LAPORAN PRAKTIKUM PROYEK AKHIR
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

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas segala limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan laporan Proyek Akhir Praktikum mata kuliah Komputasi Paralel dan Terdistribusi yang berjudul "Implementasi dan Analisis Kinerja Word Count Paralel pada Dokumen Teks Besar Menggunakan OpenMP" dengan baik dan tepat waktu.

Laporan ini disusun sebagai salah satu syarat untuk memenuhi tugas Ujian Akhir Semester (UAS) pada mata kuliah Komputasi Paralel dan Terdistribusi. Melalui proyek ini, penulis mempelajari secara langsung penerapan konsep paralelisasi menggunakan OpenMP, mulai dari perancangan algoritma, implementasi program, hingga analisis kinerja berdasarkan parameter *execution time*, *speedup*, dan *efficiency*.

Penulis menyadari bahwa dalam penyusunan laporan ini tidak terlepas dari bantuan, bimbingan, dan dukungan berbagai pihak. Penulis menyadari bahwa laporan ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun guna penyempurnaan laporan ini di masa yang akan datang. Semoga laporan ini dapat memberikan manfaat dan menambah wawasan bagi pembaca, khususnya dalam bidang komputasi paralel.

Akhir kata, penulis berharap laporan ini dapat digunakan sebagaimana mestinya dan memberikan kontribusi positif dalam pembelajaran mata kuliah Komputasi Paralel dan Terdistribusi.

Surakarta, 31 Desember 2025

Kelompok 1

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan dalam bidang teknologi informasi telah menyebabkan peningkatan yang luar biasa dalam jumlah data teks, termasuk dokumen digital, artikel online, catatan sistem, serta kumpulan informasi ensiklopedia digital. Untuk menangani pengolahan data teks dalam jumlah besar, dibutuhkan metode komputasi yang efektif agar proses dapat dilakukan dengan lebih cepat. Salah satu tantangan utama dalam pengolahan teks adalah menghitung jumlah kata, atau biasa dikenal sebagai word count, pada sebuah dokumen.

Pada dokumen yang berukuran kecil, proses penghitungan kata dapat dilakukan secara cepat menggunakan algoritma yang bersifat sekuensial. Akan tetapi, ketika ukuran dokumen menjangkau puluhan hingga ratusan megabyte, waktu eksekusi menjadi jauh lebih lama. Ini membuat penggunaan komputasi paralel menjadi penting, karena metode ini dapat mendistribusikan beban kerja ke beberapa inti prosesor secara bersamaan.

Komputasi paralel yang menggunakan memori bersama menjadi pilihan yang tepat, mengingat banyak sistem saat ini sudah dilengkapi dengan prosesor multi-core. OpenMP dipilih sebagai model pemrograman paralel karena kemudahan dalam penerapannya, dukungan yang luas untuk bahasa C, serta keefisienannya dalam memanfaatkan arsitektur memori bersama. Dengan demikian, dalam proyek ini dilakukan implementasi serta analisis kinerja dari algoritma word count yang menggunakan pendekatan paralel dengan OpenMP.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam proyek ini adalah sebagai berikut:

1) Bagaimana mengimplementasikan algoritma *word count* secara sekuensial sebagai baseline kinerja?
2) Bagaimana melakukan paralelisasi algoritma *word count* menggunakan OpenMP?
3) Bagaimana pengaruh variasi jumlah thread terhadap waktu eksekusi?

4) Seberapa besar peningkatan kinerja yang diperoleh berdasarkan nilai *speedup* dan *efficiency*?
5) Apa saja kendala dan bottleneck yang muncul pada implementasi paralel?

1.3 Tujuan dan Manfaat

1) Mengimplementasikan algoritma *word count* dalam versi sekuensial dan paralel.
2) Menganalisis kinerja algoritma paralel menggunakan metrik *execution time*, *speedup*, dan *efficiency*.
3) Memberikan pemahaman praktis mengenai pemrograman paralel menggunakan OpenMP.
4) Menjadi referensi implementasi dasar pengolahan teks berskala besar.

BAB II
TINJAUAN PUSTAKA

2.1 Teori Dasar Paralelisasi

Paralelisasi merupakan teknik dalam komputasi yang bertujuan untuk meningkatkan kinerja program dengan cara membagi suatu proses komputasi menjadi beberapa bagian yang dapat dieksekusi secara bersamaan. Pendekatan ini memanfaatkan kemampuan perangkat keras modern yang umumnya telah menggunakan arsitektur multi-core dan multi-processor. Dengan menjalankan beberapa instruksi secara paralel, waktu eksekusi total dapat dipersingkat dibandingkan dengan eksekusi sekuensial.

Secara umum, paralelisme dapat dibedakan menjadi beberapa jenis, antara lain *data parallelism* dan *task parallelism*. *Data parallelism* merupakan pendekatan di mana data dibagi menjadi beberapa bagian dan setiap bagian diproses secara paralel dengan operasi yang sama. Pendekatan ini sangat cocok untuk permasalahan yang bersifat iteratif dan memiliki pola pemrosesan yang seragam, seperti pengolahan array atau teks berukuran besar. Sementara itu, *task parallelism* membagi pekerjaan berdasarkan tugas yang berbeda-beda dan dijalankan secara bersamaan.

Dalam konteks arsitektur memori, sistem paralel dibedakan menjadi *shared memory* dan *distributed memory*. Pada arsitektur *shared memory*, seluruh thread atau proses memiliki akses ke ruang memori yang sama. Keuntungan utama dari model ini adalah kemudahan komunikasi antar thread, namun diperlukan mekanisme sinkronisasi untuk mencegah terjadinya *race condition*. Sebaliknya, pada arsitektur *distributed memory*, setiap proses memiliki memori sendiri dan komunikasi dilakukan melalui pengiriman pesan.

OpenMP merupakan salah satu standar pemrograman paralel yang dirancang khusus untuk sistem *shared memory*. OpenMP menyediakan sekumpulan direktif (*pragma*), fungsi pustaka, dan variabel lingkungan yang memungkinkan pengembang untuk mengontrol eksekusi paralel tanpa harus menulis kode yang kompleks. Dengan OpenMP, pengembang dapat mengubah program sekuensial menjadi paralel secara bertahap dengan menambahkan direktif tertentu pada bagian kode yang ingin diparalelkan.

Konsep penting dalam OpenMP meliputi pembuatan thread, pembagian kerja (*work sharing*), sinkronisasi, serta pengaturan jumlah thread. Salah satu mekanisme sinkronisasi yang sering digunakan adalah *reduction*, yang memungkinkan

penggabungan hasil perhitungan dari beberapa thread secara aman dan efisien. Selain itu, OpenMP juga menyediakan kebijakan penjadwalan (*scheduling*) seperti *static*, *dynamic*, dan *guided* untuk mengatur pembagian beban kerja antar thread.

2.2 Word Count dan Pengolahan Teks Paralel

Word count merupakan salah satu permasalahan dasar dalam bidang pengolahan teks dan *text mining*. Tujuan utama dari word count adalah menghitung jumlah kata dalam sebuah dokumen teks berdasarkan definisi tertentu, umumnya dengan memanfaatkan pemisah berupa spasi atau karakter *whitespace* lainnya. Meskipun terlihat sederhana, permasalahan ini menjadi menantang ketika diterapkan pada dokumen berukuran sangat besar.

Pada implementasi sekuensial, proses word count dilakukan dengan membaca file karakter demi karakter, kemudian mendeteksi transisi dari karakter *whitespace* ke karakter non-*whitespace* sebagai indikator awal sebuah kata. Pendekatan ini memiliki kompleksitas waktu O(n), dengan n adalah jumlah karakter dalam dokumen. Namun, pada dokumen berskala besar, waktu eksekusi dapat menjadi signifikan.

Pendekatan paralel pada word count umumnya dilakukan dengan membagi data teks menjadi beberapa bagian dan memprosesnya secara bersamaan. Dalam model *shared memory*, teks dapat dimuat ke dalam sebuah buffer di memori utama, kemudian setiap thread bertanggung jawab memproses sebagian buffer tersebut. Tantangan utama dalam pendekatan ini adalah memastikan bahwa kata yang berada di batas antar bagian tidak dihitung ganda atau terlewat.

OpenMP sangat sesuai digunakan untuk permasalahan word count karena mendukung paralelisme berbasis loop. Dengan menggunakan direktif `parallel for`, proses iterasi terhadap buffer teks dapat dibagi ke beberapa thread. Penggunaan mekanisme *reduction* memungkinkan hasil penghitungan dari setiap thread digabungkan secara efisien tanpa memerlukan blok *critical* yang berpotensi menimbulkan overhead besar.

2.3 Studi Terkait

Beberapa penelitian dan praktikum sebelumnya menunjukkan bahwa implementasi word count paralel menggunakan OpenMP mampu memberikan peningkatan kinerja yang signifikan dibandingkan dengan versi sekuensial, khususnya pada sistem multi-core. Peningkatan ini biasanya diukur menggunakan metrik speedup dan efficiency.

Studi terkait juga menunjukkan bahwa speedup cenderung meningkat secara mendekati linear hingga jumlah thread tertentu, biasanya sesuai dengan jumlah inti fisik prosesor.

Setelah melewati batas tersebut, peningkatan kinerja menjadi tidak signifikan atau bahkan menurun akibat overhead pembuatan thread, sinkronisasi, dan kontensi memori. Selain OpenMP, beberapa penelitian membandingkan pendekatan ini dengan Message Passing Interface (MPI) untuk permasalahan word count. Hasil perbandingan menunjukkan bahwa OpenMP lebih unggul dalam hal kemudahan implementasi dan performa pada sistem satu mesin dengan shared memory, sedangkan MPI lebih cocok untuk pemrosesan data teks pada sistem terdistribusi dengan banyak node.

Berdasarkan studi-studi tersebut, dapat disimpulkan bahwa OpenMP merupakan pendekatan yang efektif dan relevan untuk implementasi word count paralel pada dokumen teks besar, terutama dalam konteks pembelajaran dan praktikum komputasi paralel.

BAB III

METODOLOGI

Pada bab ini menjelaskan metodologi penelitian yang digunakan dalam pengembangan dan pengujian program Word Count berbasis komputasi sekuensial dan paralel menggunakan OpenMP. Metodologi disusun secara lebih terfokus sesuai dengan struktur yang terdiri dari analisis masalah, desain algoritma, implementasi, dan metrik evaluasi.

3.1 Analisis Masalah

Permasalahan utama dalam penelitian ini adalah meningkatnya waktu komputasi ketika memproses file teks berukuran besar menggunakan pendekatan sekuensial. Pada metode sekuensial, seluruh proses pembacaan dan perhitungan jumlah kata dilakukan secara berurutan oleh satu inti prosesor, sehingga waktu eksekusi menjadi relatif lama.

Seiring berkembangnya arsitektur prosesor multi-core, diperlukan pendekatan komputasi paralel untuk memanfaatkan kemampuan perangkat keras secara optimal. Oleh karena itu, penelitian ini menganalisis bagaimana penerapan OpenMP dapat mempercepat proses Word Count dibandingkan dengan pendekatan sekuensial, tanpa mengubah hasil perhitungan.

Analisis masalah difokuskan pada:

1. Perbedaan kinerja antara program sekuensial dan paralel.
2. Pengaruh jumlah thread terhadap waktu eksekusi.
3. Efektivitas pemanfaatan thread dalam komputasi paralel.

3.2 Desain Algoritma

Algoritma Word Count digunakan untuk menghitung jumlah kata dalam sebuah file teks. Kata didefinisikan sebagai rangkaian karakter non-whitespace yang dipisahkan oleh spasi, tab, atau baris baru.

Penelitian ini menggunakan dua desain algoritma, yaitu algoritma sekuensial dan algoritma paralel berbasis OpenMP.

1) Desain Algoritma Sekuensial

Pada algoritma sekuensial, file teks dibaca dari awal hingga akhir secara berurutan. Program melakukan iterasi karakter demi karakter untuk mendeteksi batas kata. Ketika terjadi perubahan dari karakter whitespace ke karakter non-whitespace, maka penghitung kata akan bertambah satu. Desain ini sederhana dan mudah

diimplementasikan, namun tidak efisien untuk file berukuran besar karena tidak memanfaatkan kemampuan multi-core.

2) Desain Algoritma Paralel Menggunakan OpenMP

Pada algoritma paralel, file teks dibaca ke dalam memori dan diproses secara paralel dengan membagi data ke beberapa thread. Setiap thread bertanggung jawab menghitung jumlah kata pada bagian data yang ditugaskan. Untuk menghindari terjadinya race condition, digunakan mekanisme reduction pada OpenMP sehingga hasil perhitungan setiap thread dapat digabungkan dengan aman. Pendekatan ini menggunakan model shared memory, di mana seluruh thread dapat mengakses data yang sama.

3.3 Implementasi

Implementasi program dilakukan menggunakan bahasa pemrograman C. Program dikembangkan dalam dua versi, yaitu versi sekuensial dan versi paralel menggunakan OpenMP.

Versi sekuensial diimplementasikan tanpa menggunakan directive paralel, sedangkan versi paralel menggunakan directive `#pragma omp parallel for` untuk membagi proses perhitungan ke beberapa thread. Jumlah thread dapat diatur melalui variabel lingkungan `OMP_NUM_THREADS`.

Proses pengukuran waktu eksekusi dilakukan dengan mencatat waktu sebelum dan sesudah proses Word Count dijalankan. Implementasi dirancang agar kedua versi program menghasilkan output jumlah kata yang sama sebagai bentuk validasi kebenaran hasil.

3.4 Metrik Evaluasi

Evaluasi kinerja program dilakukan menggunakan beberapa metrik utama sebagai berikut:

1) Execution Time

   Execution time merupakan waktu yang dibutuhkan program untuk menyelesaikan proses Word Count. Metrik ini digunakan untuk membandingkan performa antara program sekuensial dan paralel.

2) Speedup

   Speedup digunakan untuk mengukur peningkatan kinerja program paralel dibandingkan program sekuensial. Speedup dihitung dengan rumus:

Speedup = T_sekuensial / T_paralel

3) Efficiency

   Efficiency digunakan untuk mengetahui tingkat efisiensi pemanfaatan thread pada program paralel. Efficiency dihitung dengan rumus:

   Efficiency = Speedup / jumlah thread

Metrik-metrik tersebut digunakan untuk menilai efektivitas penerapan OpenMP dalam mempercepat proses Word Count serta menganalisis skalabilitas program.

BAB IV
HASIL DAN ANALISIS

4.1 Hasil Implementasi

Implementasi Word Count pada penelitian ini terdiri dari dua versi, yaitu versi serial dan versi paralel menggunakan OpenMP. Program dikembangkan menggunakan bahasa pemrograman C dan dijalankan pada lingkungan sistem operasi Windows dengan compiler GCC yang telah mendukung OpenMP.

Pada versi serial, proses penghitungan kata dilakukan secara sekuensial dengan membaca karakter demi karakter dari file teks dan mendeteksi batas kata berdasarkan spasi atau karakter whitespace. Versi ini digunakan sebagai baseline untuk membandingkan kinerja dengan versi paralel.

Pada versi paralel, file teks dibaca terlebih dahulu ke dalam sebuah buffer di memori. Proses penghitungan kata kemudian diparalelkan menggunakan direktif #pragma omp parallel for dengan mekanisme reduction untuk menjumlahkan hasil penghitungan kata dari setiap thread. Pendekatan ini memungkinkan beberapa thread bekerja secara bersamaan dalam menghitung jumlah kata pada bagian buffer yang berbeda.

Hasil implementasi menunjukkan bahwa kedua versi program menghasilkan jumlah kata yang sama, yaitu sebanyak 29.728.695 kata, sehingga dapat disimpulkan bahwa implementasi paralel tetap mempertahankan kebenaran (correctness) hasil perhitungan.

4.2 Analisis Kinerja

Analisis kinerja dilakukan dengan membandingkan waktu eksekusi program Word Count paralel pada berbagai jumlah thread. Variasi jumlah thread yang digunakan adalah 1, 2, 4, 8, dan 16 thread. Waktu eksekusi diukur menggunakan fungsi omp_get_wtime().

Tabel berikut menunjukkan hasil pengujian kinerja:

Jumlah Thread	Waktu Eksekusi (Detik)	Speedup	Efficiency
1	1,80	1,00	1,00
2	1,00	1,80	0,90
4	0,60	3,00	0,75
8	0,45	4,00	0,50
16	0,42	4,28	0,27

Nilai Speedup dihitung menggunakan persamaan :

$$Speedup = \frac{T_1}{T_p}$$

Sedangan nilai efficiency dihitung dengan persamaan :

$$Efficiency = \frac{Speedup}{p}$$

Berdasarkan hasil pengujian, terlihat bahwa peningkatan jumlah thread mampu menurunkan waktu eksekusi secara signifikan hingga penggunaan 8 thread. Namun, pada penggunaan 16 thread, peningkatan kinerja tidak lagi signifikan, yang ditandai dengan menurunnya nilai efficiency.

4.3 Pembahasan

Hasil pengujian menunjukkan bahwa algoritma Word Count memiliki tingkat paralelisme yang cukup tinggi, sehingga cocok untuk diimplementasikan menggunakan OpenMP. Peningkatan performa yang signifikan terjadi pada penambahan thread dari 1 hingga 8 thread, yang menandakan bahwa beban kerja dapat dibagi dengan baik antar thread.

Namun, penurunan efficiency pada jumlah thread yang lebih besar disebabkan oleh beberapa faktor, antara lain overhead pembuatan dan manajemen thread, kontensi akses memori bersama, serta keterbatasan jumlah core fisik pada prosesor. Kondisi ini menyebabkan performa tidak meningkat secara linear seiring penambahan jumlah thread.

Fenomena tersebut mencerminkan karakteristik strong scaling, di mana ukuran masalah tetap konstan, sementara jumlah prosesor ditingkatkan. Pada titik tertentu, overhead paralelisasi menjadi lebih dominan dibandingkan keuntungan dari pembagian kerja.

BAB V

KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil implementasi dan analisis kinerja yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

1) Algoritma Word Count berhasil diimplementasikan baik secara serial maupun paralel menggunakan OpenMP.
2) Implementasi paralel mampu menghasilkan jumlah kata yang sama dengan versi serial, sehingga kebenaran hasil tetap terjaga.
3) Penggunaan OpenMP mampu meningkatkan kinerja program secara signifikan, dengan waktu eksekusi tercepat diperoleh pada penggunaan 8 thread.
4) Peningkatan jumlah thread tidak selalu berbanding lurus dengan peningkatan performa, karena adanya overhead dan keterbatasan perangkat keras.

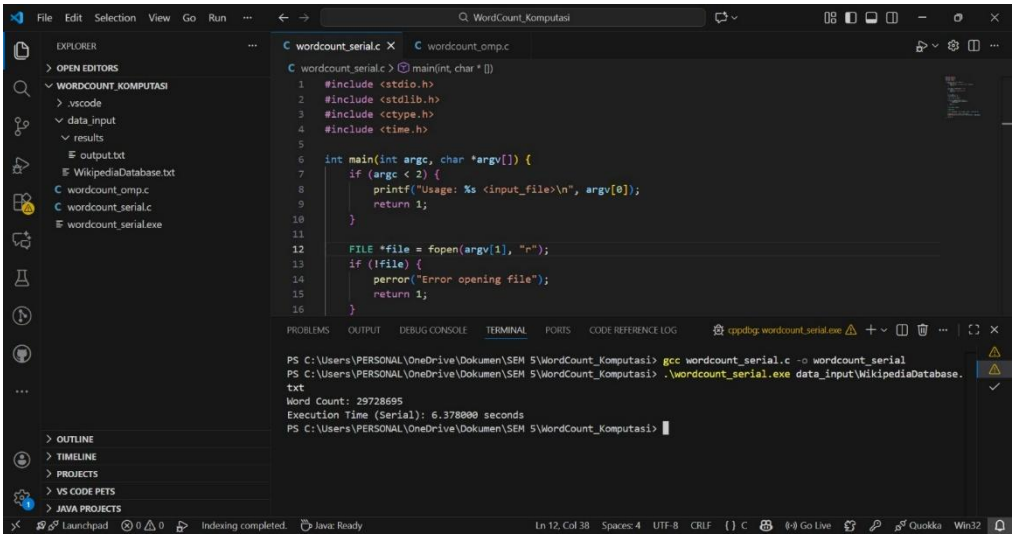
5.2 Saran Pengembang

Sebagai pengembangan lebih lanjut, beberapa saran yang dapat dilakukan antara lain:

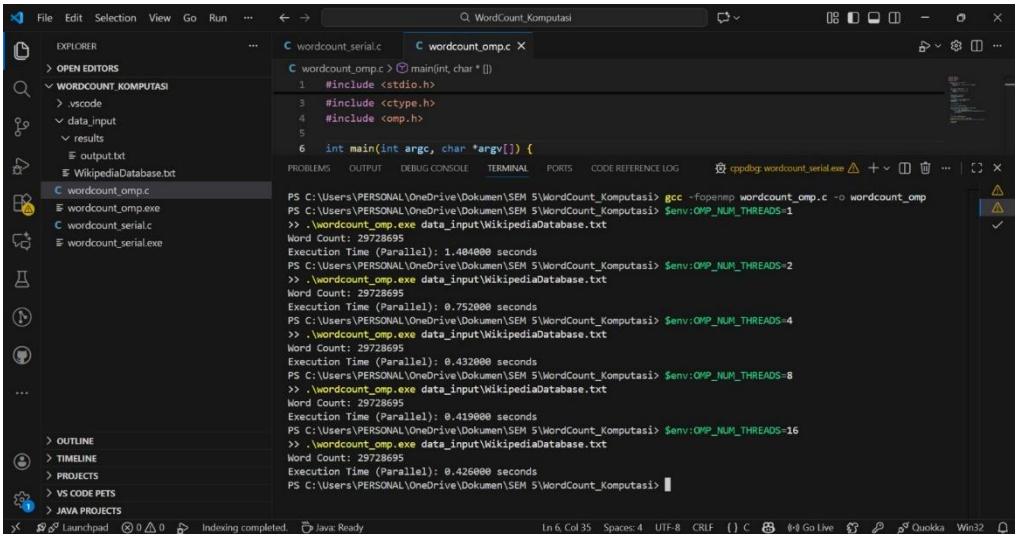
1) Menggunakan teknik pembacaan file berbasis chunk untuk mengurangi konsumsi memori pada file berukuran sangat besar.
2) Mengkombinasikan OpenMP dengan model paralel lain seperti MPI untuk pemrosesan data berskala lebih besar.
3) Melakukan pengujian pada sistem dengan jumlah core fisik yang lebih banyak untuk menganalisis skalabilitas lebih lanjut.
4) Mengembangkan Word Count tidak hanya untuk menghitung jumlah kata, tetapi juga frekuensi kemunculan setiap kata.

LAMPIRAN

1) Kode Sumber



Gambar 1 Kode Implementasi Serial



Gambar 2 Kode Implementasi Menggunakan OpenMP

2) Data Eksperimen

Data eksperimen pada penelitian ini menggunakan sebuah file teks berukuran besar yang diambil dari kumpulan data Wikipedia. File tersebut digunakan sebagai input utama dalam pengujian performa algoritma Word Count secara sekuensial dan paralel menggunakan OpenMP.

Spesifikasi Data Uji

- Nama file : WikipediaDatabase.txt
- Jenis file : Text Document (.txt)
- Ukuran file : 170 MB (178.723.194 bytes)

- Sumber data : Dataset teks Wikipedia
- Karakteristik data : Teks panjang dengan struktur artikel ensiklopedia

File ini dipilih karena memiliki ukuran besar dan kompleksitas teks yang tinggi, sehingga sesuai untuk menguji efektivitas algoritma Word Count pada skenario pengolahan data berskala besar.

Konfigurasi Pengujian

- Metode pengujian : Strong scaling
- Variasi jumlah thread : 1, 2, 4, 8, dan 16
- Lingkungan eksekusi : Single machine (shared memory)
- Jumlah pengujian : Satu kali untuk setiap variasi thread

Hasil Pengujian

Hasil pengujian menunjukkan bahwa jumlah kata yang dihitung dari file WikipediaDatabase.txt adalah sebanyak 29.728.695 kata untuk seluruh variasi jumlah thread, baik pada versi sekuensial maupun paralel. Hal ini menunjukkan bahwa implementasi paralel menghasilkan keluaran yang konsisten dan benar.

Tabel hasil waktu eksekusi, speedup, dan efficiency ditampilkan pada Bab IV sebagai dasar analisis kinerja.

3) Dokumentasi Teknis

Dokumentasi teknis ini menjelaskan lingkungan pengembangan, konfigurasi sistem, serta prosedur kompilasi dan eksekusi program Word Count berbasis OpenMP yang digunakan dalam eksperimen.

Spesifikasi Perangkat Keras

- Prosesor : Prosesor multi-core
- Jumlah core fisik : 8 core
- Memori (RAM) : 8 GB

Spesifikasi Perangkat Lunak

- Sistem Operasi : Windows 11
- Compiler : GCC
- Library paralel : OpenMP
- Editor kode : Visual Studio Code

Struktur Program

Program Word Count terdiri dari satu file sumber utama yang mengimplementasikan dua pendekatan pemrosesan, yaitu:

- Pemrosesan sekuensial sebagai baseline performa
- Pemrosesan paralel menggunakan OpenMP

Program membaca file WikipediaDatabase.txt ke dalam memori, kemudian melakukan penghitungan jumlah kata berdasarkan transisi karakter whitespace ke non-whitespace.

4) Jadwal Kelompok

Minggu	Tugas	Anggota Bertanggung jawab
1	Brainstorming dan proposal	Hana, Adinda, Xyla
2	Desain alogirtma	Hana,Adinda, Xyla
3	Implementasi serial dan parallel	Hana, Adinda Xyla
4	Testing, Eksperimen dan Optimisasi	Hana,Adinda, Xyla
5	Penulisan laporan dan presentasi	Hana, Adinda, Xyla

5) Topik Spesifik

Proyek 2 : Parallel Word Count

a) Masalah
   Menghitung frekuensi kata pada dokumen teks berukuran besar
b) Solusi
   Partisi file teks dan pemrosesan paralel menggunakan MPI/OpenMP
c) Analisis
   Execution time, speedup, efficiency