Nama : Ilil Musyarof Asfiani

Npm : 21083010073

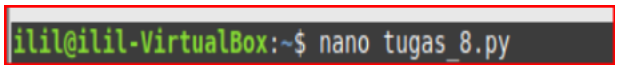
Multiprocessing

Objektifitas pada modul 7 adalah melakukan pemrograman paralel(yang merupakan salah satu konsep dasar system operasi) dengan Multiprocessing. Pemrograman paralel adalah sebuah teknik eksekusi perintah yang mana dilakukan secara bersamaan pada CPU. Seluruh bahasa pemrograman yang populer dapat melakukan pemrograman paralel dengan modul bawaan atau memang pengaturan defaultnya seperti itu. Praktikan akan melakukan penerapan pemrograman paralel sederhana dengan Python. Kami memilih Python karena bahasa pemrograman tersebut sudah otomatis terinstal di hampir seluruh sistem operasi berbasis Linux selain itu secara default komputasi di Python dilakukan secara sekuensial. Biarpun sekuensial kita dapat menjadikannya paralel dengan fungsi bawaan yang telah disediakan oleh Python

Manfaat Multipreprocessing

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

Input



```
 GNU nano 6.2                                    tugas_8.py *
from os import getpid
from time import time,sleep
from multiprocessing import cpu_count, Pool, Process

def cetak(i):
        if (i+1)%2==0:
                print(i+1, "genap - ID Process", getpid())
        else:
                print(i+1, "ganjil - ID Process", getpid())
        sleep(1)

n=int(input("Angka batasan? "))

#SEKUENSIAL
sekuensial_awal = time()
print("Sekuensial")
for i in range(n):
        cetak(i)
sekuensial_akhir=time()

#MULTIPROCESSING DENGAN KELAS PROCESS
process_awal=time()
print("Multiprocess.process")
for i in range(n):
        p=Process(target=cetak, args=(i, ))
```

```
        p.start()
        p.join()
process_akhir=time()

#MULTIPROCESSING DENGAN KELAS POOL
pool_awal=time()
pool = Pool()
print("Multiprocess.pool")
pool.map(cetak,range(0,n))
pool.close()
pool_akhir=time()

#BANDINGKAN WAKTU EKSEKUSI
print("Perbandingan waktu")
print("Sekuensial:", sekuensial_akhir - sekuensial_awal, "detik")
print("Kelas Process:", process_akhir - process_awal, "detik")
print("Kelas Pool:", pool_akhir - pool_awal, "detik")
```

Pertama Import modul yang diperlukan.

- getpid digunakan untuk mendapatkan ID proses
- time digunakan untuk mengambil waktu(dektik) pada proses dijalankan atau diakhiri
- sleep digunakan untuk memberi jeda waktu(detik)
- cpu_count digunakan untuk menghitung jumlah cpu yang tersedia
- Pool adalah sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan parallel dengan menggunakan proses sebanyak jumlah CPU pada komputer
- Process adalah sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan parallel dengan menggunakan proses secara beruntun pada computer.

Fungsi bernama ‘cetak’ digunakan untuk mencetak angka dari variabel i beserta ID proses sejumlah parameter apakah angka yang masuk ganjil atau genap. Disini ketika angka yang dimasukkan menghasilkan 0 ketika di modulo 2, maka angka tersebut genap dan jika menghasilkan angka lainnya, maka angka tersebut berarti ganjil. Kita panggil fungsi sleep untuk memberi jeda waktu(detik) sebanyak parameter yang diberikan. Kemudian fungsi ‘n’ digunakan untuk User diminta menginputkan sebuah angka bulat yang digunakan sebagai Nilai Batasan. Lalu print ("\n") untuk memberi spasi agar mudah dipahami.

Proses pertama gunakan sekuensial processing.

• sekuansial_awal adalah variabel untuk mendapatkan waktu durasi sebelum proses sekuensial processing berlangsung.

• sekuensial_akhir adalah variabel untuk mendapatkan waktu durasi setelah proses sekuensial processing berlangsung.

• Lakukan Looping sebanyaknya angka yang dimasukkan oleh user, dan gunakan fungsi ‘cetak’ yang sudah kita isi di awal untuk mencetak setiap angka ganjil atau genap dengan id prosesnya masing – masing.

• Lalu print ("\n") untuk memberi spasi agar mudah dipahami.

Proses kedua gunakan multiprocessing dengan kelas process

• process_awal adalah variabel untuk mendapatkan waktu awal mulainya proses dijalankan

• process_akhir adalah variabel untuk mendapatkan waktu berakhirnya proses dijalankan

• Dapat diperhatikan dengan seksama bahwa ID proses tiap memanggil fungsi cetak adalah berbeda-beda. Ini menandakan bahwa tiap pemanggilan fungsi cetak ditangani

oleh satu proses saja. Kemudian untuk pemanggilan selanjut'nya ditangani oleh proses yang lain.

• Lakukan Looping sebanyaknya angka yang dimasukkan oleh user, dan gunakan fungsi cetak yang sudah kita isi di awal untuk mencetak setiap angka ganjil atau genap dengan id proses masing – masing

• p.start() digunakan untuk mengeksekusi fungsi cetak di kelas process

• p.join() digunakan agar proses ditunggu hingga proses sebelumnya selesai. Sehingga akan menghasilkan id proses yang berbeda – beda tiap prosesnya.

• Lalu print ("\n") untuk memberi spasi agar mudah dipahami.

Proses Ketiga gunakan multiprocessing dengan kelas pool

• pool_awal adalah variabel untuk mendapatkan waktu awal mulainya proses dijalankan

• pool_akhir adalah variabel untuk mendapatkan waktu berakhirnya proses proses dijalankan

• fungsi map() digunakan untuk memetakan pemanggilan fungsi cetak ke dalam setiap CPU yang tersedia sebanyak 0-n kali yang mana ‘n’ adalah inputan batasan dari user.

• Lalu print ("\n") untuk memberi spasi agar mudah dipahami.

Proses terakhir yaitu bandingkan setiap jenis eksekusi dengan waktu akhir – waktu awal untuk melihat berapa lama pemrosesan berlangsung. Kemudian print waktu eksekusi sekuensial, kelas process, dan kelas pool untuk melihat hasil waktu berapa detiknya. Lalu print ("\n") untuk memberi spasi agar mudah dipahami.

Output