PERANCANGAN SOLUSI
IMPLEMENTASI DAN ANALISIS KINERJA WORD COUNT PARALEL PADA DOKUMEN TEKS BESAR MENGGUNAKAN OPENMP

MATA KULIAH KOMPUTASI PARALEL DAN TERDISTRIBUSI

Dosen Pengampu : Firdhaus Hari S A H, ST., M.Eng.

Disusun oleh :

Adinda Putri Nur Rhoqimah	**2023061022**
Hana Fithri Sabiila	**2023061023**
Xyla Syarifatuzzahra A	**2024062006**

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS TEKNOLOGI DAN KESEHATAN

UNIVERSITAS SAHID SURAKARTA

TAHUN AJARAN 2025/2026

A. Analisis Sequential Baseline

1. Implementasi Solusi serial sebagai baseline

   Solusi baseline pada proyek ini adalah algoritma Word Count sekuensial yang diimplementasikan menggunakan bahasa pemrograman C tanpa mekanisme paralel. Algoritma ini membaca file teks karakter demi karakter dari awal hingga akhir, kemudian menghitung jumlah kata berdasarkan transisi dari karakter *whitespace* (spasi, tab, atau baris baru) ke karakter non-whitespace.

   Pendekatan sekuensial dipilih sebagai baseline karena bersifat sederhana, mudah dipahami, serta umum digunakan pada pengolahan teks berskala kecil. Hasil dan waktu eksekusi dari implementasi ini menjadi acuan utama dalam evaluasi kinerja algoritma paralel.

   Langkah langkah utama algoritma sekuensial :

   1) Membuka file teks input.
   2) Membaca karakter satu per satu.
   3) Mendeteksi awal kata berdasarkan perubahan karakter whitespace ke non-whitespace.
   4) Menambah penghitung kata setiap kali kata baru terdeteksi.
   5) Menampilkan total jumlah kata.

2. Analisis Kompleksitas Komputasi

   Algoritma word count sekuensial memiliki kompleksitas waktu O(n), dengan *n* adalah jumlah karakter dalam dokumen teks. Setiap karakter diproses tepat satu kali, sehingga waktu eksekusi meningkat secara linear seiring bertambahnya ukuran file.

   Kompleksitas ruang dari algoritma ini relatif kecil, yaitu O(1), karena tidak memerlukan struktur data tambahan yang besar. Namun, keterbatasan utama pendekatan sekuensial adalah tidak mampu memanfaatkan kemampuan multi-core prosesor, sehingga kurang efisien untuk file teks berukuran besar.

B. Desain Paralel

1. Pemilihan Model Pemrograman Paralel

   Model pemrograman paralel yang digunakan dalam proyek ini adalah **OpenMP (Open Multi-Processing)**. OpenMP dipilih karena:

   a) Mendukung arsitektur *shared memory* yang umum pada komputer modern.

b) Mudah diimplementasikan pada bahasa C.
c) Cocok untuk paralelisme berbasis data (*data parallelism*).
d) Overhead komunikasi relatif rendah dibandingkan model terdistribusi.

Model MPI tidak dipilih karena proyek dijalankan pada satu mesin (single node), sedangkan model hybrid belum diperlukan untuk kompleksitas masalah word count.

2. Identifikasi bagian yang dapat diparalelkan

   Bagian utama yang dapat diparalelkan adalah proses **penghitungan kata pada data teks**. Setelah file teks dimuat ke dalam buffer memori, data dapat dibagi ke beberapa bagian dan diproses secara paralel oleh beberapa thread.

   Bagian yang diparalelkan:

   1) Iterasi karakter pada buffer teks.
   2) Penghitungan jumlah kata pada setiap segmen data.

   Bagian yang tidak diparalelkan :

   1) Proses pembacaan file dari disk ke memori.
   2) Penggabungan hasil akhir (ditangani oleh mekanisme *reduction* OpenMP).

3. Desain Algoritma Paralel

   Algoritma paralel dirancang menggunakan pendekatan *data parallelism*, di mana buffer teks dibagi ke beberapa bagian berdasarkan indeks karakter.

   Pseudocode Algoritma Paralel Word Count (OpenMP)

   Baca seluruh file teks ke dalam buffer
   Inisialisasi total_word = 0

   #pragma omp parallel for reduction(+:total_word)
   for i = 0 sampai panjang_buffer-1:
   if buffer[i] bukan whitespace dan
   (i == 0 atau buffer[i-1] adalah whitespace):
   total_word = total_word + 1

   Tampilkan total_word

Pendekatan ini memastikan bahwa setiap thread menghitung jumlah kata pada bagian data yang berbeda, sementara mekanisme reduction mencegah terjadinya race condition.

4. Data Partitioning dan Load Balancing

   Data partitioning dilakukan secara otomatis oleh OpenMP melalui direktif parallel for, di mana indeks iterasi dibagi ke beberapa thread. Skema pembagian data bersifat static, sehingga setiap thread memperoleh jumlah data yang relatif sama.

   Karena setiap karakter memiliki beban komputasi yang seragam, maka beban kerja antar thread cenderung seimbang (*load balanced*). Pendekatan ini efektif untuk word count karena tidak terdapat perbedaan signifikan dalam kompleksitas pemrosesan tiap bagian data.

C. Rencana Pembagian Tugas

1. Pembagian Modul implementasi

   Pengembangan proyek dibagi ke dalam beberapa modul sebagai berikut :

   1) Modul Analisis Masalah

      Mengkaji permasalahan word count dan menentukan pendekatan solusi.

   2) Modul Implementasi Serial

      Mengembangkan program word count sekuensial sebagai baseline.

   3) Modul Implementasi Paralel

      Mengembangkan versi paralel menggunakan OpenMP.

   4) Modul Pengujian dan Evaluasi

      Melakukan pengujian kinerja dan pengumpulan data waktu eksekusi.

   5) Modul Dokumentasi

      Menyusun laporan dan analisis hasil.

2. Timeline Internal

   Rencana waktu pengerjaan proyek dirancang sebagai berikut:

   Minggu 1 : Analisis Masalah dan Studi Literatur

   Minggu 2 : Implementasi algoritma sekuensial.

   Minggu 3 : Implementasi algoritma paralel menggunakan OpenMP.

   Minggu 4 : Pengujian performa dan pengumpulan data.

   Minggu 5 : Analisis hasil dan penyusunan laporan akhir.