Nama : Alisa Jihan Azizah

NPM : 21083010064

Mata Kuliah : Sistem Operasi – B

Laporan Materi-8 Multiprocessing

Python memiliki modul multiprocessing yang artinya dapat menjalankan sebuah fungsi di subproses baru. Overhead untuk memulai proses baru cukup besar, jadi bisa juga memakai process Pool supaya lebih optimal. Di titik Ketika membuat subprocess, isi memori saat ini di proses utama akan di clone ke memori yang sifatnya copy-on-write (COW). Artinya selama memori tersebut tidak diubah, maka tidak akan dibuat salinannya, hanya akan dishare.

Manfaat dari multiprocessing sendiri adalah sebagai berikut:

- Menggunakan CPU untuk komputasi
- Tidak berbagi sumber daya memori
- Memerlukan sumber daya memori dan waktu yang tidak sedikit
- Tidak memerlukan sinkronisasi memori

Untuk procobaannya dapat dilakukan dari tugas dibawah ini.

Dengan menggunakan pemrosesan paralel buatlah program yang dapat menentukan sebuah bilangan itu ganjil atau genap!

1. Hal pertama yang dilakukan adalah membuat file baru pada terminal linux dengan perintah nano ‘namafile’ (contoh : nano tugas_8.py).
2. Setelah masuk pada file nano, kemudian import sebuah library bawaan dari python untuk memudahkan dalam mengeksekusi.
   ```
   from os import getpid
   from time import time,sleep
   from multiprocessing import cpu_count, Pool, Process
   ```
   Keterangan :
   - getpid digunakan untuk mengambil ID proses
   - time digunakan untuk mengambil waktu(detik)
   - sleep digunakan untuk memberi jeda waktu(detik)
   - cpu_count digunakan untuk melihat jumlah CPU
   - Pool adalah sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada computer
   - Process adalah sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada computer
3. Lakukan instalisasi fungsi yang akan digunakan dengan mendeklarasikan dengan sebuah nama yaitu cetak. Sebelum itu, saya membuat inputan user yang akan memiliki hubungan deklarasi lanjutannya.

```
print ("Input Bilangan : ")
x = int(input())

def cetak(i):
        for i in range(x):
                if i % 2 == 0:
                        print(f"angka ganjil-{i+1}"," - ID prosesnya ", getpid())
                else:
                        print(f"angka genap-{i+1}"," - ID prosesnya ", getpid())
        sleep(1)
```

Pada def cetak, saya membuat loopingan for yang dibatasi oleh inputan yang akan diberikan oleh user, saya juga menggunakan if-else, yang akan menunjukan perulangan dari inputan user itu merupakan bilangan ganjil atau genap.

4. Selanjutnya masuk kedalam multiprocessing yang pertama, yaitu multiprocessing sekuensial yang menggunakan looping for dengan rangenya 1 yang artinya setiap perulangan hanya terjadi satu kali. Selain itu juga memberikan sekuensial_awal dan sekuensial_akhir untuk menghitung waktu eksekusi yang akan dibuat setelahnya.

```
#Multiprocessing Sekuensial
print("Multiprocessing Sekuensial")
sekuensial_awal = time()

for i in range(1):
        cetak(i)
sekuensial_akhir = time()
print(" ")
```

5. Yang kedua yaitu multiprocessing process. Tidak jauh berbeda dari multiprocessing sekuensial, disini juga menggunakan looping for untuk perulangannya. Perbedaannya adalah kita menyimpan sebuah prosesnya agar tidak merambat ke proses selanjutnya.

```
#Multiprocessing Process
print("Multiprocessing Process")
kumpulan_proses = []
process_awal = time()

for i in range(1):
        p = Process(target=cetak, args=(i,))
        kumpulan_proses.append(p)
        p.start()
for i in kumpulan_proses:
        p.join()
process_akhir = time()
print(" ")
```

6. Yang ketiga yatiu multiprocessing pool. Pada multiprocessing ini, kita menggunakan fungsi .map() untuk memetakan panggilan fungsi cetak kedalam sebanyak 1 kali.

```
#Multiprocessing Pool
print("Multiprocessing Pool")
pool_awal = time()

pool = Pool()
pool.map(cetak, range(0,1))
pool.close()

pool_akhir = time()
print(" ")
```

7. Langkah terakhir adalah membuat banding waktu pengeksekusian berlangsung, dengan cara mengurangi waktu_akhir dengan waktu_awal, seperti yang telah dideklarasikan pada setiap awal dan akhir sebuah program multiprocessing.

```
#Banding Waktu Eksekusi
print("Sekuensial : ", sekuensial_akhir - sekuensial_awal, "detik")
print("Multiprocessing.Process : ", process_akhir - process_awal, "detik")
print("Multiprocessing.Pool : ", pool_akhir - pool_awal, "detik")
```

8. Hasil outputnya adalah sebagai berikut.

```
alisa@alisa-VirtualBox:~$ python3 tugas_8.py
Input Bilangan :
3
Multiprocessing Sekuensial
angka ganjil-1  - ID prosesnya  2727
angka genap-2  - ID prosesnya  2727
angka ganjil-3  - ID prosesnya  2727

Multiprocessing Process
angka ganjil-1  - ID prosesnya  2728
angka genap-2  - ID prosesnya  2728
angka ganjil-3  - ID prosesnya  2728

Multiprocessing Pool
angka ganjil-1  - ID prosesnya  2729
angka genap-2  - ID prosesnya  2729
angka ganjil-3  - ID prosesnya  2729

Sekuensial :  1.0975184440612793 detik
Multiprocessing.Process :  1.0346465110778809 detik
Multiprocessing.Pool :  1.0961408615112305 detik
```