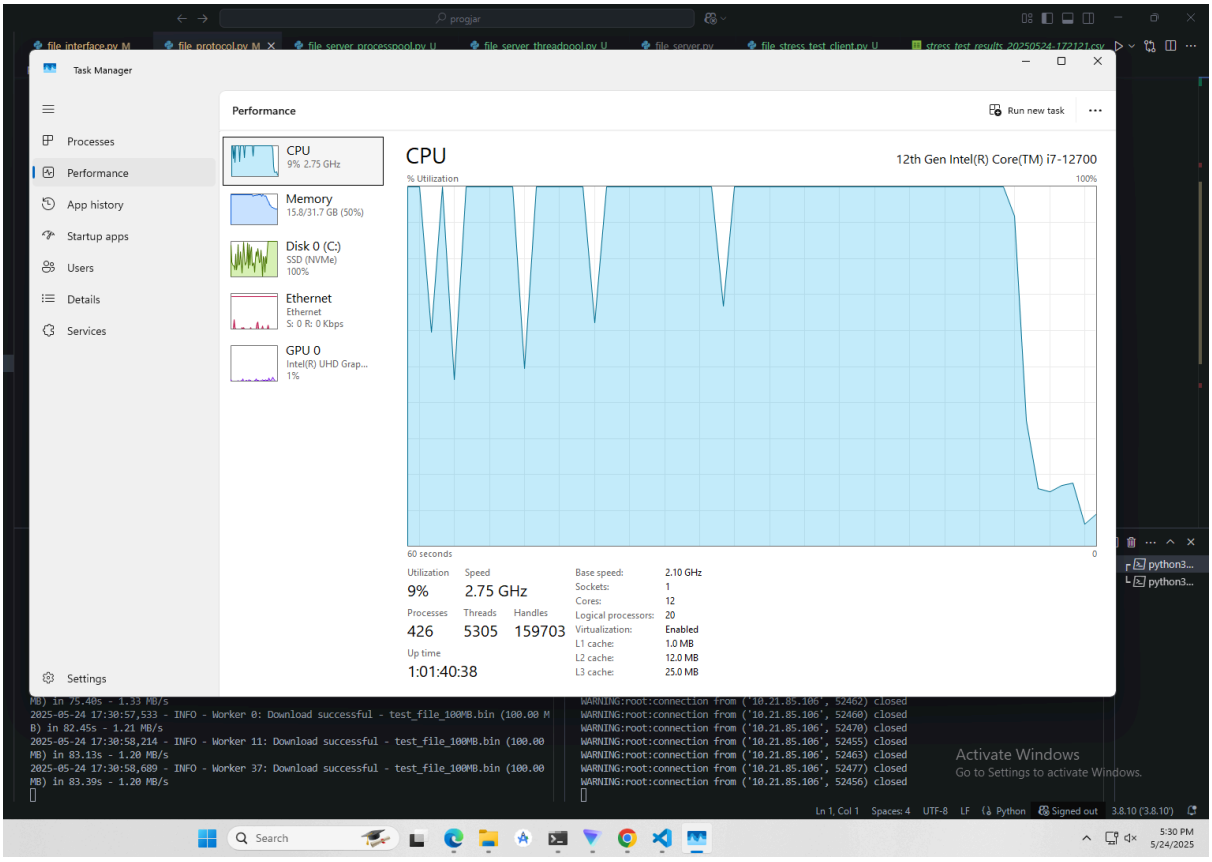


NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005



NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005





NABILAH ATIKA RAHMA
5025221005 - PROGJAR C
https://github.com/ranabel/network-programming-5025221005