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
   1. 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() |

* 1. 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,)) |

* 1. 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() |

1. 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

Chart

Description automatically generated

* + 1. Sesudah

Chart

Description automatically generated with medium confidence

* 1. Thread Pool
     1. Sebelum

Chart

Description automatically generated

* + 1. Sesudah

Chart

Description automatically generated

* 1. Process
     1. Sebelum

A screenshot of a computer

Description automatically generated with medium confidence\

* + 1. Sesudah

Graphical user interface

Description automatically generated

1. Jalankan di lab environment
2. 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.

* + 1. 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.

Graphical user interface, text

Description automatically generated

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.

* + 1. 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.