Nama : **Arjuna Putra**

Nim : **20231337020**

Matkul : **Komputasi Parallel Dan Terdistribusi**

Jawab

1. Elemen Penting dalam Komputasi Paralel

Komputasi paralel adalah sistem terintegrasi yang menggabungkan elemen keras dan lunak untuk bekerja secara serentak. Berikut adalah empat elemen utamanya:

- Node Komputasi (Processing Elements): Merupakan unit pemroses utama ("otak" sistem), seperti CPU multi-core atau komputer mandiri dalam sebuah klaster. Tugas utamanya adalah mengeksekusi instruksi program secara nyata.
- Arsitektur Memori: Wadah penyimpanan data yang akan diolah. Terdapat dua jenis utama: Shared Memory (satu memori diakses bersama, mudah diprogram) dan Distributed Memory (setiap prosesor memiliki memori sendiri, lebih cepat untuk skala besar).
- Jaringan Interkoneksi: Jalur komunikasi fisik (seperti kabel fiber optik atau bus sistem) yang menghubungkan antar node dan memori. Efisiensinya sangat bergantung pada bandwidth (kapasitas data) dan latency (kecepatan respon).
- Perangkat Lunak Paralel: Sistem yang mengatur agar hardware dapat bekerja bersamaan. Ini mencakup Sistem Operasi, compiler, dan library khusus (seperti MPI atau OpenMP) untuk memecah tugas besar menjadi bagian-bagian kecil.

2. Variabel Utama Klasifikasi Flynn (1966)

Michael J. Flynn memperkenalkan sistem klasifikasi pada tahun 1966 yang dikenal sebagai Taksonomi Flynn. Klasifikasi ini didasarkan pada dua variabel utama, yaitu Alur Instruksi (Instruction Stream) yang mengacu pada jumlah perintah yang diproses CPU, dan Alur Data (Data Stream) yang mengacu pada jumlah data yang dimanipulasi oleh instruksi tersebut pada satu waktu. Kombinasi dari kedua variabel ini menjadi fondasi untuk membedakan berbagai jenis arsitektur komputer yang ada hingga saat ini.

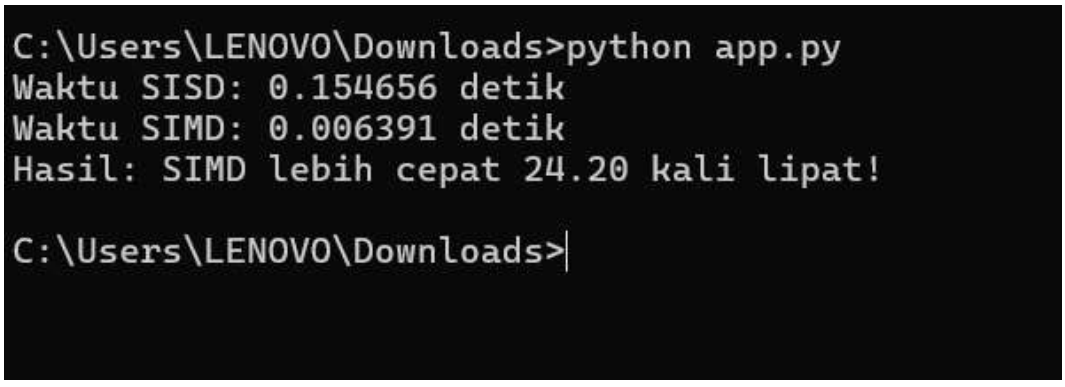
- Pengelompokan ini dianggap sangat vital dalam perkembangan komputasi paralel karena alasan berikut:
- Standarisasi Bahasa Teknik: Memberikan kosakata standar (seperti istilah SIMD atau MIMD) yang memungkinkan insinyur komputer di seluruh dunia untuk berkomunikasi dan membandingkan spesifikasi mesin yang berbeda dengan bahasa yang seragam.
- Panduan Desain Arsitektur: Menjadi kerangka kerja bagi perancang chip untuk memahami keseimbangan (trade-off) antara kompleksitas pengaturan instruksi dan kecepatan aliran data, sehingga arsitektur dapat dibuat lebih efisien sesuai tujuannya.
- Evolusi GPU dan Superkomputer: Memperjelas perbedaan jalur pengembangan antara CPU modern (umumnya MIMD) dan GPU (umumnya SIMD). Hal ini sangat penting untuk mendorong inovasi spesifik pada akselerator grafis dan teknologi kecerdasan buatan (AI).

3. Perbandingan SIMD, MIMD, dan Teknologi Hybrid.

Aspek	SIMD	MIMD	Hybrid
Pendekatan	Satu perintah dieksekusi serentak oleh banyak unit prosesor pada banyak data yang berbeda.	Banyak perintah berbeda dieksekusi secara independen oleh banyak prosesor pada data yang berbeda.	Menggabungkan arsitektur kontrol MIMD (sebagai pengelola) dengan unit eksekusi SIMD (sebagai akselerator).
Fleksibilitas	Rendah. Sangat bergantung pada keseragaman data. Kinerja turun jika banyak percabangan (if-else).	Tinggi. Setiap prosesor bebas mengerjakan tugas apa saja (asinkron), tidak harus seragam.	Terpadu/Seimbang. Menggunakan fleksibilitas MIMD untuk logika kontrol dan kekuatan SIMD untuk hitungan matematis.

Kompleksitas	Sedang. Desain hardware lebih sederhana (satu unit kontrol), namun pemrograman harus terstruktur (vektorisasi).	Tinggi. Membutuhkan sinkronisasi yang rumit antar prosesor agar tidak terjadi konflik data (deadlock).	Sangat Tinggi. Paling sulit diprogram karena harus mengelola komunikasi antar node dan akselerasi di dalam node sekaligus.
Aplikasi Utama	Pemrosesan Grafik (GPU), Multimedia (Audio/Video), Matriks.	Server Database, Sistem Operasi Multitasking, Klaster Web.	Superkomputer (HPC), Pelatihan AI (Deep Learning), Simulasi Cuaca Global.
Kinerja	Sangat efisien untuk tugas yang berulang dan struktur datanya sama (high throughput).	Efisien untuk tugas yang kompleks, acak, dan bervariasi (low latency).	Optimal. Memberikan kinerja terbaik untuk sistem skala besar dengan memaksimalkan kelebihan kedua arsitektur.

4. **Analisis Perbandingan Performa Arsitektur SISD dan SIMD dalam Operasi Aritmatika Skala Besar**



Berdasarkan hasil uji coba perkalian **8 X 2.000.000** elemen, metode SIMD (Single Instruction, Multiple Data) terbukti jauh lebih cepat dan efektif dibandingkan metode SISD (Single Instruction, Single Data). Dalam eksperimen, SIMD menunjukkan peningkatan performa hingga 24.20 kali lipat dengan waktu eksekusi yang hampir instan (0,15 detik),

sementara SISD membutuhkan waktu jauh lebih lama (0,17 detik). Hal ini membuktikan bahwa pengolahan data secara paralel di level instruksi sangat krusial untuk efisiensi komputasi skala besar.

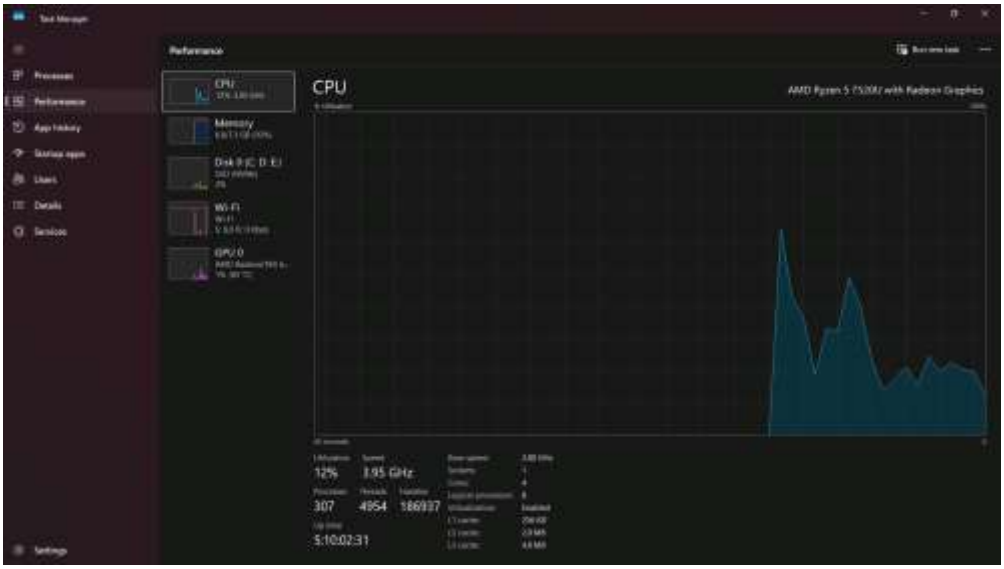
Perbandingan Efektivitas

- SIMD (NumPy): Sangat efektif karena menggunakan teknik Vectorization, di mana satu instruksi memproses beberapa elemen data sekaligus menggunakan register vektor pada CPU.
- SISD (Loop Biasa): Kurang efektif karena memproses data secara serial (satu per satu) dan memiliki overhead tinggi akibat pengecekan tipe data berulang oleh interpreter Python.
- Akses Memori: SIMD lebih efisien karena mengakses data yang tersusun rapat di memori (contiguous memory), sehingga mempercepat kinerja cache prosesor

Pembebanan (Workload) yang Diberikan

Pembebanan yang digunakan dalam pembuktian ini meliputi:

- Skala Data: 2.000.000 elemen data dalam satu array.
- Operasi Komputasi: Perkalian skalar tunggal dengan angka 8.
- Metode Pengukuran: Menggunakan time.perf_counter() untuk menangkap selisih waktu eksekusi dalam skala milidetik secara akurat.



5. **Traditional Sequential Processing vs Parallel Processing**

   a. **Perbedaan Mendasar Pengiriman Instruksi**

   Perbedaan utamanya terletak pada antrean dan jumlah jalur eksekusi:

   - Traditional Sequential Processing (SISD): Instruksi dikirim secara satu per satu dalam satu garis lurus (serial). CPU hanya menerima dan menyelesaikan satu instruksi sebelum melanjutkan ke instruksi berikutnya. Jika ada jutaan data, CPU harus mengulang proses ambil-eksekusi-simpan sebanyak jutaan kali secara berurutan.
   - Parallel Processing: Masalah atau beban kerja dipecah menjadi beberapa bagian yang lebih kecil. Instruksi-instruksi tersebut dikirimkan secara bersamaan (simultan) ke beberapa unit pemrosesan (Multi-CPU atau Multi-Core). Dalam model ini, banyak instruksi diproses pada waktu yang sama.

   b. **Mengatasi Masalah Bottleneck pada Data Skala Besar**

   Model Parallel Processing mengatasi kemacetan data (bottleneck) dengan cara:

   - Pembagian Beban (Divide and Conquer): Data skala besar tidak dipaksakan melewati satu pintu sempit (satu core), melainkan disebar ke seluruh core yang tersedia. Ini mencegah satu unit pemrosesan bekerja terlalu keras (overload) sementara unit lainnya menganggur.
   - Reduksi Waktu Eksekusi: Karena banyak instruksi dijalankan pada detik yang sama, total waktu yang dibutuhkan untuk menyelesaikan tugas menjadi jauh lebih singkat. Seperti hasil eksperimen Anda sebelumnya, perkalian 2 juta data yang memakan waktu lama di SISD bisa selesai hampir instan di model paralel karena dikerjakan secara gotong royong oleh hardware.
   - Efisiensi Bandwidth Memori: Parallel processing sering kali menggunakan teknik vectorization (seperti pada SIMD) yang memungkinkan pengambilan blok data besar sekaligus, sehingga mengurangi waktu tunggu CPU terhadap memori.

Parallel Processing jauh lebih cepat karena mengerjakan banyak tugas sekaligus secara gotong royong, sedangkan Sequential Processing lambat karena harus mengantre

satu per satu. Dengan membagi beban ke banyak jalur, model paralel menghilangkan hambatan (bottleneck) sehingga jutaan data bisa selesai diproses hampir instan.