ANALISIS PERBANDINGAN PERFORMA PEMROGRAMAN SEKUENSIAL DAN PARALEL DENGAN SKEMA UJI MATRIX, FILTER DAN QUICK SORT

Griffani Megiyanto Rahmatullah[1)], Andry Fajar Zulkarnain[2)], M. Reza Hidayat[3)]
[1)]Jurusan Elektro, Politeknik Negeri Bandung
Jl. Gegerkalong Hilir, Ciwaruga, Kec. Parongpong, Kabupaten Bandung Barat, Jawa Barat 405592
[2)]Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Lambung Mangkurat
Jl. Brigjen Hasan Basry, Banjarmasin 70123
[3)]Fakultas Elektro Universitas Jenderal Achmad Yani
Jl. Terusan Jend. Sudirman, Cibeber, Kec. Cimahi Selatan, Kota Cimahi, Jawa Barat 40531
e-mail: griffani.megiyanto@polban.ac.id[1)], andry.zulkarnain@ulm.ac.id[2)], mreza@lecture.unjani.ac.id[3)]

ABSTRAK

Komputasi sekuensial merupakan sebuah proses untuk melakukan pemecahan masalah dengan melakukan setiap langkah secara berurutan. Konsekuensi pemecahan sebuah masalah dengan menggunakan komputasi sekuensial adalah sebuah hasil akan muncul apabila langkah pengerjaan telah dilakukan. Pengembangan teknologi dari komputasi sekuensial adalah komputasi paralel yang melibatkan penggunaan sumber daya secara bersamaan. Khusus pada bidang IT, sumber daya tersebut dapat berupa core processor atau juga dimungkinkan untuk melibatkan graphical unit. Skema uji yang dilakukan yaitu berfokus pada perbandingan performa komputasi sekuensial dan komputasi paralel. Skema uji tersebut terdiri dari pengujian perkalian matrix, proses filter sebuah gambar, dan proses quick sort. Hasil skema uji 1 menunjukkan bahwa komputasi paralel dapat melakukan perkalian dengan dimensi 2000x2000 dengan hasil berkisar 4x lebih cepat dibandingkan komputasi serial. Berikutnya, hasil skema uji 2 menunjukkan proses filter dapat dilakukan oleh komputasi paralel dengan efisiensi 50% lebih baik menggunakan 4 buah core. Terakhir, hasil skema uji 3 menghasilkan nilai efektivitas tertinggi menggunakan CUDA yaitu berkisar 96% dengan proses quick sort pada data sebesar 32Mb.

***Kata Kunci**: CUDA, filter, komputasi paralel, matrix, quick sort*

ABSTRACT

Sequential computing is a process for troubleshooting by performing each step sequentially. The consequence of solving a problem using sequential computing is that a result will appear when the work has been performed. Technological development of sequential computing is parallel computing that involves the simultaneous use of resources. Especially in the IT field, the resource can be a core processor or it is also possible to involve graphical units. The test scheme focuses on comparing sequential computing and parallel computing performance. The test scheme consists of matrix multiplication testing, an image filter process, and a quick sort process. The results of test scheme 1 show that parallel computing can perform multiplication with dimensions of 2000x2000 with results ranging from 4x faster than serial computing. Next, the results of test scheme 2 show the filter process can be done by parallel computing with 50% better efficiency using 4 cores. Lastly, the results of the 3rd test scheme produce the highest effectiveness value using CUDA which is around 96% with a quick sort process on the data of 32Mb.

***Keywords**: CUDA, filter, parallel computation, matrix, quick sort*

I. PENDAHULUAN

Komputasi adalah sebuah pemecahan masalah dari sebuah data masukan dan didapatkan data keluaran menggunakan algoritma yang telah ditentukan. Implementasi pada bidang computer dapat diterapkan menggunakan pemrograman serial / sekuensial atau pemrograman paralel. Dua kategori tersebut dipisahkan berdasarkan cara membaca dan menyelesaikan sebuah permasalahan. Pada pemrosesan serial, pemecahan masalah yang dilakukan oleh Central Processing Unit (CPU) dengan cara menjalankan setiap langkah secara berurutan dan mendapatkan sebuah hasil apabila langkah sebelumnya telah dilakukan [1]. Sedangkan Pemrosesan paralel yaitu penggunaan lebih dari satu CPU untuk menjalankan sebuah program secara simultan [1]. Idealnya, paralel processing membuat program berjalan lebih cepat karena semakin banyak CPU yang digunakan.

Komputasi paralel dapat diartikan sebagai salah satu teknik untuk melakukan komputasi secara bersamaan dengan memanfaatkan beberapa CPU secara bersamaan [2]. Hal tersebut biasanya diperlukan saat kapasitas atau proses pengerjaan sebuah masalah yang sangat besar, baik karena harus mengolah data dalam jumlah besar ataupun karena tuntutan proses komputasi yang banyak. Dengan mengetahui kemampuan tersebut, maka diperlukan analisis mendalam untuk melihat perbandingan antara komputasi sekuensial dan komputasi paralel.

II. GPU, CUDA, DAN MPI

Graphics Processing Unit (GPU) adalah sebuah alat yang berfungsi sebagai render grafis dalam kesatuan sistem komputer. GPU dapat berada pada Video Card atau terintegrasi dalam Motherboard berupa Integrated GPU. GPU berfungsi untuk mengolah dan memanipulasi grafis pada CPU (Central Processing Unit), untuk nantinya ditampilkan dalam bentuk Visual Grafis pada Monitor (output). Untuk dicapainya komputasi yang optimum, maka dikembangkan komputasi menggunakan GPU. Pemanfaatan komponen GPU menjadi sebuah hal penting karena dapat meningkatkan performa dari pemecahan masalah yang dilakukan [3].

Salah satu GPU yang umum telah digunakan adalah Compute Unified Device Architecture (CUDA). CUDA adalah salah satu contoh komputasi yang memanfaatkan GPU dengan Arsitektur yang dikembangkan oleh NVIDIA [4]. Pemanfaatan GPU pada paralel programming adalah dengan menggunakan CPU dan GPU bersama-sama dalam suatu model komputasi heterogen co-processing [5].

Berikutnya adalah salah satu pendukung pada perangkat lunak untuk paralel processing adalah penggunaan proses Message Passing [6]. Message Passing adalah suatu teknik untuk mengatur suatu alur komunikasi messaging terhadap proses pada system sedangkan interface penggunaan Message Passing dikenal sebagai Message Passing Interface (MPI). Message passing dalam ilmu komputer adalah suatu bentuk komunikasi yang digunakan dalam komputasi paralel, pemrograman-berorientasi objek, dan komunikasi interprocess. Salah satu contohnya yaitu OpenMP yang merupakan API dengan dukungan pada multi-platform serta berbagi memori multiprocessing dengan pemrograman C, C + +, dan Fortran. Dengan memanfaatkan OpenMP, maka dapat dilakukan proses compile secara paralel [7].

III. SKEMA PENGUJIAN

Skema pengujian pertama yaitu melakukan perkalian matrix. Perkalian matrix tersebut akan didefinisikan dengan dimensi 500x500 dan 2000x2000 dengan nilai yang berbeda di dalamnya. Pada pengujian tersebut akan dilakukan dengan menggunakan Teknik message passing yang melibatkan 3 threads.

Skema pengujian kedua yaitu simulasi proses filter gambar. Sebuah gambar memiliki noise dikarenakan nilai pixel yang tidak sesuai. Nilai pixel tersebut mengatur warna Red, Green dan Blue. Terdapat salah satu filter gambar yaitu center weighted average dimana dikenal juga filter teknik gaussian yaitu merubah nilai sebuah pixel berdasarkan perkalian pixel sebelumnya dengan nilai yang ditentukan lalu dibagi dengan rata – rata nilai pengali tersebut. Kegunaan lainnya adalah mengurangi noise pada gambar. Pada pengujian ini pixel dikalikan dengan nilai pengali sebanyak 3x3 dengan nilai pengalinya:

TABEL I
BLOK NILAI PENGALI 3X3

-1.25	0	-1.25
0	10	0
-1.25	0	-1.25

Penjumlahan dari setiap perkalian akan dibagi dari penjumlahan nilai pengali dan disimpan di center block. Pengujian dilakukan dengan membandingkan kinerja secara sekuensial dan paralel. Pada proses paralel gambar akan difilter dengan masing – masing prosesor diberikan nilai pixel untuk diproses dan dilakukan gather untuk dijadikan satu gambar kembali. Pengujian dilakukan menggunakan 2 gambar yaitu dengan ukuran 256*256 dan 512*512.

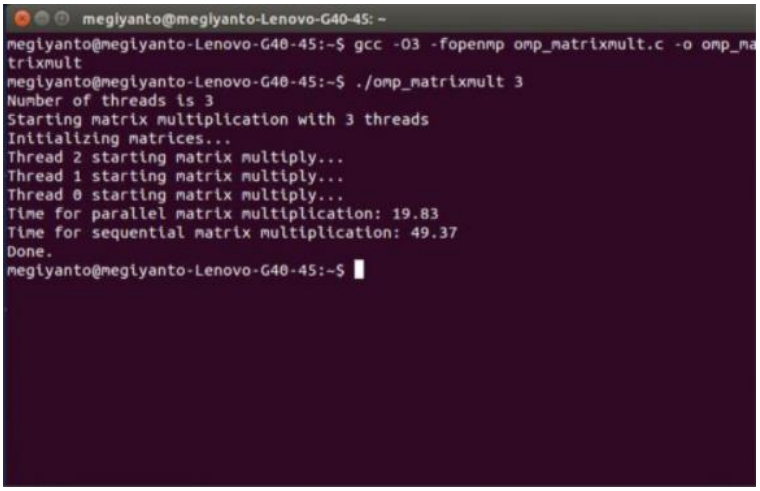
Skema pengujian ketiga yaitu pengujian dengan menerapkan teknik quick sort. Pengujian quick sort yang dilakukan akan menggunakan data dengan ukuran 1Mb sampai dengan 32Mb data.

IV. HASIL PENGUJIAN

Hasil pengujian skema 1 dapat dilihat pada gambar 1 berikut.

Gambar. 1. Pengujian matrix 500x500

Pengujian pertama yaitu perkalian matrix dengan dimensi 500x500 dihasilkan kecepatan sekuensial 0.90 detik dan kecepatan paralel 0.28 detik. Hasil tersebut menunjukkan penyelesaian secara paralel hampir 4x lebih cepat dibandingkan sekuensial. Namun, waktu yang didapat pada pengujian pertama dirasakan memiliki kecepatan yang sangat cepat sehingga tidak terlalu signifikan perbedaannya.

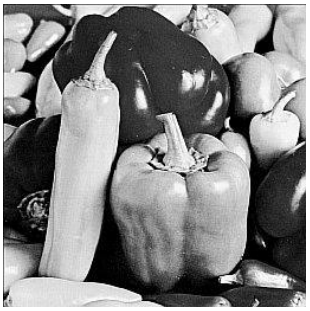


Gambar. 2. Pengujian matrix 2000x2000

Untuk pengujian berikutnya dilakukan dengan melakukan perkalian matrix dimensi 2000x2000 dan hasilnya ditunjukkan pada Gambar 2. Hasil yang didapatkan sangat terlihat secara signifikan pada perbedaan waktu dari paralel dan sekuensial. Proses paralel membutuhkan waktu 19.83 detik dan sekuensial membutuhkan waktu 49.37 detik. Hal tersebut menghasilkan simpulan bahwa sistem paralel yang dilakukan dapat melakukan perkalian matrix dimensi besar dengan waktu pengerjaan lebih cepat dibandingkan dengan sekuensial.

Berikutnya adalah hasil pengujian skema 2 ditunjukkan pada gambar berikut.

Gambar. 3. Gambar Original 256*256



Gambar. 4. Hasil filter metoda sekuensial

Gambar. 5. Hasil filter metoda paralel

Gambar. 6. Gambar Original 512*512

Gambar. 7. Hasil filter metoda sekuensial

Gambar. 8. Hasil filter metoda paralel

Pengujian skema 2 yang dilakukan membuktikan bahwa filter yang dibuat dapat mengurangi noise dari gambar original. Namun, akan terjadi perbedaan yang sangat mencolok apabila terjadi kesalahan perhitungan. Hal tersebut dapat terlihat dari hasil filter dengan metoda paralel yaitu terjadi perbedaan hasil perhitungan pixel sehingga membuat gambar menjadi lebih terang.

Untuk pengujian sekuensial dan paralel, khusus untuk paralel digunakan skenario menggunakan 2, 4, 6, 8, dan 10 prosesor. Hasil pengujian gambar 256*256 dapat dilihat pada tabel II.

TABEL II
TABEL PARAMETER HASIL PENGUJIAN GAMBAR 256*256

	Sekuensial	Paralel	Speed Up	Efisiensi	Efisiensi %
256*256	0.044868	0.023398	1.91759979	0.9587999	95.88%
		0.020579	2.18028087	0.54507022	54.51%
		4.83763	0.00927479	0.0015458	0.15%
		9.066256	0.0049489	0.00061861	0.06%
		11.177281	0.00401421	0.00040142	0.04%

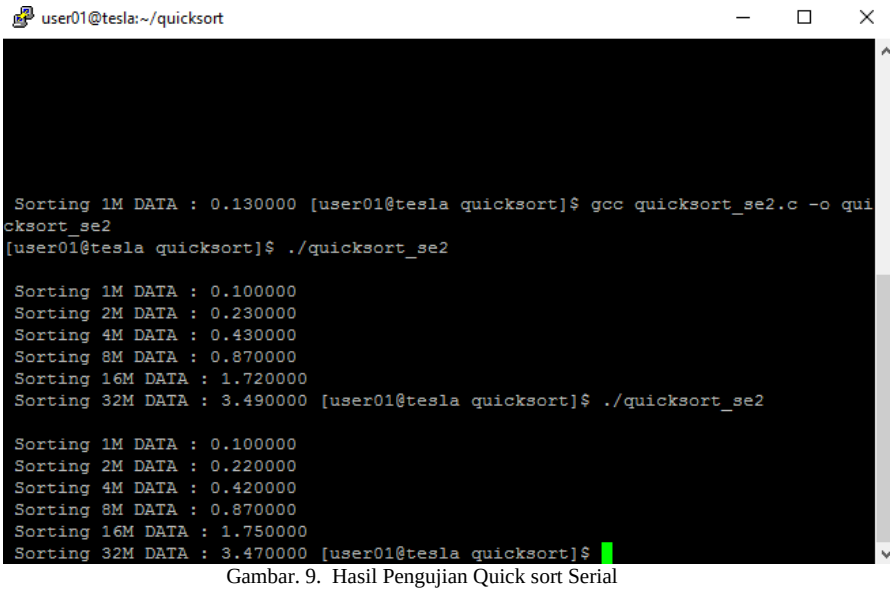
Berdasarkan hasil pengujian didapatkan nilai optimal proses filtering gambar dengan waktu 0.02 detik pada saat penggunaan 4 prosesor. Hal ini dikarenakan penggunaan alat uji yaitu sebuah laptop dengan spesifikasi quadcore (4 prosesor). Terjadi perbedaan yang sangat mencolok bila dipaksakan melakukan kinerja dengan konfigurasi lebih dari 4 prosesor. Hal ini terjadi pula untuk proses filter dengan gambar 512*512 yang dapat dilihat pada tabel III.

TABEL III
TABEL PARAMETER HASIL PENGUJIAN GAMBAR 512*512

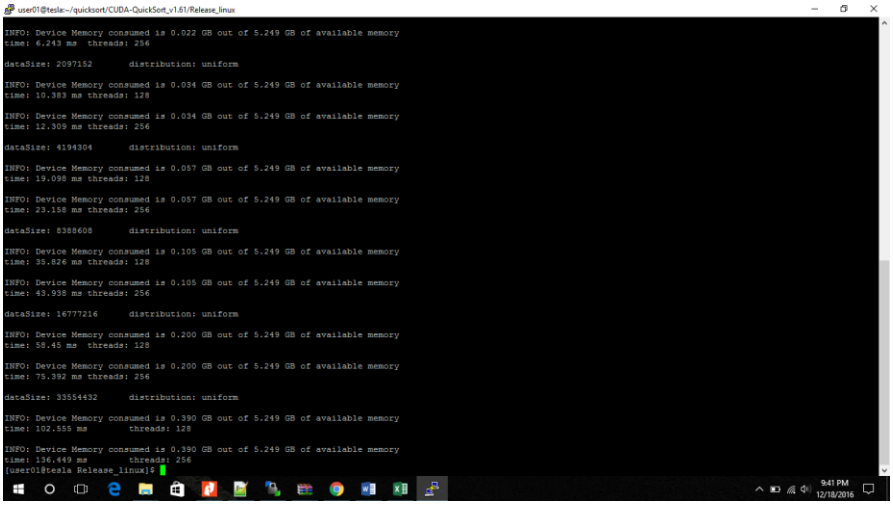
	Sekuensial	Paralel	Speed Up	Efisiensi	Efisiensi %
512*512	0.158404	0.08036	1.97117969	0.98558985	98.56%
		0.078793	2.01038163	0.50259541	50.26%
		10.48	0.01511489	0.00251915	0.25%
		20.6	0.00768951	0.00096119	0.10%
		21.79344	0.00726843	0.00072684	0.07%

Proses filter yang dilakukan untuk gambar 512*512 berdasarkan tabel 3 menunjukkan kembali bahwa optimal waktu yang didapatkan adalah pada simulasi 4 prosesor. berdasarkan hal tersebut dapat disimpulkan bahwa proses komputasi dapat berjalan maksimal sesuai dengan jumlah prosesor yang digunakan. Persamaan yang lain dari kedua pengujian tersebut adalah dilihat dari speed up dan efisiensi yang didapat. Speed up didapatkan berdasarkan perbandingan sekuensial dengan paralel dan menghasilkan kinerja paralel sekitar 2 kali lebih baik dibandingkan dengan sekuensial. Sedangkan efisiensi adalah perbandingan speed up dengan jumlah prosesor yang digunakan. Pada penggunaan 4 prosesor, efisiensi yang didapat sekitar 50 persen dan menunjukkan bahwa alat uji yang digunakan dapat melakukan proses komputasi yang lebih kompleks.

Berikutnya adalah hasil pengujian skema 3 ditunjukkan pada gambar 9 dan gambar 10.



Gambar. 9. Hasil Pengujian Quick sort Serial



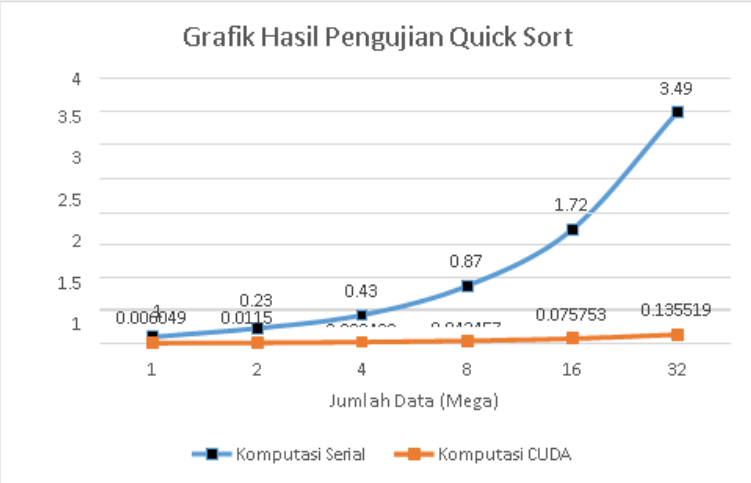
Gambar. 10. Hasil Pengujian Quick sort Cuda

Data yang didapatkan hasil dari gambar 9 dan gambar 10 dibuat pada tabel IV.

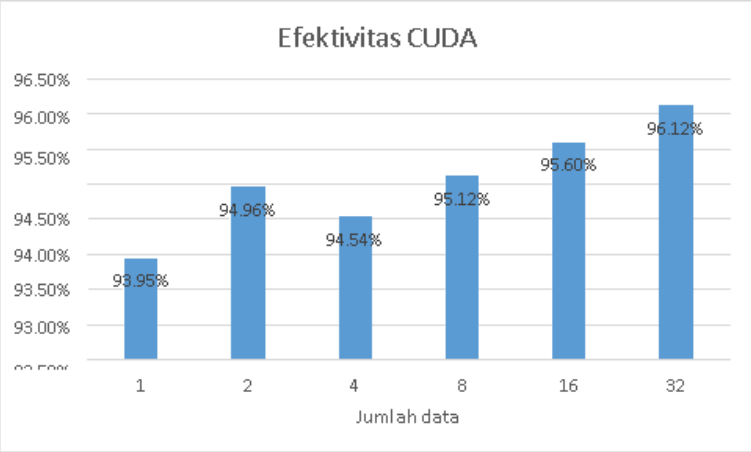
TABEL IV
Hasil Pengujian Quick Sort

Jumlah Data (Megabyte)	Waktu (detik)	
	Komputasi Serial	Komputasi CUDA
1	0.1	0.006049
2	0.23	0.011593
4	0.43	0.023492
8	0.87	0.042457
16	1.72	0.075753
32	3.49	0.135519

Hasil yang didapatkan menunjukkan bahwa CUDA dengan memanfaatkan GPU akan membuat kinerja komputasi semakin cepat. Banyaknya data dalam hal komputasi akan membuat CUDA menjadi lebih efektif dibandingkan dengan komputasi serial. Hal ini dijelaskan sebagai berikut dalam bentuk grafik pada gambar 11 dan 12.



Gambar. 11. Grafik Pengujian Quick sort



Gambar. 12. Efektivitas CUDA

Gambar 11 menunjukkan grafik yang naik untuk waktu komputasi serial. Namun, untuk komputasi GPU menggunakan CUDA grafik menunjukkan kenaikkan yang tidak signifikan. Bila dibandingkan hasil yang didapatkan, maka efektifitas yang tergambar pada gambar 12 menunjukkan bahwa komputasi paralel khususnya CUDA akan lebih efektif. Komputasi CUDA akan semakin efektif bila data yang diproses semakin banyak.

V. KESIMPULAN

Performa komputasi paralel yang telah diuji menghasilkan nilai yang lebih baik dibandingkan komputasi serial. Hal tersebut dapat ditunjukkan dari hasil masing – masing skema uji yaitu kecepatan proses berkisar 4x lebih cepat, efisiensi berkisar 50% lebih baik, dan nilai efektivitas tertinggi berkisar 96%. Walaupun nilai – nilai tersebut menunjukkan hasil yang positif, namun perlu dilakukan analisa lebih mendalam kembali dan pengujian lanjutan antara komputasi paralel dan komputasi serial.

DAFTAR PUSTAKA

[1] S. Rastogi and H. Zaheer, “Significance of paralel computation over serial computation,” 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016.
[2] M. Anas, I. G. P. S. Wijaya, and L. A. S. Irfan, “Studi Komputasi Paralel dan Implementasinya pada Kasus Komputasi Matriks Besar,” Journal of Computer Science and Informatics Engineering (J-Cosine), vol. 1, no. 1, p. 59, 2018.
[3] “Komputasi Paralel Menggunakan Nvidia Cuda Untuk Pemodelan 2D Tsunami Aceh Dengan Metode Lattice Boltzmann,” Frontiers: Jurnal Sains Dan Teknologi, vol. 1, 2018.
[4] “CUDA for All GPU and CPU Applications,” CUDA Application Design and Development, pp. 179–205, 2011.
[5] “Adaptive packet processing on CPU-GPU heterogeneous platforms,” Many-Core Computing: Hardware and Software, pp. 247–269, 2019.
[6] “Message Passing Interface,” Programming Models for Paralel Computing, 2015.
[7] S. Ohshima, S. Hirasawa, and H. Honda, “OMPCUDA: OpenMP Execution Framework for CUDA Based on Omni OpenMP Compiler,” Beyond Loop Level Paralelism in OpenMP: Accelerators, Tasking and More Lecture Notes in Computer Science, pp. 161–173, 2010.