Nama : Divayanti Febri Sakina

NPM : 21083010099

Kelas : Sistem Operasi B

Multiprocessing

Objektifitas pada modul 7 adalah melakukan pemrograman paralel(yang merupakan salah satu konsep dasar sistem 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 default' nya 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 menjadikan nya 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

Latihan Soal :

```
 GNU nano 2.9.3                                    Tugas_8.py

# Muat built-in libraries yang akan digunakan :
from os import getpid
from time import time, sleep
from multiprocessing import Pool, Process

# Inisialisasi function yang akan  digunakan :
def cetak(a):
    angka = a % 2
    if angka == 0:
        print(a, "Genap - ID proses", getpid())
    else:
        print(a, "Ganjil - ID proses", getpid())
    sleep(1)

# Menginput bilangan
y = int(input("Input bilangan: "))

# Sekuensial
print("\nPemprosesan Sukensial")
sekuensial_awal = time()

for a in range(1, y + 1):
    cetak(a)

sekuensial_akhir = time()

# Kelas Proses
print("\nMultiprocessing dengan multiprocessing.Process")
```

```
 GNU nano 2.9.3                                    Tugas_8.py


kumpulan_proses = []
proses_awal = time()

for a in range(1, y + 1):
    p = Process(target=cetak, args=(a,))
    kumpulan_proses.append(p)
    p.start()

for a in kumpulan_proses:
    p.join()

proses_akhir = time()

# Pool
print("\nMultiprocessing dengan multiprocessing.Pool")
pool_awal = time()

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

pool_akhir = time()

# Perbandingan waktu eksekusi
print("\nWaktu eksekusi sekuensial           :", sekuensial_akhir - sekuensial_awal, "detik")
print("Waktu eksekusi multiprocessing.Process :", proses_akhir - proses_awal, "detik")
print("Waktu eksekusi multiprocessing.Process :", pool_akhir - pool_awal, "detik")
```

1. Membuat file python di terminal linux dengan nano Tugas_8.py
2. Setelah masuk ke dalam nano melakukan beberapa built-in libraries yang akan digunakan dalam program, yaitu

```
# Muat built-in libraries yang akan digunakan :
from os import getpid
from time import time, sleep
from multiprocessing import Pool, Process
```

   - Getpid
     Digunakan untuk mengambil ID proses yang sedang running
   - Time
     Digunakan untuk mengambil waktu(detik)
   - Sleep
     Digunakan untuk memberi jeda waktu(detik) yang diberikan
   - Process
     Merupakan sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer
   - Pool
     Merupakan sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada komputer

3. Selanjutnya kita menginisialisasikan furngsi yang akan kita gunakan.

```
# Inisialisasi function yang akan  digunakan :
def cetak(a):
    angka = a % 2
    if angka == 0:
        print(a, "Genap - ID proses", getpid())
    else:
        print(a, "Ganjil - ID proses", getpid())
    sleep(1)
```

   - Pertama kita membuat parameter **(cetak(a)).**
   - Selanjutnya melakukan perhitungan mod terhadap nilai dalam variabel **a** untuk mengetahui apakah nilai dari variabel **a** itu genap atau ganjil.
   - Nanti hasil perhitungan di simpan di variabel **angka.**
   - Kita melakukan fungdi perulangan untuk pengecekan nilai dari variabel **angka**. Jika saat kita operasikan mod hasilnya 0 maka nilai dari variabel **a** adalah genap, namun jika hasil selain 0 makan nilai variabel **a** adalah ganjil.
   - Fungsi **sleep(1)** digunakan untuk memberi jeda 1 detiksebelum lanjut eksekusi.

4. Membuat inputan bilangan

```
# Menginput bilangan
y = int(input("Input bilangan: "))
```

   - Membuat tempat untuk menginputkan bilangan dengan tipe integer yang akan disimpan di variabel **y**

5. Sekuensial

```
# Sekuensial
print("\nPemprosesan Sukensial")
sekuensial_awal = time()

for a in range(1, y + 1):
    cetak(a)

sekuensial_akhir = time()
```

- Menggunakan fungsi **time()** digunakan untuk mengambil waktu yang akan disimpan dalam variabel **sekuensial_awal**
- Melakukan loopig for variabel **a** dengan batasan **(1, y+1)** yang menggunakan fungsi **cetak(a)**
- Menggunakan fungsi **time()** digunakan untuk mengambil waktu yang akan disimpan dalam variabel **sekuensial_akhir**

6. Kelas Proses

```
# Kelas Proses
print("\nMultiprocessing dengan multiprocessing.Process")
```

```
kumpulan_proses = []
proses_awal = time()

for a in range(1, y + 1):
    p = Process(target=cetak, args=(a,))
    kumpulan_proses.append(p)
    p.start()

for a in kumpulan_proses:
    p.join()

proses_akhir = time()
```

- Menggunakan fungsi **time()** digunakan untuk mengambil waktu yang akan disimpan dalam variabel **proses_awal**
- Diinisialisasika list kososng [] dengan nama variabel kumpulan_proses
- Melakukan perulangan for loop untuk **a** di dalam **batasan (1, y+1)** yang menggunakan kelas **target=cetak** dan **args=(a,)** disimpan dalam variabel p. Lalu setiap nilai variabel p masuk ke dalam list **kumpulan_proses**. Selanjutnya Digunakan method join() terhadap variabel **p (p.join())** untuk menggabungkan kumpulan proses yang telah ditampung agar tidak tercampur ke proses selanjutnya.
- Menggunakan fungsi **time()** digunakan untuk mengambil waktu yang akan disimpan dalam variabel **proses_akhir**

7. Kelas Pool

```
# Pool
print("\nMultiprocessing dengan multiprocessing.Pool")
pool_awal = time()

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

pool_akhir = time()
```

- Menggunakan fungsi **time()** digunakan untuk mengambil waktu yang akan disimpan dalam variabel **pool_awal**
- Diinisialisasikan kelas Pool() yang disimpan dalam variabel **pool**. Menggunakan method **map()** terhadap variabel pool **(pool.map())** untuk memetakan pemanggilan fungsi cetak ke dalam 4 CPU sebanyak yang bilangan yang berada pada batasan **(1, y +1).**
- Digunakan metode close() terhadap variabel pool **(pool.close()**) untuk mencegah tugas lain dikirim ke dalam pool. Setelah semua tugas selesai, maka proses akan keluar.

8. Perbandingan waktu eksekusi

```
# Perbandingan waktu eksekusi
print("\nWaktu eksekusi sekuensial            :", sekuensial_akhir - sekuensial_awal, "detik")
print("Waktu eksekusi multiprocessing.Process :", proses_akhir - proses_awal, "detik")
print("Waktu eksekusi multiprocessing.Process :", pool_akhir - pool_awal, "detik")
```

- Dilakukan cetak nilai durasi waktu untuk setiap pemrosesan yang didapatkan melalui perhitungan antara waktu awal setiap proses dikurangi dengan waktu akhir setiap proses seperti yang tertera pada gambar di atas.

9. Kita coba jalankan script dengan python3 Tugas_8.py dengan mencoba menginputkan bilangan 7

```
divayanti@divayanti-VirtualBox: ~/Perte
File  Edit  View  Search  Terminal  Help
divayanti@divayanti-VirtualBox:~/Pertemuan10$ nano Tugas_8.py
divayanti@divayanti-VirtualBox:~/Pertemuan10$ python3 Tugas_8.py
Input bilangan: 7

Pemprosesan Sukensial
1 Ganjil - ID proses 2596
2 Genap - ID proses 2596
3 Ganjil - ID proses 2596
4 Genap - ID proses 2596
5 Ganjil - ID proses 2596
6 Genap - ID proses 2596
7 Ganjil - ID proses 2596

Multiprocessing dengan multiprocessing.Process
1 Ganjil - ID proses 2597
3 Ganjil - ID proses 2599
2 Genap - ID proses 2598
4 Genap - ID proses 2600
5 Ganjil - ID proses 2601
7 Ganjil - ID proses 2603
6 Genap - ID proses 2602

Multiprocessing dengan multiprocessing.Pool
1 Ganjil - ID proses 2604
2 Genap - ID proses 2605
4 Genap - ID proses 2604
3 Ganjil - ID proses 2605
5 Ganjil - ID proses 2605
6 Genap - ID proses 2604
7 Ganjil - ID proses 2605
```

```
Waktu eksekusi sekuensial            : 7.016790390014648 detik
Waktu eksekusi multiprocessing.Process : 1.016885757446289 detik
Waktu eksekusi multiprocessing.Process : 4.0590174198150635 detik
divayanti@divayanti-VirtualBox:~/Pertemuan10$
```

Terlihat perbadingan waktu eksekusi dan yang paling lambat adalah waktu eksekusi sekuesial

10. Untuk mengetahui jumlah CPU yang terdapat pada komputer dapat digunakan function cpu_count() yang terdapat dalam library multiprocessing sebagai berikut:

```
divayanti@divayanti-VirtualBox:~/Pertemuan10$ python3
Python 3.6.9 (default, Jun 29 2022, 11:45:57)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from multiprocessing import cpu_count
>>> cpu_count()
2
>>>
```