

Pemrosesan Paralel pada Arsitektur x86-64 dengan menggunakan Bahasa Pemrograman Python

Sunu Ilham Pradika
Teknik Informatika, Fakultas Ilmu
Komputer
Universitas Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
sunudika@gmail.com

Alfath Daryl Alhajir
Teknik Informatika, Fakultas Ilmu
Komputer
Universitas Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
itsalfath123@gmail.com

Radical Rakhman Wahid
Teknik Informatika, Fakultas Ilmu
Komputer
Universitas Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
roploverz@gmail.com

Sejalan dengan berkembangnya ilmu pengetahuan dan teknologi di dunia, muncul permasalahan yang semakin kompleks dan membutuhkan waktu komputasi yang tidak singkat untuk memecahkan masalah tersebut. Pemrosesan paralel hadir untuk menjadi solusi. Konsepnya adalah dengan membagi suatu proses besar menjadi bagian - bagian kecil untuk bisa dikerjakan bersama oleh prosesor yang tersedia. Python hadir sebagai pelengkap dan memudahkan kita sebagai programmer untuk melakukan pemrosesan paralel dengan menggunakan library yang sudah tersedia. Hal ini semakin menarik karena programmer kini tidak perlu lagi untuk mengatur hal - hal kecil seperti *queue*, *proses* - *proses*, dan *data* yang ingin diolah karena adanya library tersebut.

Kata kunci—Pemrosesan paralel, Python, *Multiprocessing library*.

I. PENDAHULUAN

Dalam perkembangan suatu aplikasi atau program, merupakan praktek yang baik apabila kita dapat memaksimalkan *resource* seperti *core-core* komputer atau memori yang ada pada komputer secara efisien sehingga *resource* komputer yang ada tidak terbuang begitu saja. Praktik yang bisa dilakukan untuk yaitu menggunakan algoritma terbaik yang memakan *resource* sedikit mungkin dalam bentuk *memory space* atau kompleksitas komputasi.

Namun, tidak begitu saja cara yang ada untuk memaksimalkan efisiensi jalannya komputer program yaitu seperti apa yang disebut dengan pemrosesan paralel yang merupakan salah satu cara bagaimana sebuah program dieksekusi atau dilaksanakan [1]

Melihat teknologi yang terus berkembang, terutama di bidang teknologi *microprocessor* di mana satu *microprocessor* dapat mempunyai banyak inti/*core* seperti processor AMD Threadripper yang mempunyai 32 *core* kita harus dapat memanfaatkan agar *core-core* yang ada digunakan secara efisien dan tidak terbuang begitu saja.

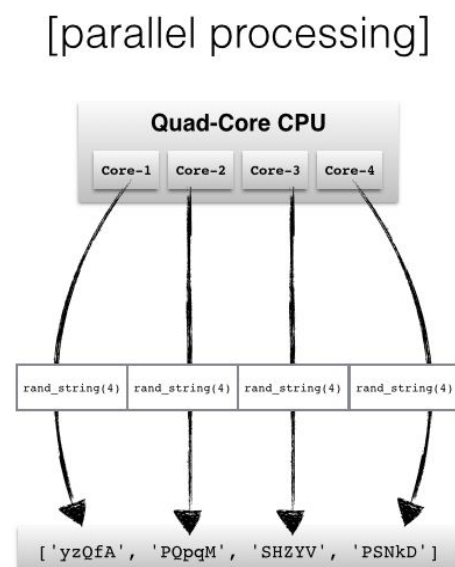
Jalannya setiap program yang dibuat menggunakan bahasa pemrograman Python pasti hanya di-*utilize* 1 *core* saja disebabkan oleh pengunci interpretasi global atau yang biasa disebut dengan *Python GIL*. *Python GIL* merupakan pengaman agar jalannya program dan setiap *thread* dari

program tersebut tidak dieksekusi secara bersamaan yang bisa menyebabkan penulisan memori yang salah [2].

Untuk melewati batasan ini, di mana hanya satu unit *CPU core* saja yang digunakan, Python menyediakan *library/modul* khusus yang menanggung masalah komputasi paralelisme yaitu *multiprocessing library*.

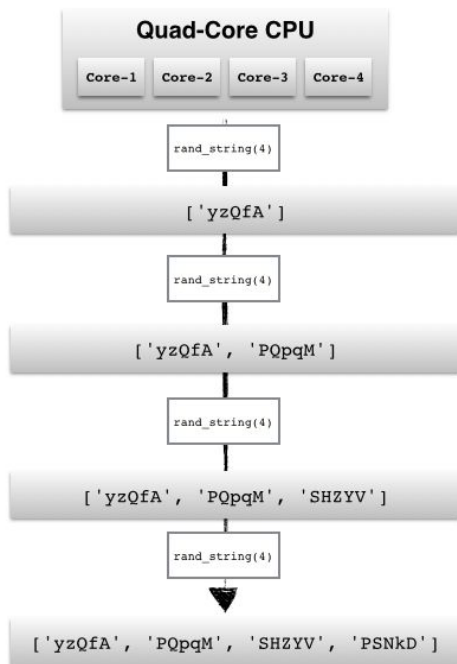
II. PEMROSESAN PARALEL

Pemrosesan paralel adalah komputasi dua pekerjaan atau lebih yang dilakukan secara bersamaan yang bertujuan untuk mempersingkat waktu eksekusi dengan cara mengoptimalkan *resource* pada sistem komputer yang tersedia. [3] Pemrosesan paralel dapat mempersingkat waktu pengerjaannya karena membagi sebuah proses besar menjadi bagian - bagian yang lebih kecil untuk bisa dikerjakan bersama oleh prosesor yang tersedia. Suatu program yang dikerjakan oleh *n* prosesor diharapkan dapat mempersingkat *n* kali daripada hanya ketika dikerjakan oleh satu prosesor. Berikut adalah gambaran perbedaan antara pemrosesan serial/sekuensial dan paralel :



Gambar 2.1 - Ilustrasi Pemrosesan Paralel

[serial processing]



Gambar 2.2 - Ilustrasi Pemrosesan Serial.

III. PYTHON

Python merupakan bahasa pemrograman tingkat tinggi yang dinamis dikembangkan oleh Python Software Foundation di bawah arahan Guido van Rossum pada tahun 1989(diperkenalkan pada tahun 1991). Beberapa alasan mengapa kami menggunakan bahasa pemrograman Python adalah karena Python mudah digunakan, memiliki kompatibilitas dan kemampuan tingkat tinggi, mendukung pemrograman berorientasi objek, *platform independent*, gratis, dan *open source*. [4]

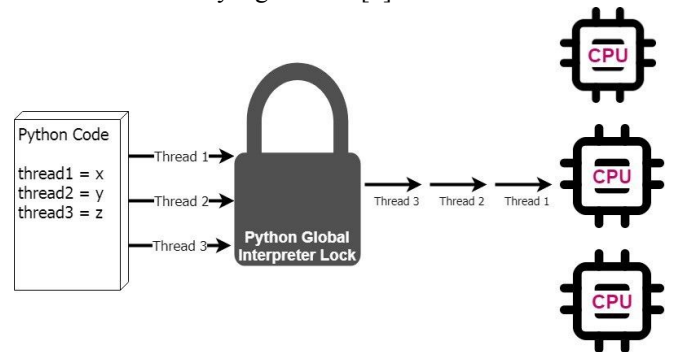
Dalam melakukan pemrosesan paralel kami menggunakan modul *built-in* yang bernama *multiprocessing*. Karena modul *multiprocessing* adalah modul *built-in*, maka pada saat pemasangan Python Interpreter(hanya pada Windows) modul *multiprocessing* juga langsung terpasang. Pada modul ini terdapat banyak fungsi-fungsi untuk melakukan pemrosesan paralel dengan bahasa pemrograman Python. [4]

IV. PYTHON GIL

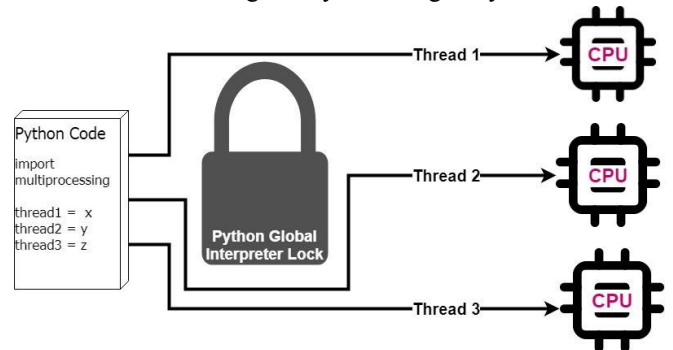
Python GIL kependekan dari Python Global Interpreter Lock merupakan *mutual exclusion* atau *mutex* yang merupakan objek yang mengontrol aktivitas *concurrency* atau aktivitas yang melibatkan komputasi paralel sehingga tidak ada terjadinya penulisan atau pengaksesan banyak *thread* dalam jalannya suatu aplikasi python. Presensi Python GIL ini disebabkan oleh management memori dari CPython yang merupakan *interpreter* official developer python yang tidak *thread-safe* atau tidak aman untuk *thread*. Dokumentasi resmi dari python sendiri menyatakan bahwa ada cara untuk melepaskan Python GIL ini sendiri, namun disarankan untuk mengikuti prasyarat seperti :

1. Program harus simpel dan gampang dipelihara/dikembangkan.

2. Program membutuhkan kecepatan.
3. Program membutuhkan fitur seperti *garbage collection* yang manual. [2]



Gambar 4.1 - Program Python dengan Python GIL.



Gambar 4.2 - Program Python tanpa Python GIL.

V. LIBRARY MULTIPROCESSING

Multiprocessing adalah sebuah modul/library di dalam bahasa pemrograman Python yang dapat memperbanyak proses menggunakan API yang mirip dengan modul/library *threading*. *Multiprocessing* dapat juga digunakan secara kongruen melalui lokal maupun *remote*, yang kerjanya adalah melewati GIL dengan sub-subproses. Sehingga modul/library ini dimanfaatkan untuk memakai beberapa prosesor sekaligus pada komputer si pemrogram dalam mengembangkan sebuah program.

Modul *multiprocessing* juga memiliki API yang tidak ada analognya seperti yang ada pada modul *threading*. Contohnya adalah objek/fungsi yang bernama *Pool*, dimana fungsi/objek tersebut dapat digunakan untuk melakukan pemrosesan paralel. Berikut adalah beberapa fungsi/objek yang digunakan untuk melakukan sebuah pemrosesan paralel sederhana. [5]

A. Class Process Library Multiprocessing

Multiprocessing library memiliki *process class* yang berfungsi sebagai pengatur *python objects* dan menginterupsinya untuk dikenakan proses yang lain. [5] Proses yang lain inilah yang akan memanipulasi objek tadi. Pengatur di sini berarti :

1. Mengontrol proses yang mengatur objek yang dibagikan.
2. Memastikan objek yang dibagikan tadi diperbaharui di semua proses ketika siapapun mengubahnya.

B. Class Pipe Library Multiprocessing

Multiprocessing library memiliki *class Pipe* yang merupakan penyediaan dari *library multiprocessing*

agar process atau fungsi yang di **start()** oleh *class Process* dapat berkomunikasi atau bertukar data. *Class Pipe* bekerja dengan memanggil 2 *object connection* yang bisa ditentukan. Sifat dari object ini bisa ditentukan agar kedua object bisa saling mengirim dan menerima data atau satu saja yang mengirim data dan sisanya yang menerima [5].

C. Class Queue Library Multiprocessing

Multiprocessing library memiliki *class Queue* yang juga merupakan penyediaan dari *library multiprocessing* yang sejenis dengan *class Pipe*, yaitu agar process atau fungsi yang di **start()** bisa berkomunikasi. Class ini bekerja dengan cara menyediakan **Queue FIFO** yang dapat diakses oleh setiap proses, dan setiap kejadian dimana proses mengakses **Queue** maka akan terjadi dequeue biasa sehingga tidak ada kejadian pemrosesan inputan variabel yang sama [5].

D. Class Pool Library Multiprocessing

Multiprocessing library memiliki *pool class* yang mendistribusikan pekerjaan antara pekerja dan mengumpulkan nilai kembalian dalam daftar. [5]. *Pool class* memiliki keuntungan, yaitu :

1. *User* tidak perlu khawatir untuk mengatur antrian, proses - proses, dan waktu/status yang dibagikan.
2. *Pool class* memudahkan untuk mengimplementasikan program konkurensi singkat/serhana.

VI. HASIL PERCOBAAN

Kami telah melakukan percobaan pemrosesan paralel sederhana, yakni dengan menentukan apakah bilangan tersebut termasuk bilangan ganjil/genap dari perpangkatan angka 1 sampai 10. Berikut adalah spesifikasi komputer yang kami gunakan untuk melakukan pemrosesan paralel :

```

ikal@hans: ~/Desktop
File Edit Tabs Help
ikal@hans:~/Desktop$ python3 oarkom.py
Python version : 3.6.7
compiler       : GCC 8.2.0

system        : Linux
release       : 4.15.0-42-generic
machine       : x86_64
processor     : x86_64
CPU count    : 4
interpreter   : 64bit

```

Gambar 6.1 - Spesifikasi Arsitektur Komputer

Kami melakukan tiga percobaan yakni, percobaan pertama kami melakukan pemrosesan serial, kedua kami melakukan pemrosesan paralel dengan modul *multiprocessing.Process*, ketiga kami melakukan pemrosesan paralel dengan modul *multiprocessing.Pool*. Berikut adalah hasil percobaan kami :

Jenis Pemrosesan	Waktu
Serial (memroses sekuensial biasa)	10.013 detik
<i>multiprocessing.Process</i>	1.027 detik
<i>multiprocessing.Pool</i>	3.174 detik

Tabel 6.1 - Hasil percobaan

ID proses dari setiap proses yang kami jalankan secara serial adalah sama yakni 1881, sedangkan ID proses dari pemrosesan paralel menggunakan library/modul *multiprocessing.Process* dan *multiprocessing.Pool* adalah berbeda tiap proses-prosesnya. Hal ini sesuai dengan konsep pemrosesan paralel yang telah kami jelaskan di atas. Kode sumber kami dapat dilihat di sini <https://github.com/Rakhid16/OArKom>.

```

ikal@hans: ~/Desktop
File Edit Tabs Help
ikal@hans:~/Desktop$ python3 ter1.py
0 pangkat 2 adalah bilangan 0
1881
1 pangkat 2 adalah bilangan ganjil
1881
2 pangkat 2 adalah bilangan genap
1881
3 pangkat 2 adalah bilangan ganjil
1881
4 pangkat 2 adalah bilangan genap
1881
5 pangkat 2 adalah bilangan ganjil
1881
6 pangkat 2 adalah bilangan genap
1881
7 pangkat 2 adalah bilangan ganjil
1881
8 pangkat 2 adalah bilangan genap
1881
9 pangkat 2 adalah bilangan ganjil
1881
That took 10.013223886489868 seconds
ikal@hans:~/Desktop$

```

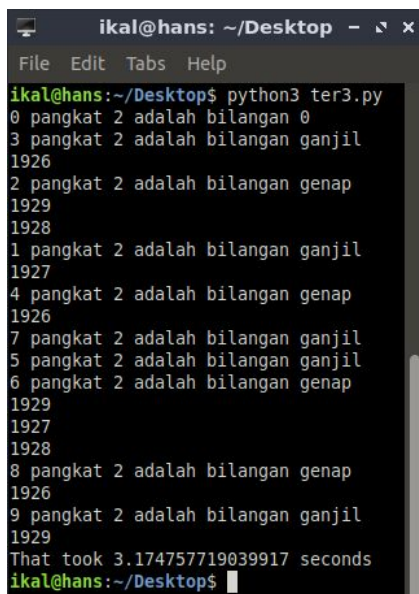
Gambar 6.2 - Menggunakan pemrosesan serial/sekuensial

```

ikal@hans: ~/Desktop
File Edit Tabs Help
ikal@hans:~/Desktop$ python3 ter2.py
0 pangkat 2 adalah bilangan 0
1911
1 pangkat 2 adalah bilangan ganjil
1912
2 pangkat 2 adalah bilangan genap
1913
3 pangkat 2 adalah bilangan ganjil
1914
4 pangkat 2 adalah bilangan genap
1915
5 pangkat 2 adalah bilangan ganjil
1916
6 pangkat 2 adalah bilangan genap
1917
7 pangkat 2 adalah bilangan ganjil
1918
8 pangkat 2 adalah bilangan genap
1919
9 pangkat 2 adalah bilangan ganjil
1920
That took 1.0276925563812256 seconds
ikal@hans:~/Desktop$

```

Gambar 6.3 - Menggunakan *multiprocessing.Process*



```
ikal@hans: ~/Desktop - x
File Edit Tabs Help
ikal@hans:~/Desktop$ python3 ter3.py
0 pangkat 2 adalah bilangan 0
3 pangkat 2 adalah bilangan ganjil
1926
2 pangkat 2 adalah bilangan genap
1929
1928
1 pangkat 2 adalah bilangan ganjil
1927
4 pangkat 2 adalah bilangan genap
1926
7 pangkat 2 adalah bilangan ganjil
5 pangkat 2 adalah bilangan ganjil
6 pangkat 2 adalah bilangan genap
1929
1927
1928
8 pangkat 2 adalah bilangan genap
1926
9 pangkat 2 adalah bilangan ganjil
1929
That took 3.174757719039917 seconds
ikal@hans:~/Desktop$
```

Gambar 6.4 Menggunakan multiprocessing.Pool

A. Kesimpulan

Setelah dilakukan percobaan, didapatkan hasil bahwa Pemrosesan paralel lebih menghemat waktu daripada pemrosesan serial/sekuensial karena pada pemrosesan paralel data diolah secara bersamaan dengan proses yang berbeda. Sedangkan pada pemrosesan sekuensial data diolah satu persatu dengan satu proses yang sama. Selain itu fungsi multiprocessing. Pool terbatas hanya pada empat proses saja, disesuaikan dengan jumlah CPU.

REFERENSI

- [1] Gottlieb, Allan; Almasi, George S. (1989). [*Highly parallel computing*](#). Redwood City, Calif.: Benjamin/Cummings. [ISBN 0-8053-0177-1](#)
- [2] Van Rossum Guido, The Python/C API, Delaware: Python Software Foundation, 2018
- [3] Mateti, Prabhaker, Cluster Computing with Linux, Ohio: Wright State University, 2005.
- [4] Sianpar, R.H. & Wadi, Hamzan. Pemrograman Python (teori dan implementasi), Bandung: Informatika, 2015
- [5] Van Rossum Guido, The Python Library Reference, Delaware: Python Software Foundation, 2018.