Journal of Computer and Information System (J-CIS)
Vol.6, No.2, Oktober 2023, Page 1-8
ISSN (print): 2622-5859, ISSN (online): 2622-0881
DOI: https://doi.org/10.31605/jcis.v6i2

Optimasi Pencarian Titik Pada Gambar Menggunakan *Parallel Computing*

Wawan Firgiawan*[1], Nur Halisah[2], Gibran Ghadavi[3], Muhammad Aditia[4], M. Darwin[5], Ulfa Maulidia[6]
[1,2,3,4,5,6]Program Studi Teknik Informatika, Universitas Sulawesi Barat
E-mail:[1]wawanfirgiawan@unsulbar.ac.id, [2]halisahica435@gmail.com, [3]gibranghadavi@gmail.com, [4]muhammadaditia237@gmail.com, [5]darwinpolman58@gmail.com [6]ulfamaulidia11@gmail.com

Abstrak

Penelitian ini membahas perbandingan antara program serial dan program paralel dalam menyelesaikan permasalahan Penjumlahan Titik. Dalam tahapannya, penjumlahan titik ini dapat diperoleh dengan cepat ketika dalam pencarian setiap titiknya menerapkan pencarian paralel. Dalam program paralel yang digunakan dalam penelitian ini adalah OpenMP. Fokus penelitian adalah pada pengoptimalan waktu eksekusi Penjumlahan Titik menggunakan pemrograman OpenMP dan perbandingannya dengan program serial. Metode penelitian ini melibatkan implementasi program serial dengan OpenMP dan pengukuran waktu eksekusi keduanya dalam menyelesaikan Penjumlahan Titik. Hasil penelitian menunjukkan bahwa pemanfaatan program paralel, khususnya OpenMP, mampu melakukan penjumlahan titik dengan waktu yang lebih efisien dibandingkan dengan penggunaan program serial. Kesimpulan dari penelitian ini adalah bahwa pemanfaatan pemrograman OpenMP berhasil mengoptimalkan waktu eksekusi Penjumlahan Titik. Program paralel dengan OpenMP terbukti lebih efektif dalam menyelesaikan Penjumlahan Titik dibandingkan dengan program serial. Implementasi program paralel memanfaatkan penugasan eksekusi program secara bersamaan, menghasilkan waktu eksekusi yang lebih efisien dan mengkonfirmasi keefektifan program paralel dalam konteks Penjumlahan Titik.

Kata kunci *– Optimasi, OpenMP, Penjumlahan Titik*

Abstract

This research discusses the comparison between serial programs and parallel programs in solving Point Addition problems. In this stage, the sum of these points can be obtained quickly when searching for each point using parallel search. The parallel program used in this research is OpenMP. The focus of the research is on the execution time mortality of Point Addition using OpenMP programming and its comparison with serial programming. This research method includes implementing a serial program with OpenMP and measuring the execution time of both in completing Point Addition. The research results show that the use of parallel programs, especially OpenMP, is able to perform Point Addition in a more efficient time compared to the use of serial programs. The conclusion of this research is that the use of OpenMP programming has succeeded in optimizing the execution time of Point Addition.

Parallel programs with OpenMP are proven to be more effective in completing Point Addition compared to serial programs. The implementation of parallel programs takes advantage of the assignment of concurrent program execution, resulting in more efficient execution times and confirming the effectiveness of parallel programs in the context of Point Addition.

Keywords *– Optimization, OpenMP, Point Addition*

1. PENDAHULUAN

Penjumlahan titik *(dot product)* adalah operasi aritmatika umum yang sering digunakan dalam berbagai bidang ilmu, termasuk matematika, fisika, dan komputasi. Operasi ini melibatkan perkalian elemen dari dua array dan penjumlahan hasil perkalian tersebut. Dalam konteks komputasi, penjumlahan titik dapat dilakukan baik secara serial maupun paralel. Penjumlahan titik secara serial dilakukan dengan cara mengulangi operasi perkalian dan penjumlahan secara berurutan. Sementara itu, penjumlahan titik secara paralel dilakukan dengan membagi array menjadi beberapa bagian dan menjalankan operasi perkalian dan penjumlahan secara bersamaan pada setiap bagian[1].

Komputasi paralel dan sistem terdistribusi memberikan berbagai keuntungan untuk penjumlahan titik, seperti peningkatan kinerja dan skalabilitas[2]. Kinerja penjumlahan titik dapat ditingkatkan dengan membagi array menjadi bagian-bagian kecil dan menjalankan operasi perkalian dan penjumlahan secara simultan pada setiap bagian. Skalabilitas penjumlahan titik dapat ditingkatkan dengan menambahkan lebih banyak sumber daya komputasi, seperti CPU *(Central Processing Unit)*, GPU *(Graphics Processing Unit)*, atau TPU *(Tensor Processing Unit)*[3].

Berdasarkan hal tersebut, penelitian ini bertujuan untuk mendapatkan waktu eksekusi yang efisien pada permasalahan penjumlahan titik serta membandingkan efektivitas dengan penggunaan pogram serial.

2. METODE

2.1 Pengenalan Penjumlahan Titik

Penjumlahan titik adalah operasi matematika yang dilakukan dengan cara menjumlahkan dua atau lebih titik pada sebuah koordinat. Dalam matematika, titik dapat direpresentasikan oleh koordinat (x,y) pada bidang kartesius. Penjumlahan titik dapat dilakukan dengan menjumlahkan koordinat x dan y dari masing-masing titik.

Penjumlahan titik dalam komputasi paralel adalah proses penjumlahan elemen-elemen dari dua array atau matriks secara paralel. Komputasi paralel memanfaatkan banyak sumber daya komputasi secara bersamaan untuk meningkatkan kecepatan dan efisiensi pemrosesan data [4]. Dalam penjumlahan titik komputasi paralel, elemen-elemen dari array atau matriks yang akan dijumlahkan dibagi menjadi bagian-bagian yang lebih kecil. Setiap bagian akan diproses secara paralel oleh beberapa unit pemrosesan yang bekerja secara bersamaan. Hasil dari setiap proses paralel kemudian dijumlahkan untuk menghasilkan hasil akhir penjumlahan titik

Penerapan komputasi paralel dalam penjumlahan titik memungkinkan pemrosesan data yang lebih cepat dan efisien, terutama ketika bekerja dengan himpunan data yang besar. Dengan memanfaatkan banyak sumber daya komputasi secara bersamaan, komputasi paralel dapat meningkatkan kinerja dan mengurangi waktu pemrosesan data.

2. 2. Permasalahan waktu eksekusi dalam penjumlahan titik

Pemrograman paralel pada penjumlahan titik menjadi imperatif ketika dihadapkan dengan skala komputasi yang besar, seperti pada dataset yang melibatkan jutaan atau bahkan milyaran titik. Dalam situasi ini, pendekatan sekuensial dapat menjadi sangat lambat dan tidak

praktis[5]. Selain itu, permasalahan lain muncul ketika kita berurusan dengan *Big Data*. Proses penjumlahan titik, yang merupakan operasi dasar dalam analisis data, dapat dipercepat secara signifikan dengan menerapkan pemrograman paralel[6]. Dengan meningkatnya popularitas *multi-core processors*, pemanfaatan penuh potensi sumber daya ini menjadi kunci. Pemrograman paralel memungkinkan distribusi pekerjaan penjumlahan titik secara efisien di antara inti-inti prosesor, memastikan optimalisasi penggunaan *multi-core processors*[7].

2. 3. Implementasi pemrograman serial dan pararel pada permasalahan penjumlahan titik

Pemrograman serial adalah teknik pemrograman yang hanya menggunakan satu prosesor untuk mengeksekusi instruksi-instruksi secara berurutan[8]. Pemrograman paralel adalah teknik pemrograman yang memanfaatkan lebih dari satu prosesor untuk mengeksekusi instruksi-instruksi secara bersamaan[9]. Tujuan dari pemrograman paralel adalah untuk meningkatkan kinerja dan efisiensi komputasi, terutama untuk permasalahan yang membutuhkan banyak data atau proses[7]. Salah satu pustaka pemrograman paralel yang populer adalah OpenMP, yang memungkinkan kita untuk membagi tugas ke beberapa bagian dan menjalankannya secara paralel dengan menggunakan direktif pragma[4].

Salah satu permasalahan yang dapat diselesaikan dengan pemrograman serial dan paralel adalah penjumlahan titik. Penjumlahan titik adalah operasi yang menghasilkan titik baru dari dua titik yang diberikan[9]. Misalnya, jika kita memiliki dua titik A(x1, y1, z1) dan B(x2, y2, z2), maka hasil penjumlahannya adalah C(x1+x2, y1+y2, z1+z2). Untuk melakukan penjumlahan titik secara serial, kita dapat menggunakan algoritma sederhana seperti berikut:

```
// Algoritma penjumlahan titik secara serial
// Input: dua titik A dan B
// Output: titik C yang merupakan hasil penjumlahan A dan B
C.x = A.x + B.x
C.y = A.y + B.y
C.z = A.z + B.z
return C
```

Untuk melakukan penjumlahan titik secara pararel dengan menggunakan OpenMP, kita dapat menggunakan direktif pragma untuk membagi tugas ke beberapa bagian dan menjalankannya secara paralel. Contoh algoritma penjumlahan titik secara pararel dengan menggunakan OpenMP adalah sebagai berikut:

```
// Algoritma penjumlahan titik secara pararel dengan OpenMP
// Input: dua titik A dan B
// Output: titik C yang merupakan hasil penjumlahan A dan B
#pragma omp parallel sections // membagi tugas ke beberapa bagian
{
  #pragma omp section // bagian pertama
  {
    C.x = A.x + B.x // menjumlahkan komponen x
  }
  #pragma omp section // bagian kedua
  {
    C.y = A.y + B.y // menjumlahkan komponen y
  }
```

```
  #pragma omp section // bagian ketiga
  {
    C.z = A.z + B.z // menjumlahkan komponen z
  }
}
return C
```

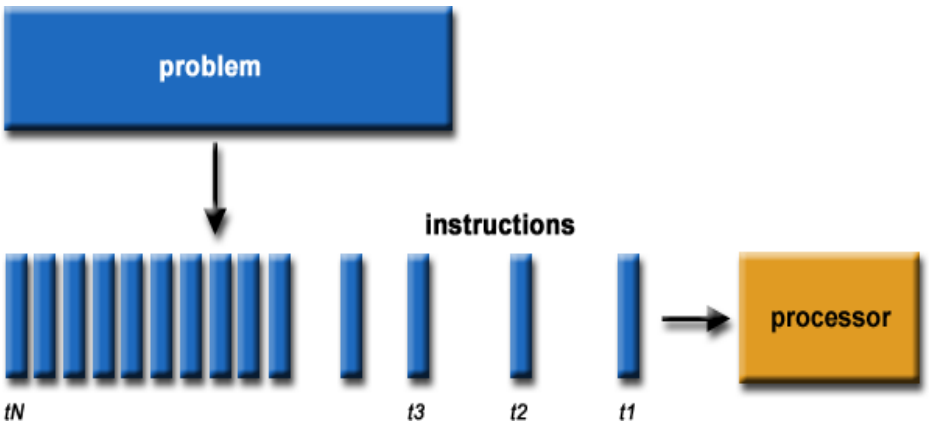
2. *4. Rancangan Program Serial dan Paralel Pada Penjumlahan Titik*

2.4.1. Rancangan Program Serial

Rancangan program serial pada program penjumlahan titik dapat dilihat pada pseudocode dibawah ini. Implementasi program serial dalam program penjumlahan titik ini digunakan pada iterasi bersarang kemudian dilakukan pencarian pada index iterasi pertama dan kedua. Dalam pencarian ini, digunakan kondisi apabila nilai yang didapatkan pada index pertama dan kedua adalah 1, maka nilai pada indeks tersebut akan dihitung sebagai titik dan akan ditambah kedalam suatu variabel untuk menyimpan nilai yang dibaca sebagai titik. Ketika semua index telah ditelusuri, maka selanjutnya adalah pengembalian jumlah titik pada variabel yang menampung jumlah titik. Program yang digunakan secara serial ini dapat dilihat pada setiap langkahnya dilakukan eksekusi satu persatu dari awal hingga akhir proses.

```
int hitungJmlTitikSerial() {
        int jmlTitik = 0;
        for (int i = 0; i < lokasiTitik.size(); i++){
            for (int j = 0; j < lokasiTitik[i].size(); j++){
                if (lokasiTitik[i][j] == 1) {
                    jmlTitik++;
                }
            }
        }
        return jmlTitik;
    }
```

Ilustrasi dari program yang dilakukan secara serial dapat dilihat pada gambar 1 dibawah ini.



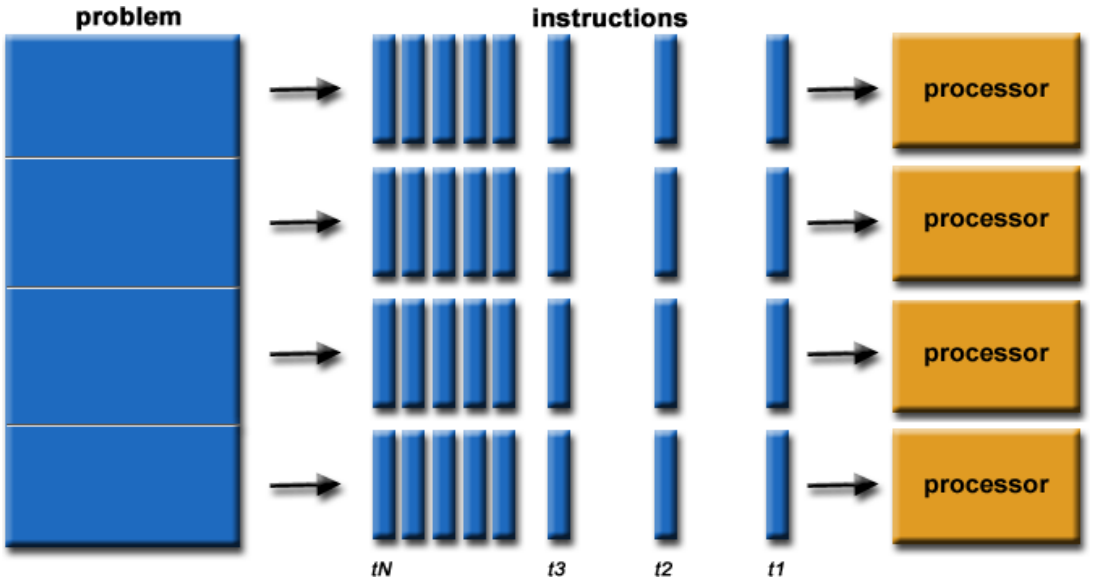
Gambar 1. Penyelesaian masalah dengan cara sekuensial.
(sumber : [6])

2.4.2. Implementasi Program Paralel

Rancangan program paralel pada program penjumlahan titik dapat dilihat pada pseudocode dibawah ini. Implementasi program paralel dalam penjumlahan titik ini menggunakan *"#pragma omp parallel for"* dengan *"collapse(2)"* serta *"reduction(+:jmlTitik)"*. Dengan menggunakan openMP memungkinkan iterasi bersarang akan dijalankan secara paralel menggunakan beberap thread. Penggunaan reduksi dilakukan untuk memastikan penjumlahan titiknya dilakukan secara aman pada lingkungan paralel. Setelah dilakukan perhitungan, maka selanjutnya adalah mengembalikan nilai hasil penjumlahan titiknya.

```
int hitungJmlTitikParalalel() {
      int jmlTitik = 0;
#pragma omp parallel for collapse(2) reduction(+:jmlTitik)
      for (int i = 0; i < lokasiTitik.size(); i++) {
         for (int j = 0; j < lokasiTitik[i].size(); j++) {
            if (lokasiTitik[i][j] == 1) {
               jmlTitik++;
            }
         }
      }
      return jmlTitik;
   }
```

Ilustrasi dari program yang dilakukan secara paralel dapat dilihat pada gambar 2 dibawah ini.



Gambar 2. Penyelesaian masalah secara paralel
(sumber : [6])

3. HASIL DAN PEMBAHASAN

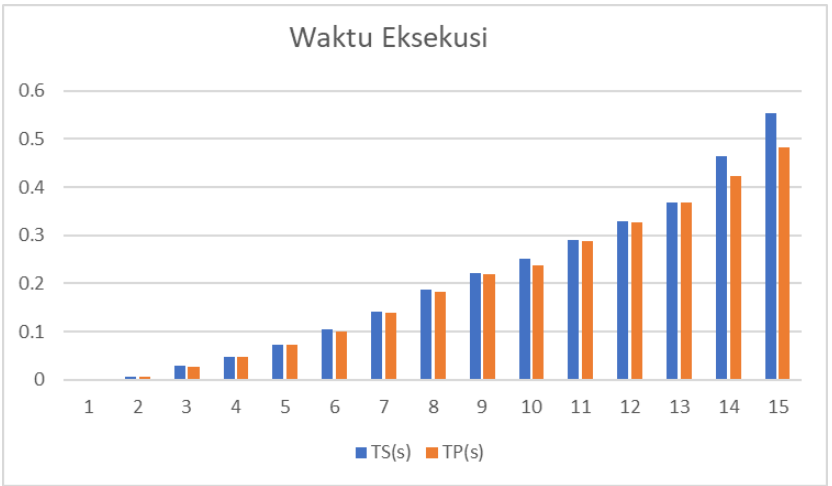
Dalam penelitian ini, perbandingan antara implementasi program secara serial dan paralel pada operasi penjumlahan titik dilakukan dengan tujuan untuk mencapai waktu eksekusi optimal. Fokus penelitian ini adalah pada pengujian efektivitas penerapan program paralel OpenMP dalam konteks penjumlahan titik. Pengujian dilakukan sebanyak 15 kali pada kedua jenis program (serial dan paralel) dengan variasi parameter luas bidang array yang berbeda. Setiap parameter diuji sebanyak 10 kali, dan rata-rata waktu eksekusi diambil dari algoritma yang berjalan secara serial (TS) dan algoritma yang berjalan paralel (TP).

Hasil eksekusi dengan waktu yang paling efisien dipilih sebagai hasil pengujian untuk setiap parameter. Detail hasil pengujian terlampir pada tabel di bawah ini.

Tabel 1. Hasil Perhitungan Waktu Komputasi pada Penjumlahan Titik

No.	Luas Array	TS	TP	*SpeedUp*
1	10 x 10	0.0023	0.0022	1.04545455
2	20 x 20	0.0077	0.0077	1
3	30 x 30	0.0284	0.0279	1.01792115
4	40 x 40	0.0481	0.048	1.00208333
5	50 x 50	0.0733	0.0731	1.00273598
6	60 x 60	0.1055	0.101	1.04455446
7	70 x 70	0.1421	0.1402	1.01355207
8	80 x 80	0.1877	0.1828	1.02680525
9	90 x 90	0.2212	0.2202	1.00454133
10	100 x 100	0.2513	0.2388	1.05234506
11	110 x 110	0.2893	0.2881	1.00416522
12	120 x 120	0.3303	0.3271	1.00978294
13	130 x 130	0.368	0.3678	1.00054377
14	140 x 140	0.465	0.4227	1.10007097
15	150 x 150	0.5534	0.4824	1.14718076

Berdasarkan hasil pengujian pada tabel 1, dapat dihasilkan grafik perbandingan penggunaan program serial dan paralel pada penjumlahan titik sebagai berikut.



Gambar 3. Grafik Perbandingan Waktu Eksekusi

Pada gambar 3 diatas, secara keseluruhan hasil pengujian menunjukkan waktu eksekusi yang terbaik dilakukan pada program OpenMP. Ini menunjukkan keefektifan program OpenMP dalam menyelesaikan permasalahan Penjumlahan Titik. Penggunaan OpenMP memungkinkan untuk melakukan pencarian dan penjumlahan secara paralel. Dalam Penjumlahan Titik ini openMP diterapkan pada iterasi pencarian titik yang dibagi kedalam 4 bidang array berdasarkan jumlah *processor* yang digunakan pada komputer tempat penelitian dilakukan. Oleh karena itu, penerapan OpenMP sangat bermanfaat dalam Penjumlahan Titik, hal ini terlihat pada efisiensi waktu eksekusi pada setiap parameter luas bidang array yang digunakan pada setiap pengujian permasalahan Penjumlahan Titik.

Program paralel yang digunakan pada penelitian ini menggunakan OpenMP sebagai opsi pilihan dengan kinerja yang dihasilkan lebih baik dibandingkan dengan program serial seperti yang ditunjukkan pada gambar 2 diatas. Meskipun hasil perbandingan yang dihasilkan belum menunjukkan perubahan yang signifikan, hal ini mungkin bergantung pada ukuran dan kompleksitas bidang array, sehingga untuk mendapatkan perubahan yang signifikan dapat menggunakan parameter yang lebih kompleks. Ketika parameter yang digunakan semakin kompleks semakin signifikan pula efisiensi waktu yang dihasilkan, tentunya ini akan sangat bermanfaat dalam pemecahan masalah yang memerlukan program paralel tersebut seperti pada pemrosesan citra.

Selain itu, penggunaan OpenMp pada permasalahan Penjumlahan Titik ini merupakan dasar dari implementasi program paralel. Selain OpenMP, terdapat beberapa program paralel yang dapat digunakan seperti pemrograman MPI, *Multithreading* dan lain-lain. Pemrograman paralel ini memiliki banyak manfaat dalam mengefisiensi sebuah penyelesaian masalah yang besar.

4. KESIMPULAN

Penelitian ini memanfaatkan pemrograman OpenMP dalam mengoptimalkan waktu eksekusi penyelesaian Penjumlahan Titik. Pemanfaatan OpenMP juga dilakukan untuk membandingkan penggunaan program serial dalam Penjumlahan Titik. Hasil penelitian menunjukkan bahwa pemanfaatan program paralel dalam hal ini adalah OpenMP berhasil melakukan penjumlahan titik dengan waktu yang lebih efisien dibandingkan dengan penggunaan program serial.

Pemanfaatan program OpenMP telah membuktikan keefektifan program paralel dalam penyelesaian Penjumlahan Titik. Implementasi program paralel dalam penjumlahan titik menghasilkan waktu eksekusi yang efisien karena program paralel menerapkan penugasan eksekusi program yang secara bersamaan, sehingga dapat mengoptimalkan waktu eksekusi program.

REFERENSI

[1] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, 5th ed. Waltham: Morgan Kaufmann, 2012.

[2] A. Goel, S. S. Iyengar, and A. Raghuraman, “Efficient Parallel Algorithms for Dot Product on Modern Processors,” Journal of Parallel and Distributed Computing, vol. 72, no. 12, pp. 2947-2959, 2012.

[3] J. Zhang, J. Li, and S. Zhou, “Scalable Parallel Algorithms for Dot Product on Distributed Systems,” IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 1, pp. 191-203, 2016.

[4] M. J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw-Hill Education, 2004.

[5] I. Foster et al., “Parallel Programming in Split-C,” 1991.
[6] A. Ghoting et al., “SystemML: Declarative Machine Learning on MapReduce,” 2011.
[7] B. P. Miller et al., “Parallel Processing: Concurrency, Performance, and Latency,” 1993.
[8] A. Grama et al., Introduction to Parallel Computing, 2nd ed. Boston: Addison-Wesley, 2003.
[9] B. Chapman et al., Using OpenMP: Portable Shared Memory Parallel Programming, Cambridge: MIT Press, 2008.
[10] J. D. Foley et al., Computer Graphics: Principles and Practice, 3rd ed. Boston: Addison-Wesley, 2014.
[11] Bastian, A., Zaliluddin, D., & Al Maroghi, M. S. (2022). Implementasi Pemrograman Paralel Menggunakan Platform OpenMP pada Citra Digital dengan Metode Low-Pass Filter dan Histogram Equalization. INFOTECH Journal, 8(1), Juni.
[12] E. Ozturk and A. C. Yao, “A Survey on Parallel and Distributed Algorithms for Dot Product,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 2, pp. 487-504, 2014.