Pemrograman Jaringan A

5025201195

Naufal Adli Purnama

https://github.com/napurnama/Progjar-23/tree/master/Tugas%203

Tugas 3

1. Buatlah program client untuk Tugas 2 dengan mengimplementasikan ketentuan di tugas 2 dengan metode
    a. Thread
    Dibuatkan kelas Client yang mewariskan kelas Thread. Subroutine koneksi dan pengiriman data dimasukkan dalam fungsi run() yang akan dijalankan ketika thread dijalankan.

```
class Client(Thread):
    def __init__(self, ip, port):
        self.addr_fam = socket.AF_INET
        self.stream = socket.SOCK_STREAM
        self.sock = socket.socket(self.addr_fam, self.stream)
        self.ip = ip
        self.port = port
        Thread.__init__(self)

    def run(self):
        logging.info("Client running on thread {}", self.ident)
        try:
            server_address = (self.ip, self.port)
            logging.info(f"connecting to {server_address}")
            self.sock.connect(server_address)
            message = 'TIME\r\n'
            logging.info(f"sending {message}")
            self.sock.sendall(message.encode())
            data = self.sock.recv(16)
            if data:
                data = data.decode()
                if data[-2:] == '\r\n' and len(data) == 14:
                    logging.info(f"{data}")
            else:
                logging.info(f"ERROR: Invalid Server Response
Format")
        except Exception as ee:
            logging.info(f"ERROR: {str(ee)}")
            exit(0)
        finally:
            logging.info("closing")
            self.sock.close()
```

b. Threadpool

Dibuatkan fungsi run() yang menerima argument berupa objek client. Fungsi ini yang akan dijalankan dalam pool

```
def run(client):
    try:
        server_address = (client.ip, client.port)
        logging.info(f"connecting to {server_address}")
        client.sock.connect(server_address)
        message = 'TIME\r\n'
        logging.info(f"sending {message}")
        client.sock.sendall(message.encode())
        data = client.sock.recv(16)
        if data:
            data = data.decode()
            if data[-2:] == '\r\n' and len(data) == 14:
                logging.info(f"{data}")
        else:
            logging.info(f"ERROR: Invalid Server Response
Format")
    except Exception as ee:
        logging.info(f"ERROR: {str(ee)}")
        exit(0)
    finally:
        logging.info("closing")
        client.sock.close()

if __name__ == "__main__":
    pool = ThreadPool()
    for i in range(100):
        clt = Client("localhost", 45000)
        pool.apply_async(run, args=(clt,))
```

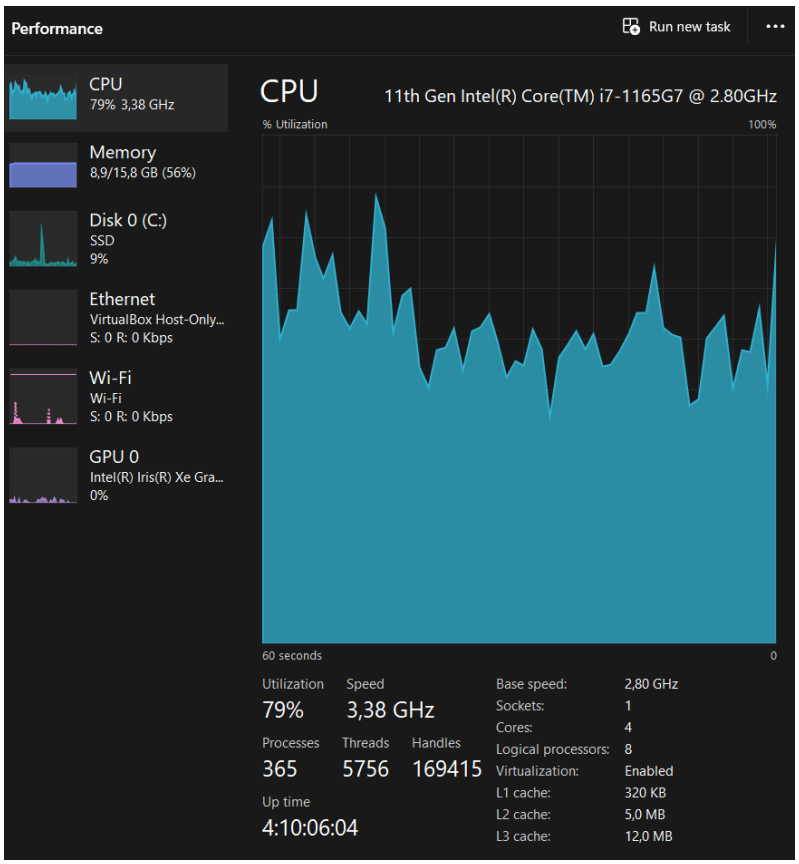
c. Process

Kode sama seperti variasi thread, hanya saja kelas Client mewariskan dari kelas Process

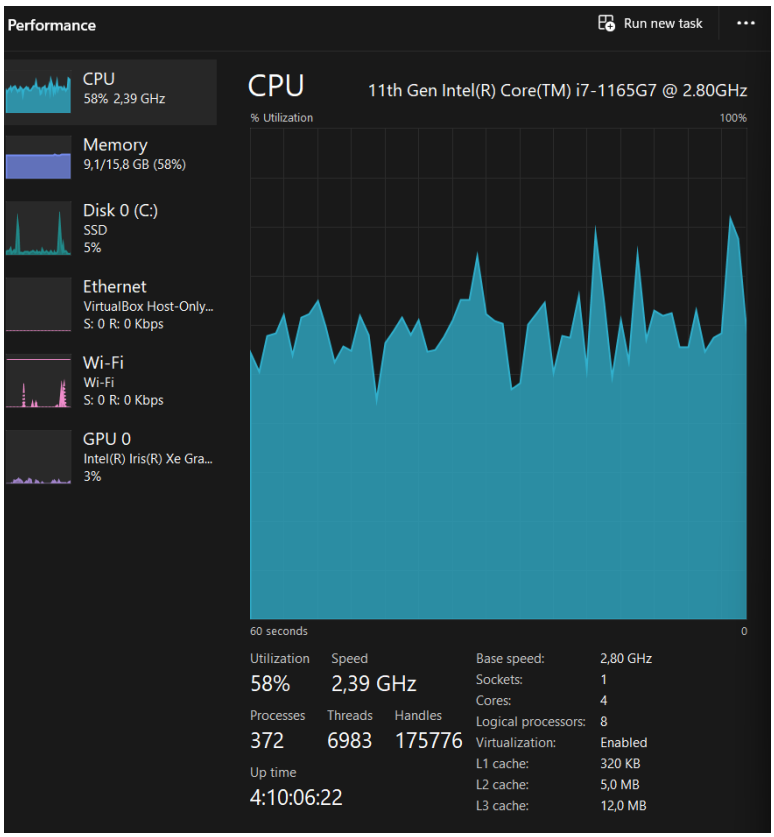
```
class Client(Process):
    def __init__(self, ip, port):
        self.addr_fam = socket.AF_INET
        self.stream = socket.SOCK_STREAM
        self.sock = socket.socket(self.addr_fam, self.stream)
        self.ip = ip
        self.port = port
        Process.__init__(self)

    def run(self):
        logging.info("Client running on thread {}", self.ident)
        try:
            server_address = (self.ip, self.port)
```

```
            logging.info(f"connecting to {server_address}")
            self.sock.connect(server_address)
            message = 'TIME\r\n'
            logging.info(f"sending {message}")
            self.sock.sendall(message.encode())
            data = self.sock.recv(16)
            if data:
                data = data.decode()
                if data[-2:] == '\r\n' and len(data) == 14:
                    logging.info(f"{data}")
            else:
                logging.info(f"ERROR: Invalid Server Response
Format")
        except Exception as ee:
            logging.info(f"ERROR: {str(ee)}")
            exit(0)
        finally:
            logging.info("closing")
            self.sock.close()
```

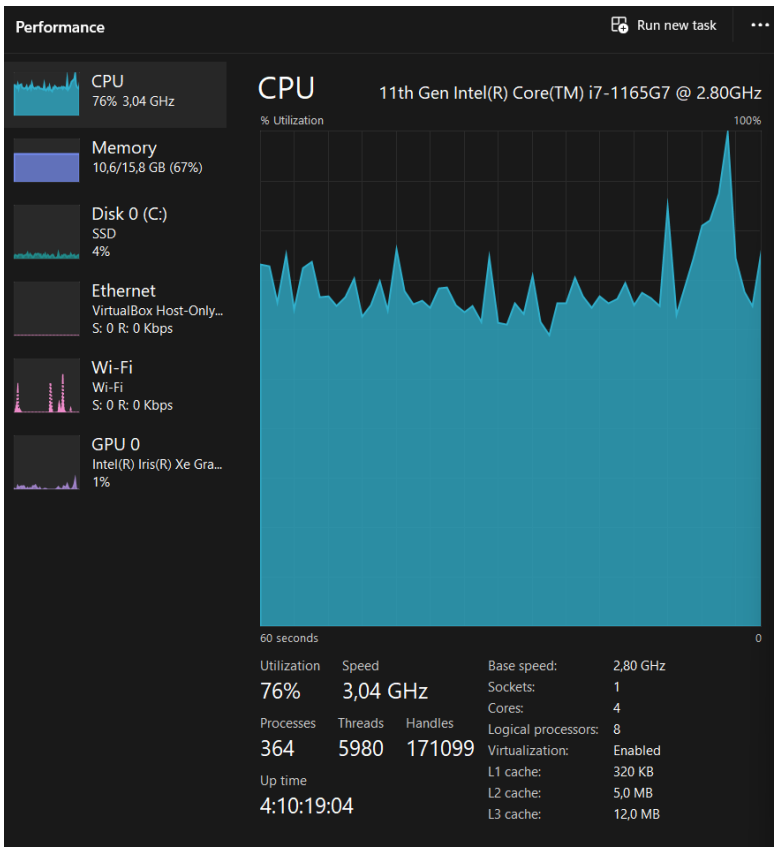
2. Jalankan server pada tugas 2, lalu, untuk setiap metode pada nomor 1, jalankan tugas 3 nomor 1 dan amatilah jumlah maksimum thread yang dapat dijalankan di lingkungan eksekusi anda.
    1. Thread
        1. Sebelum



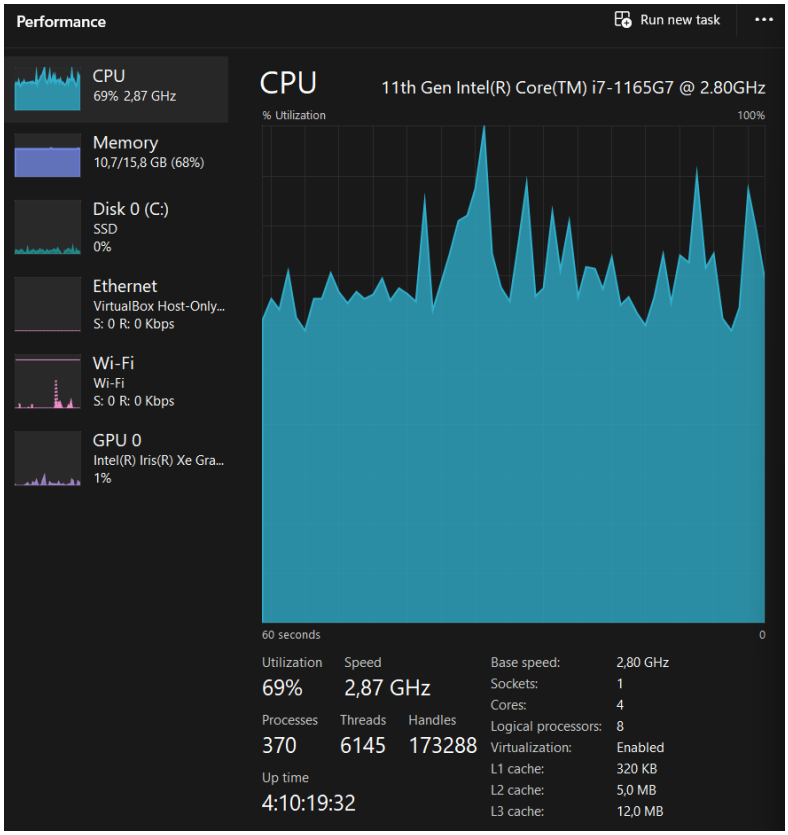
2. Sesudah


Performance
Run new task
CPU
58% 2,39 GHz
Memory
9,1/15,8 GB (58%)
Disk 0 (C:)
SSD
5%
Ethernet
VirtualBox Host-Only...
S: 0 R: 0 Kbps
Wi-Fi
Wi-Fi
S: 0 R: 0 Kbps
GPU 0
Intel(R) Iris(R) Xe Gra...
3%
CPU
11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
% Utilization
100%
60 seconds
0
Utilization
58%
Speed
2,39 GHz
Processes
372
Threads
6983
Handles
175776
Up time
4:10:06:22
Base speed: 2,80 GHz
Sockets: 1
Cores: 4
Logical processors: 8
Virtualization: Enabled
L1 cache: 320 KB
L2 cache: 5,0 MB
L3 cache: 12,0 MB

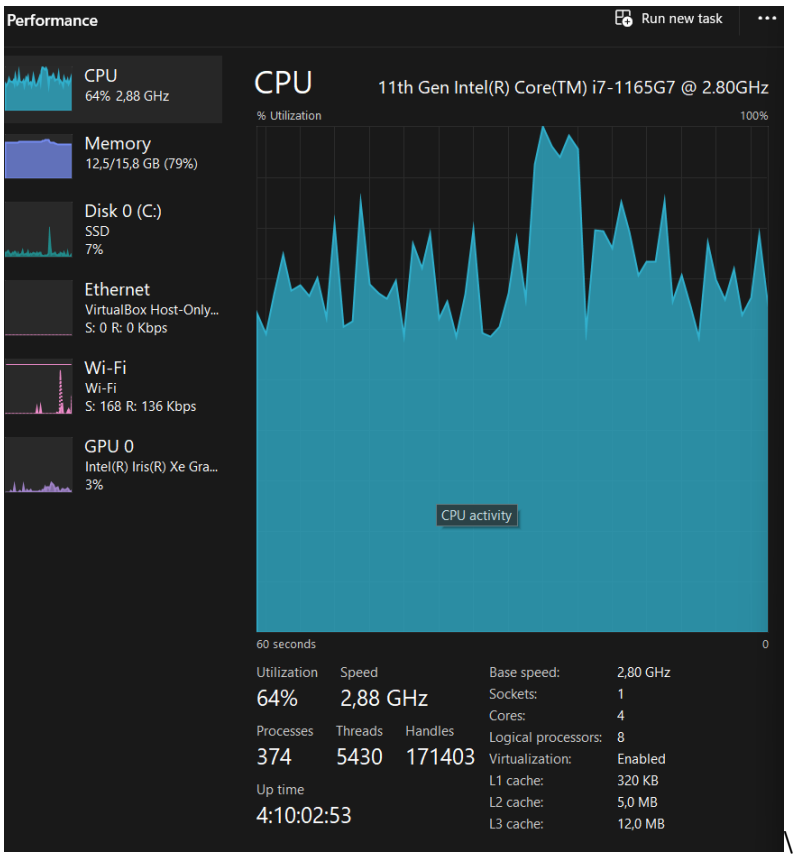
2. Thread Pool
    1. Sebelum



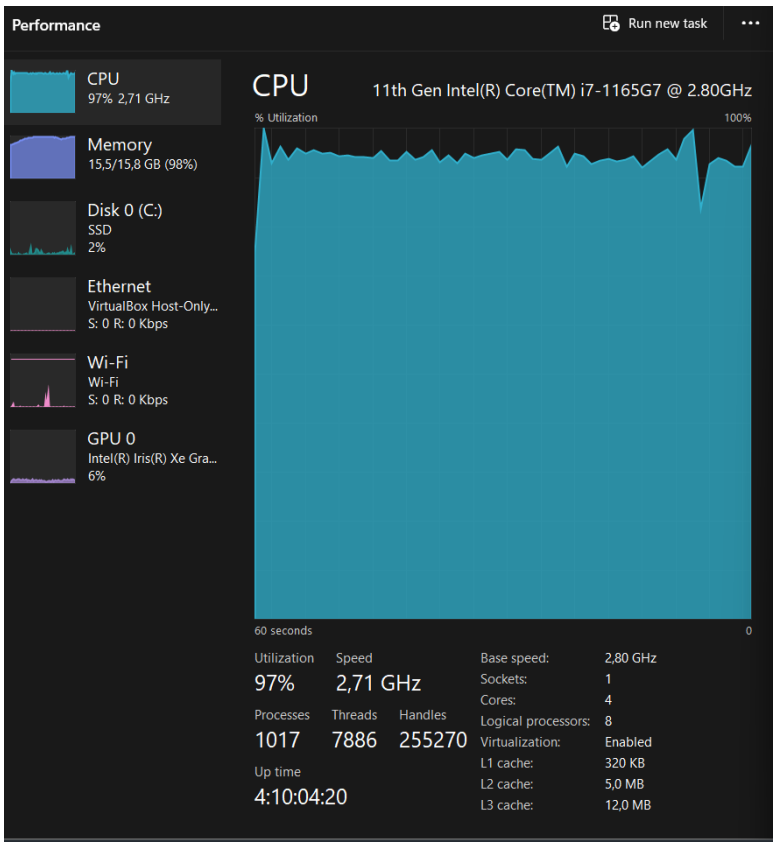
    2. Sesudah



3. Process
    1. Sebelum



    2. Sesudah



3. Jalankan di lab environment

a. Jalankan program client (tugas 3 nomor 2) dan server (pada tugas 2), capturelah interaksi antara client dan server. Berikan deskripsi tentang interaksi ini terutama dengan jumlah thread di masing-masing sisi (client / server)

1. Thread

   Seperti bayangan, menjalankan client pada thread akan berdampak minim pada jumlah proses yang dijalankan mesin. Dengan pengujian 1000 request client, terekam kenaikan jumlah thread kurang lebih 1000 buah thread.

2. Thread Pool

   Terlihat dari analytics sistem bahwa thread tidak berubah drastic meskipun pengujian disetel untuk mencoba 1000 request client. Apabila dilihat, server hanya menerima delapan buah request client.

```
C:\Users\User\Documents\CompSci\UNI\SM-6\Progjar
\progjar\Tugas 3>python client_pool.py
INFO:root:connecting to ('localhost', 45000)
INFO:root:connecting to ('localhost', 45000)
INFO:root:connecting to ('localhost', 45000)
INFO:root:connecting to ('localhost', 45000)
INFO:root:connecting to ('localhost', 45000)
INFO:root:connecting to ('localhost', 45000)
INFO:root:connecting to ('localhost', 45000)
INFO:root:connecting to ('localhost', 45000)
INFO:root:sending TIME

INFO:root:sending TIME

INFO:root:sending TIME

INFO:root:sending TIME

INFO:root:sending TIME

INFO:root:sending TIME

INFO:root:sending TIME

INFO:root:sending TIME
```

   Hal ini awalnya terlihat aneh. Namun apabila dilihat dari mesin penguji, angka ini sama seperti jumlah logical processor. Dapat ditarik kesimpulan bahwa perilaku default dari ThreadPool adalah untuk membatas jumlah proses sesuai dengan jumlah processor.

3. Process

   Pada skenario ini, menjalankan client pada proses tersendiri memakan sangat banyak sumber daya. Mesin harus membuat proses baru serta thread baru untuk menjalankan satu request client.