JARINGAN DAN PENGOLAHAN DATA PARALEL

LAPORAN PRAKTIKUM

Mohammad Rizka Fadhli

Ikang

20921004@mahasiswa.itb.ac.id

31 December 2021

Contents

List of Figures

List of Tables

1 *INTRODUCTION*

1.1 Definisi

Parallel processing adalah metode komputasi untuk menggunakan dua atau lebih *processors* untuk menjalankan beberapa tugas secara terpisah atau secara keseluruhan. Setiap komputer yang memiliki lebih dari satu *CPUs* atau memiliki *processor multi cores* bisa melakukan *parallel processing.*[1]

1.2 Perbedaan *Serial Processing* dan *Parallel Processing*

Perbedaan mendasar dari *serial processing* dan *parallel processing* adalah dari segi bagaimana komputer melakukan proses komputasi. *Serial processing* berarti komputer melakukan tugasnya secara sekuensial (berurutan) menggunakan satu *processor*. Akibatnya adalah saat melakukan suatu proses yang kompleks, *runtime* yang diperlukan lebih lama karena *processor* harus memproses data satu-persatu.

Berbeda halnya dengan *parallel processing*. Tugas yang dilakuan komputer didistribusikan kepada sejumlah *processors* untuk diolah secara bersamaan. Konsekuensinya adalah *runtime* komputasi lebih singkat. Namun perlu diperhatikan dengan seksama bahwa tidak semua tugas bisa kita buat paralelisasinya dan cara kita menulis algoritma atau *coding* harus disesuaikan.

> Kenapa tidak semua tugas bisa diparalelisasi?

Beberapa tugas sekuensial yang tidak bisa dihindari tidak bisa diparalelisasi.

Sebagai contoh:

1. *Looping* yang prosesnya tidak saling bergantung bisa diparalelisasi. Misalkan ada suatu fungsi untuk menghitung suatu *array* bisa diparalelisasi dengan cara memecah *array* tersebut untuk diproses bersamaan di beberapa *processors*.
2. *Looping* yang prosesnya saling bergantung tidak bisa diparalelisasi. Misalkan suatu *looping* ke i nilainya bergantung pada proses *looping* ke $i-1$.

[1] https://searchdatacenter.techtarget.com/definition/parallel-processing

Berikut adalah ilustrasi perbedaan serial dan *parallel processing*:

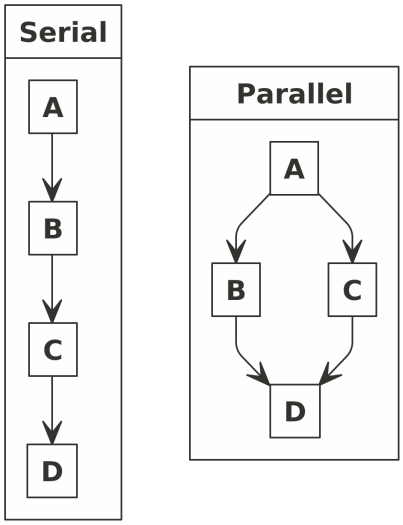

Figure 1: Ilustrasi Perbedaan Serial dan Parallel Processing

1.3 Cara Kerja *Parallel Processing*

Untuk melakukan *parallel processing*, dibutuhkan *hardware* dan *software* yang mendukung hal tersebut. Secara *hardware* dibutuhkan komputer dengan *multiple cores processors* atau dibutuhkan beberapa komputer yang digabung menjadi satu kesatuan. Secara *software* dibutuhkan tidak hanya `Python` tapi juga *middleware* bernama `Open MPI`. Bagian *hardware* dan *software* ini akan dibahas pada bagian selanjutnya.

Pada sistem *parallel processing* terdiri dari beberapa unit *processors* dan beberapa unit *memory*. Ada dua teknik berbeda yang digunakan untuk mengakses data di unit *memory*, yaitu: *shared memory address* dan *message passing*.

Berdasarkan cara mengorganisasikan memori ini komputer bisa dibedakan menjadi *shared memory parallel machine* dan *distributed memory parallel machine*.

Ada empat model komputasi yang dikenal dalam taksonomi Flynn, yaitu:

1. **SISD** (*Single Instruction, Single Data*)
2. **SIMD** (*Single Instruction, Multiple Data*)
3. **MISD** (*Multiple Instruction, Single Data*)
4. **MIMD** (*Multiple Instruction, Multiple Data*)

1.3.1 SISD

Komputer ini adalah tipikal komputer konvensional yang hanya memiliki satu *processor* dan satu instruksi yang dieksekusi secara serial. Komputer jenis ini tidak bisa melakukan *parallel processing*.

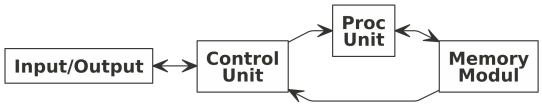

Figure 2: Ilustrasi SISD

1.3.2 SIMD

Komputer ini memiliki lebih dari satu *processor* tapi hanya mengeksekusi satu instruksi secara paralel pada data yang berbeda pada level *lock-step*. Contohnya adalah komputer vektor.

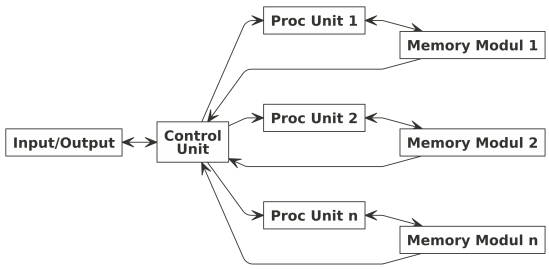

Figure 3: Ilustrasi SIMD

1.3.3 MISD

Komputer jenis ini belum diciptakan karena secara arsitekturnya tidak mudah dipahami. Secara teori komputer ini memiliki satu *processor* dan mengeksekusi beberapa instruksi secara paralel.

1.3.4 MIMD

Komputer berarsitektur ini paling banyak digunakan untuk membangun *super computer*. Komputer ini memiliki lebih dari satu *processors* dan mengeksekusi lebih dari satu instruksi secara paralel.

1.4 TUGAS PRAKTIKUM

Pada praktikum ini, saya akan mengerjakan:

- Dua buah tugas terkait penyelesaian integral secara numerik memanfaatkan metode dikritisasi nilai tengah (*midpoint*) dan simulasi Monte Carlo.
- Satu buah tugas terkait penjumlahan dan perkalian matriks $n \times n$.

Ketiga tugas tersebut akan diselesaikan menggunakan serial dan *parallel processing*.

1.5 *SERVER* YANG DIGUNAKAN

Pada praktikum kali ini, saya tidak bisa menggunakan *server* **HPC** yang disediakan oleh ITB karena masalah koneksi. Oleh karena itu, saya menggunakan *server* lain agar bisa menduplikasi apa yang seharusnya dikerjakan di *server* ITB.

Saya menggunakan *server virtual machine* milik *Google Cloud*[2]. *Server* ini memiliki *processor* **Intel Xeon 8 *cores***. *Hostname* dari *server* ini saya beri nama `praktikum`.

[2] https://ikanx101.com/blog/vm-cloud/

Berikut adalah spesifikasinya:

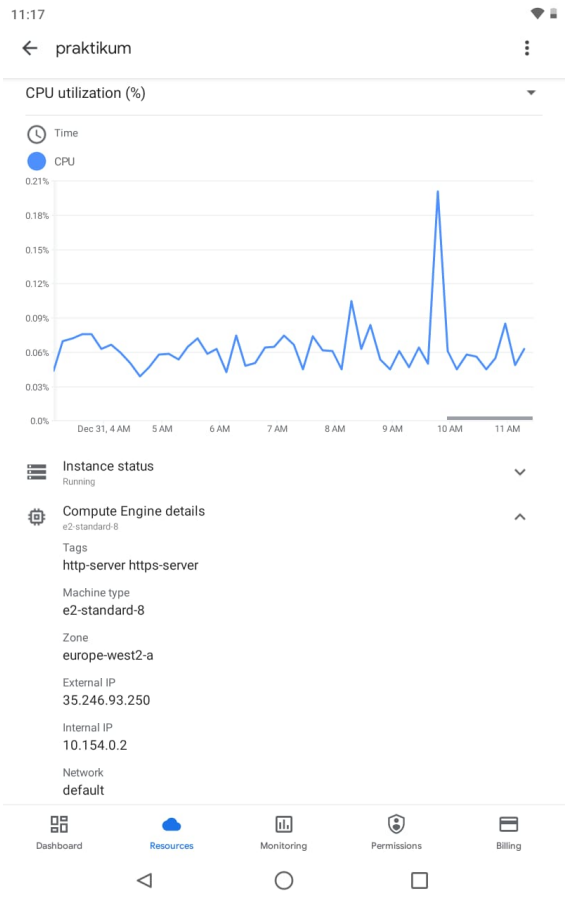

Figure 4: Spesifikasi Server yang Digunakan

Server ini bisa diakses menggunakan *command line* menggunakan `ssh` langsung ke *IP Public* yang diberikan *Google*.

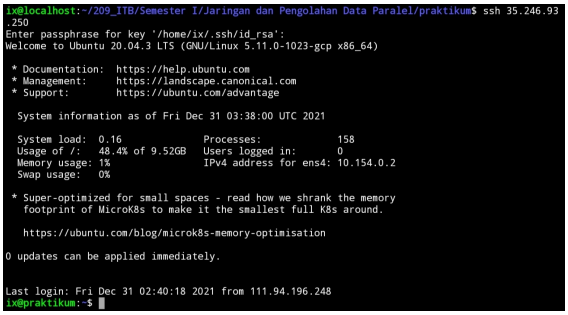

Figure 5: Tampilan Awal Setelah Login ssh

Server ini berjalan di *operating system* `Ubuntu Linux 20.04 LTS`.

Berikut adalah tampilan hasil `lscpu`:

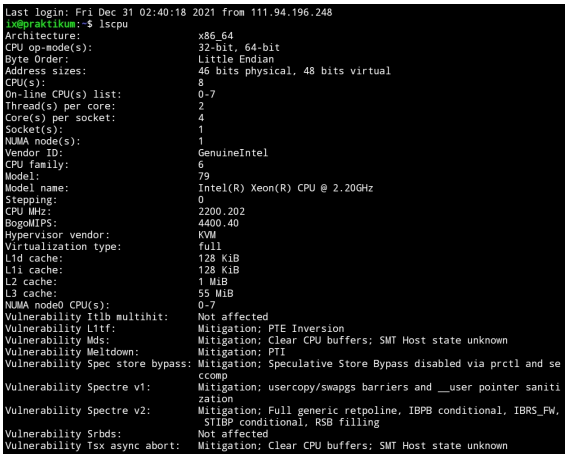

Figure 6: lscpu dari Server

Berikut adalah tampilan hasil `htop`:

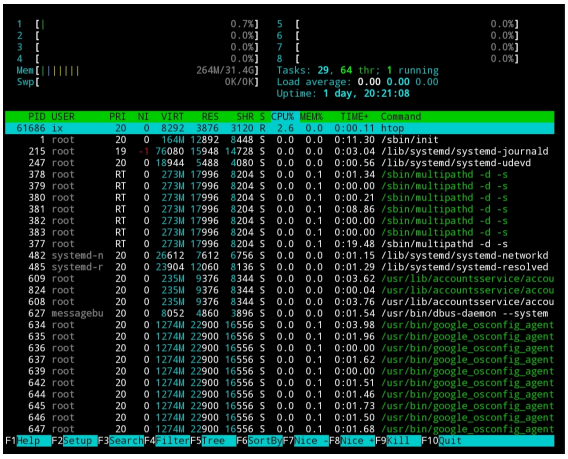

Figure 7: htop dari Server

1.6 MPI

Secara *default* `Python` sudah ter-*install* di *server* namun tidak untuk `Open MPI`. Oleh karena itu, salah satu langkah yang perlu dilakukan adalah meng-*install*-nya terlebih dahulu.

Pada `Ubuntu`, proses instalasinya bisa kita lakukan dengan mengetikkan perintah berikut ini di *command line*:

```
sudo apt install openmpi-bin openmpi-dev openmpi-common openmpi-doc libopenmpi-dev
```

Setelah proses instalasi selesai, kita bisa mengecek versi `Open MPI` yang berjalan di *server* sebagai berikut:

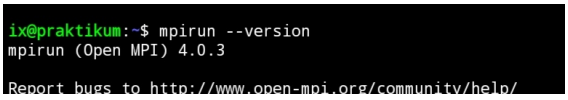

Figure 8: Versi MPI yang Digunakan

2 *METHOD*

Pada praktikum ini, kita akan melakukan *parallel processing* menggunakan `Python` versi `3.8.10` di *server* berbasis `Linux Ubuntu OS`. Ada beberapa metode *parallel processing* yang hendak dilakukan, yakni:

1. *Broadcast*,
    1. *Broadcast-gather*,
    2. *Broadcast-reduce*,
2. *Scatter*,
    1. *Scatter-reduce*,
    2. *Scatter-gather*,
3. *Gather*,
4. *Reduce*,
5. *Multi-processing*,
6. *Multi-thread*,
7. *Point-to-point*.

Kemudian semua metode *parallel processing* ini akan dibandingkan *runtime*-nya dengan *serial processing*.

2.1 Metode Integral Numerik

2.1.1 Metode Titik Tengah (*Midpoint*)

2.1.2 Simulasi Monte Carlo

3 *RESULT AND DISCUSSION*

3.1 Soal I

3.2 Soal II

Perhitungan π menggunakan rumus: $4 \times sqrt1 - x^2$

3.3 Soal III

4 *CONCLUSION*

lalala (Hillier and Lieberman 2001)

REFERENCES

Hillier, Frederick S., and Gerald J. Lieberman. 2001. *Introduction to Operations Research.* 7th ed. New York, US: McGraw Hill. www.mhhe.com.