MODUL 8
Evaluasi Performa Paralel

A. Tujuan Praktikum

1. Mengukur dan menganalisis performa eksekusi paralel menggunakan *multi-core* pada komputer pribadi.
2. Menghitung nilai Speedup, Efficiency, dan memahami batas performa berdasarkan Amdahl's Law.

B. Materi

1. Evaluasi Performa Paralel

   Dalam dunia komputasi modern, hampir semua komputer memiliki multi-core processor. Untuk memanfaatkan kemampuan ini, program harus ditulis agar bisa berjalan secara paralel. Namun seberapa besar peningkatan performa yang bisa didapat jika menggunakan banyak core adalah dengan melakukan evaluasi performa paralel, yang berfokus pada tiga metrik utama:

   1. Speedup (percepatan eksekusi)
   2. Efficiency (efisiensi pemanfaatan core)
   3. Amdahl's Law (batas teoritis peningkatan kecepatan)

   Komputasi paralel bertujuan untuk mempercepat waktu eksekusi program dengan cara membagi pekerjaan menjadi beberapa bagian yang dapat dijalankan secara bersamaan (concurrently) oleh banyak prosesor atau core CPU. Jika program 100% bisa diparalelkan dan memiliki 4 core, maka secara teori diselaikan 4x lebih cepat. Namun kenyataannya tidak karena tidak semua bagian program bisa diparalelkan.

2. Speedup (S)

   Dalam komputasi paralel speedup (s) adalah ukuran seberapa cepat sebuah program dapat dijalankan secara paralel dibandingkan dengan versi serialnya. Tujuan utama speedup dari komputasi paralel adalah untuk mengurangi waktu eksekusi suatu program dengan memanfaatkan banyak prosesor/core agar beberapa bagian pekerjaan bisa dilakukan secara bersamaan. Speedup mengukur *tingkat percepatan kinerja* yang diperoleh karena adanya paralelisme.

$$S = \frac{Tserial}{Tparalel}$$

   Keterangan:
   S : Nilai Speedup
   Tserial : Waktu eksekusi program secara serial (1 core)
   Tparalel : Waktu eksekusi program secara paralel (N core)

   Perhatikan tabel berikut:

Nilai Speedup (S)	Arti	Interpretasi
S = 1	Tidak ada percepatan	Program tidak mendapat manfaat dari paralelisme

S > 1	Ada percepatan	Program berjalan lebih cepat secara paralel
S = N	Speedup ideal	Semua core bekerja sempurna tanpa overhead
S < N	Speedup nyata	Umum terjadi karena overhead & bagian serial
S < 1	Penurunan performa	Paralelisme lebih lambat (karena overhead tinggi)

Nilai *speedup (S)* menunjukkan seberapa besar percepatan yang dicapai ketika program dijalankan secara paralel dibandingkan dengan versi serialnya. Jika S=1, tidak ada percepatan yang berarti, program paralel tidak memberikan manfaat apa pun. Jika S>1, program berhasil berjalan lebih cepat karena sebagian proses dapat dijalankan secara bersamaan. Nilai S=N menggambarkan kondisi ideal di mana semua core bekerja sempurna tanpa hambatan atau overhead, meskipun dalam praktiknya jarang terjadi. Sebaliknya, S<N menunjukkan *speedup nyata* yang umum ditemukan di dunia nyata akibat adanya overhead komunikasi, sinkronisasi, dan bagian program yang tetap serial. Namun, jika S<1, performa justru menurun karena proses paralel menambah beban koordinasi antarproses, sehingga waktu eksekusi total lebih lama daripada versi serial.

3. Efficiency (E)

   Efficiency (E) atau efisiensi paralel adalah ukuran seberapa efektif sistem paralel memanfaatkan sumber daya (misalnya jumlah core atau prosesor) untuk mempercepat eksekusi program. Efisiensi menunjukkan seberapa besar kontribusi rata-rata tiap core dalam mempercepat proses. Jika E = 1 maka setiap core bekerja sempurna dan tidak ada waktu yang terbuang (kondisi ideal). Jika E < 1 maka sebagian waktu core terbuang karena overhead komunikasi, sinkronisasi, atau idle time (menunggu core lain selesai). Jika E mendekati 0 maka sistem paralel tidak efisien; core banyak menganggur, atau ukuran pekerjaan terlalu kecil dibanding biaya koordinasi.

$$E = \frac{S}{N} = \frac{Ts}{N \times Tp}$$

   Keterangan:

   E : Efficiency (nilai antara 0 dan 1)
   S : Nilai Speedup
   N : Jumlah core/prosesor yang digunakan
   Ts : Waktu eksekusi program secara serial (1 core)
   Tp : Waktu eksekusi program secara paralel (N core)

   Perhatikan tabel berikut:

Nilai E	Kategori	Interpretasi
E ≈ 1	Sangat efisien	Semua core bekerja optimal tanpa banyak overhead
0.7 ≤ E < 1	Efisien	Overhead masih kecil
0.4 ≤ E < 0.7	Cukup efisien	Overhead dan bagian serial memengaruhi performa
0.1 ≤ E < 0.4	Kurang efisien	Banyak waktu terbuang akibat sinkronisasi dan idle
E < 0.1	Tidak efisien	Paralelisme tidak bermanfaat;

Nilai *efficiency (E)* menunjukkan seberapa efektif setiap core atau prosesor digunakan dalam komputasi paralel. Jika nilai E mendekati 1, berarti semua core bekerja optimal tanpa banyak waktu terbuang untuk komunikasi atau sinkronisasi, sehingga sistem sangat efisien. Ketika E berada pada rentang 0,7 hingga 1, paralelisme masih memberikan keuntungan

nyata meskipun ada sedikit overhead. Nilai efisiensi antara 0,4 hingga 0,7 menunjukkan bahwa sebagian performa hilang karena adanya koordinasi antarproses, namun sistem masih cukup bermanfaat. Jika nilai E turun di bawah 0,4, artinya banyak waktu yang terbuang karena proses harus idle, overhead komunikasi, atau beban kerja yang tidak seimbang antarcore. Sedangkan E<0,1 menandakan bahwa pemrosesan paralel tidak efisien dan justru lebih baik menjalankan program secara serial.

4. Amdahl's Law

Amdahl's Law adalah hukum dasar dalam komputasi paralel yang menjelaskan batas maksimum peningkatan kecepatan (speedup) dari suatu program ketika sebagian dari program tersebut dapat dijalankan secara paralel. Hukum ini dikemukakan oleh Gene Amdahl pada tahun 1967, ketika ia bekerja di IBM. Hukum ini muncul untuk mengetahui seberapa cepat suatu program bisa dijalankan jika sebagian besar kodenya dijalankan secara paralel. Tidak semua bagian program bisa diparalelkan. Selalu ada bagian yang harus tetap dijalankan secara serial (tidak bisa dibagi ke banyak core). Hukum Amdahl menyatakan bahwa Kecepatan total program paralel dibatasi oleh bagian yang tidak bisa diparalelkan. Artinya meskipun ada1000 core, jika 10% program tetap berjalan secara serial, speedup maksimum tidak akan lebih dari 10 kali lipat.

$$S(n) = \frac{1}{(1-p) + \frac{p}{n}}$$

Keterangan:

S(n) : Speepup maksimum dengan n prosesor
p : Proporsi bagian program yang bisa diparalelkan ($0 \leq p \geq 1$)
$(1-p)$: Proporsi bagian program yang tidak bisa diparalelkan
n : Jumlah prosesor (core)

Jika p kecil maka speedup rendah walaupun memakai banyak prosesor. Jika p besar (hampir seluruh program bisa paralel), speedup bisa mendekati jumlah prosesor, tapi tidak pernah lebih. Jika n → ∞ (sangat banyak prosesor), maka hal tersebut adalah batas teoritis maksimum speedup. Implikasi Hukum Amdahl adalah ada batasan alami pada speedup, Jika hanya 90% kode bisa paralel, speedup maksimum = 10 kali, meski punya 1000 prosesor. Optimasi bagian serial lebih penting daripada hanya menambah core, Jika bagian serial bisa diperkecil, efek paralelisasi meningkat drastis. Overhead komunikasi memperburuk efisiensi, menambah core justru bisa *memperlambat* jika beban komunikasi terlalu besar. Perhatikan contoh berikut:

Misalkan sebuah program terdiri dari 80% (p = 0.8) bagian bisa diparalelkan dan 20% (1-p = 0.2) bagian tetap serial.

Jumlah Core (n)	Rumus	Speedup (S)
1	$\frac{1}{(0.2 + 0.8) \div 1}$	1.00
2	$\frac{1}{(0.2 + 0.8) \div 2}$	1.67

4	$\frac{1}{(0.2 + 0.8) \div 4}$	2.50
8	$\frac{1}{(0.2 + 0.8) \div 8}$	3.33
16	$\frac{1}{(0.2 + 0.8) \div 16}$	3.81
∞	$\frac{1}{(0.2 + 0)}$	5.00 (batas maksimum)

Dari contoh dapat dilihat bahwa walaupun menambah prosesor tak terbatas, program ini tidak akan pernah lebih cepat dari 5 kali versi serialnya, karena 20% kode tidak bisa diparalelkan.

C. Langkah Praktikum

1. Perhatikan gambar 1, pada gambar program paralel berjalan 2,59 kali lebih cepat dibanding versi serial, menunjukkan paralelisme berhasil memberikan percepatan. Setiap core bekerja dengan efisiensi sekitar 81%, menunjukkan overhead komunikasi dan sinkronisasi masih relatif kecil.

$$S = \frac{Tserial}{Tparalel} = \frac{0,38}{0,1468} = 2,59 \; ; \; E = \frac{S}{N} = \frac{2,59}{10} = 0,26$$

```python

def kerja_berat(n):
    total = 0
    for i in range(1, n):
        total += i ** 0.5
    return total

def main():
    N = 10_000_000

    mulai_serial = time.time()
    kerja_berat(N)
    selesai_serial = time.time()
    T_serial = selesai_serial - mulai_serial
    print(f"Waktu Serial: {T_serial:.4f} detik")

    cpu_count = mp.cpu_count()
    print(f"Jumlah Core: {cpu_count}")

    mulai_paralel = time.time()
    with mp.Pool(cpu_count) as pool:
        pool.map(kerja_berat, [N // cpu_count] * cpu_count)
    selesai_paralel = time.time()
    T_paralel = selesai_paralel - mulai_paralel
    print(f"Waktu Paralel: {T_paralel:.4f} detik")


    S = T_serial / T_paralel
    E = S / cpu_count
    print(f"Speedup = {S:.2f}")
    print(f"Efficiency = {E:.2f}")

if __name__ == '__main__':
    main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/local/bin/python3 "/Users/yuliana/Komputasi Paralel/speedup_test.py"
yuliana@MacBook-Air-Yuliana ~ % /usr/local/bin/python3 "/Users/yuliana/Komputasi Paralel/speedup_test.py"
Waktu Serial: 0.3800 detik
Jumlah Core: 10
Waktu Paralel: 0.1468 detik
Speedup = 2.59
Efficiency = 0.26
yuliana@MacBook-Air-Yuliana ~ %
```

Gambar 1.

2. Perhatikan gambar 2. Meskipun 90% program bisa diparalelkan, percepatan maksimum hanya sekitar 3,08 kali pada 4 core. Artinya, bagian kecil yang tidak bisa diparalelkan (10%) tetap menjadi penghambat utama (*bottleneck*).

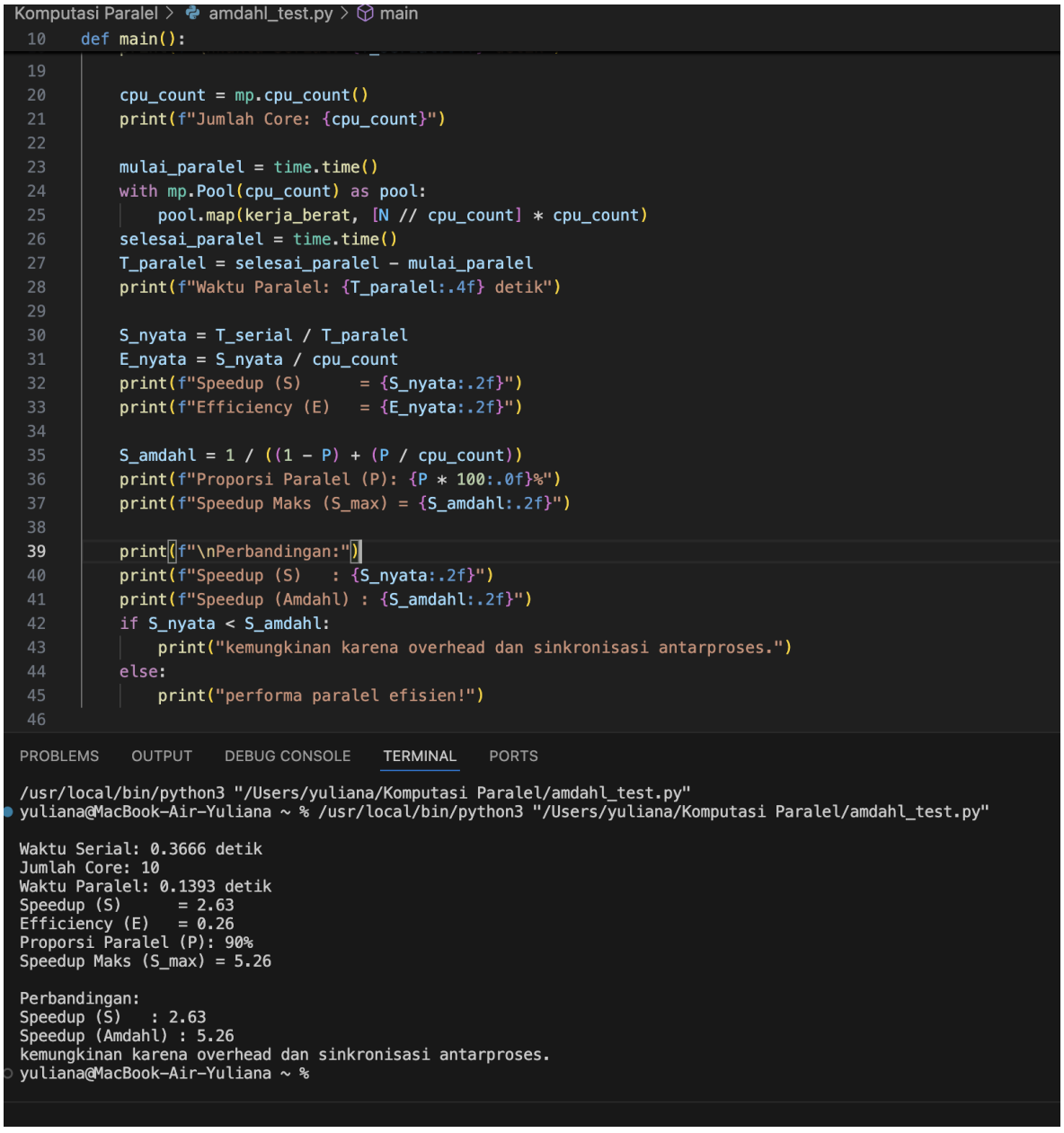
```python
# Proporsi bagian program yang bisa diparalelkan (P)
P = 0.9  # 90% kode dapat diparalelkan

for N in [1, 2, 4, 8, 16]:
    S_max = 1 / ((1 - P) + (P / N))
    print(f"N={N:2d} core → Speedup maksimum teoritis = {S_max:.2f}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/local/bin/python3 "/Users/yuliana/Komputasi Paralel/amdahl_test.py"
yuliana@MacBook-Air-Yuliana ~ % /usr/local/bin/python3 "/Users/yuliana/Komputasi Paralel/amdahl_test.py"
N= 1 core → Speedup maksimum teoritis = 1.00
N= 2 core → Speedup maksimum teoritis = 1.82
N= 4 core → Speedup maksimum teoritis = 3.08
N= 8 core → Speedup maksimum teoritis = 4.71
N=16 core → Speedup maksimum teoritis = 6.40
yuliana@MacBook-Air-Yuliana ~ %
```

Gambar 2.

Perhatikan gambar 3, program bisa dipercepat hingga 5,26x dengan 10 core, dalam praktiknya hanya tercapai 2,63x karena adanya overhead dan inefisiensi dalam implementasi paralel. Efisiensi 26% menandakan sebagian besar waktu masih habis untuk koordinasi antarproses dibanding eksekusi aktual tugas. Dari eksperimen, dapat disimpulkan bahwa komputasi paralel dapat meningkatkan kecepatan eksekusi program secara signifikan. Namun, peningkatan tersebut tidak selalu sebanding dengan jumlah core karena adanya overhead dan bagian program yang tetap berjalan secara serial. Konsep Amdahl's Law membantu memprediksi batas percepatan yang mungkin dicapai, sedangkan speedup dan efficiency memberikan gambaran nyata tentang performa sistem paralel.



Gambar 3.