Analisis Performa Komputasi Paralel GPU Menggunakan PYCUDA dan PYOPENCL dengan Komputasi Serial CPU pada Citra Digital

Muhammad Koprawi, Teguh Bharata Adji, Dani Adhipta
Jurusan Teknik Elektro dan Teknologi Informasi
Universitas Gadjah Mada
Jl.Grafika No. 2 Yogyakarta - 55281
m.koprawi.mti13@mail.ugm.ac.id, adji@ugm.ac.id, dani@ugm.ac.id

Intisari- Penggunaan teknologi pemrosesan paralel menjadi salah satu alternatif untuk menyelesaikan sebuah tugas yang membutuhkan komputasi besar. Teknologi ini sekarang banyak digunakan dibidang yang membutuhkan komputasi atau pemrosesan yang cepat seperti *computer science, image processing, cryptography, signal processing* dan komunikasi [1].

Pengolahan citra digital menjadi salah satu bidang yang bisa diproses menggunakan komputasi paralel. Semakin besar citra semakin besar pula kebutuhan komputasi yang diperlukan.

CUDA (*Compute Unified Device Architecture*) merupakan salah satu alternatif pemrosesan paralel dengan memanfaatkan GPU (*Graphics Processing Unit*). Pycuda hadir sebagai alernatif API (*Application Programming Interface*) python untuk mengakses GPU, selain itu adalagi alternatif lain yaitu pyopencl yang juga merupakan alternatif API dari python dengan memanfaatkan OPENCL.

Hasil penelitian ini menunjukan bahwa komputasi paralel lebih cepat dalam mengolah operasi citra (*graysacle, negatif, black and white*) daripada komputasi serial. Jenis perubahan (*grayscale, negatif dan black and white*) yang telah dijalankan dan diuji performanya, alokasi waktu yang diperlukan CPU untuk mengolah tiga jenis perubahan tersebut adalah yang paling lama dibandingkan dengan yang memanfaatkan GPU dalam hal ini adalah NVIDIA CUDA. Contoh untuk citra dengan ukuran 1600x1200 ditransformasikan menjadi citra *grayscale* memerlukan waktu 7,579 detik untuk komputasi serial (CPU) 0,233 detik untuk komputasi paralel menggunakan pycuda dan 0,236 detik untuk komputasi paralel menggunakan pyopencl.

Kata kunci - pengolahan citra digital, komputasi paralel, cuda, pycuda, pyopencl

Abstract—The use of parallel processing technology becomes an alternative to solve a task that requires large computing. This technology is now widely used in fields that require fast computing or processing such as computer science, image processing, cryptography, signal processing and communication [1].

Digital image processing becomes one of the areas that can be processed using parallel computing. The bigger the image the greater the computing needs are needed.

CUDA (Compute Unified Device Architecture) is an alternative to parallel processing by utilizing GPU (Graphics Processing Unit). Pycuda comes as an all-in-one python API to access the GPU, besides that it's another alternative that pyopencl is also an API alternative to python by using OPENCL.

The results of this study show that parallel computing is faster in processing image operations (graysacle, negative, black and white) than serial computing. The type of change (grayscale, negative and black and white) that has been run and tested for performance, the CPU time allocation to process the three types of change is the longest compared to the GPU in this case is NVIDIA CUDA. An example for a 1600x1200 image transformed into a grayscale image takes 7.579 seconds for serial computing (CPU) 0.233 seconds for parallel computing using pycuda and 0.236 seconds for parallel computing using pyopencl.

keywords - parallel computing, cuda, opencl, pycuda, pyopencl, python

I. PENDAHULUAN

A. Latar Belakang

Penggunaan teknologi pemrosesan paralel menjadi salah satu alternatif untuk menyelesaikan suatu tugas yang membutuhkan komputasi besar. Teknologi ini sekarang banyak digunakan di bidang yang membutuhkan komputasi atau pemrosesan yang cepat seperti *computer science, image processing, cryptography, signal processing* dan komunikasi [1].

Di era modern perkembangan teknologi informasi sangat cepat dan begitu pesat, belum selesai suatu produk *hardware* atau *software* dikenal atau dipakai oleh masyarakat keluar lagi produk baru yang lebih canggih dari yang sebelumnya. Semua teknologi terbaru saat ini yang paling banyak diperhatikan adalah bagian kecepatan dalam melakukan sebuah *processing* baik itu dari sisi *hardware* ataupun *software*. Tidak masalah jika *hardware* yang dipakai merupakan *hardware* keluaran terbaru, tapi bagaimana dengan *hardware* lama yang menginginkan kecepatan dalam melakukan *processing* tidak kalah dengan *hardware* keluaran terbaru. Masalah tersebut sekarang bisa teratasi dengan hadirnya teknologi CUDA (*Compute Unified Device Architecture*) yang berjalan pada *graphic card* yang dikeluarkan oleh NVIDIA.

Teknologi CUDA memanfaatkan GPU (*Graphics Processing Unit*) sebagai komputasi paralel yang bisa mempercepat kinerja komputer jauh lebih cepat dibandingkan dengan komputasi yang dilakukan oleh CPU (*Central Processing Unit*).

Teknologi CUDA ini dikembangkan oleh salah satu vendor *graphic card* yaitu NVIDIA. Dengan adanya CUDA, kartu grafis yang semula hanya dimanfaatkan sebagai pemrosesan grafis sekarang bisa juga dimanfaatkan sebagai komputasi lain di luar pemrosesan grafis.

Selain CUDA ada sebuah *framework* yang dirancang untuk keperluan komputasi paralel yaitu OpenCL (*Open Computing Language*). OpenCL sendiri memiliki sifat heterogen yaitu bisa dijalankan pada berbagai vendor kartu grafis seperti NVIDIA atau AMD[15] .

B. Rumusan Masalah

1. Pycuda sebagai API python yang merupakan kombinasi dari python dan CUDA hanya bisa berjalan pada kartu grafis NVIDIA untuk keperluan pemrosesan paralel, sedangkan berdasarkan paper [9][15] dijelaskan bahwa pyopencl yang merupakan kombinasi python dan opencl dapat berkerja pada komputasi CPU, komputasi GPU dan *mobile device,* tidak seperti pycuda yang hanya bisa berjalan pada kartu grafis NVIDIA, opencl bisa berjalan pada lintas vendor kartu grafis sehingga tidak terbatas pada salah satu kartu grafis saja. Karena hal tersebut perlu dilakukan pengujian antara kedua API tersebut.
2. Untuk menunjukkan peningkatan performa dan sebagai pembanding perhitungan komputasi paralel maka kedua API python (pycuda dan pyopencl) tersebut perlu dibandingkan dengan komputasi serial CPU.

C. Tujuan Penelitan

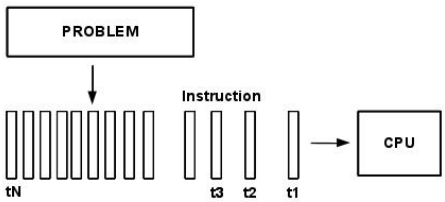
Penelitian ini bertujuan untuk melihat kinerja dan performa komputasi paralel GPU antara PYCUDA dan PYOPENCL terhadap citra digital dan juga membandingkan dengan komputasi serial CPU.

II. LANDASAN TEORI

A. Komputasi Paralel

Komputasi paralel adalah penggunaan sumber daya secara bersamaan dalam proses komputasi [2]. Komputasi paralel bisa menggunakan sebuah komputer dengan memanfaatkan beberapa processor atau bisa juga dengan memanfaatkan beberapa komputer yang terhubung melalui jaringan (*computer cluster*) atau juga bisa menggunakan kombinasi dari keduanya [2].

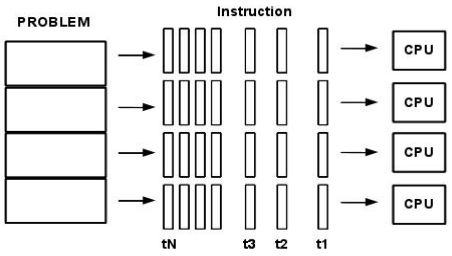
Perbandingan antara komputasi serial dan komputasi paralel bisa di lihat dari gambar 2.1 [2]



Gambar 2.1 Komputasi Serial

Terlihat komputasi serial untuk mengatasi suatu *problem* atau masalah akan dibagi menjadi beberapa instruksi kemudian dijalankan secara satu persatu oleh CPU, instruksi selanjutnya akan dijalankan ketika instruksi sebelumnya sudah selesai jadi tidak bisa dijalankan secara bersamaan. Ketika komputasi yang diperlukan besar maka otomatis akan memerlukan sumber daya yang besar juga.

Gambar 2.2 [2] menunjukan bagaimana komputasi paralel bekerja.



Gambar 2.2 Komputasi Paralel

Bagaimana komputasi paralel bekerja ketika ada suatu *problem* atau masalah akan membagi menjadi beberapa bagian *problem* yang masing-masing bagian memiliki instruksi-instruksi yang bisa dijalankan bersamaan dengan instruksi-instruksi lain dengan memanfaatkan jumlah *core* pada CPU.

B. Graphics Processing Unit (GPU)

GPU (*Graphics Processing Unit*) adalah salah satu teknologi pemrosesan data yang ada pada *graphics card* atau kartu grafis. GPU biasanya dipakai untuk keperluan multimedia seperti proses *rendering* [11]. GPU memiliki banyak *core* untuk melakukan pemrosesan yang mendukung banyak pemrosesan dalam satu waktu. Karena awalnya GPU hanya ditujukan untuk keperluan multimedia jadi ketika tidak ada *task* atau tugas yang berhubungan dengan multimedia otomatis GPU tidak digunakan.

GPU memiliki dua fitur yaitu GPU menggunakan banyak *core* dalam pemrosesan paralel untuk meningkatkan kinerja atau performa dan GPU memiliki memori *bandwidth* yang luas atau lebar untuk memberikan GPU kualitas transmisi data yang lebih baik [4] .

C. CUDA

CUDA (*Compute Unified Device Architecture*) merupakan toolkit NVIDIA yang memungkinkan seorang programer mengakses kartu grafis untuk tujuan komputasi umum [5]. CUDA hanya jalan pada *device* kartu grafis yang dibuat oleh NVIDIA. Untuk mengakses kartu grafis atau GPU diperlukan pemrograman tersendiri yang mirip dengan bahasa C, karena bahasa C dikhususkan untuk pemrograman yang bisa mengakses *hardware*. Ekstensi program dari CUDA ada .cu. Keahlian dalam bahasa C setidaknya sangat diperlukan supaya memudahkan adaptasi dengan programing CUDA. Karena C dan CUDA merupakan bahasa *compiler* maka ketika menjalankan program seluruh code dalam akan dieksekusi secara keseluruhan dan cukup sedikit menyulitkan ketika melakukan *debug error*.

D. OpenCL

OpenCL (*Open Computing Language*) adalah *framework* yang dibuat untuk tujuan komputasi paralel yang bisa berjalan pada CPU dan juga GPU. Berbeda dengan CUDA yang hanya bisa dijalankan pada GPU NVIDIA, OpenCL bisa dijalankan lintas vendor jadi tidak terbatas pada satu vendor saja misal NVIDIA tetapi juga bisa dijalankan pada kartu grafis AMD [15].

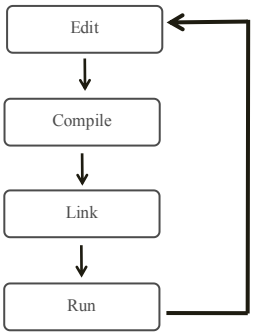
E. PYCUDA dan PYOPENCL

PYCUDA adalah salah satu alternatif untuk mengakses GPU. PYCUDA atau Python CUDA merupakan API yang disedikan oleh python untuk mengakses kartu grafis NVIDIA. PYCUDA menjadi salah satu pilihan karena kode yang dituliskan adalah layaknya seperti kode python biasa. Salah satu keuntungan menggunakan PYCUDA adalah seperti penanganan *error* yang lebih sederhana dibandingkan dengan bahasa C atau bahasa CUDA itu sendiri.

Keuntungan menggunakan PYCUDA [6] :

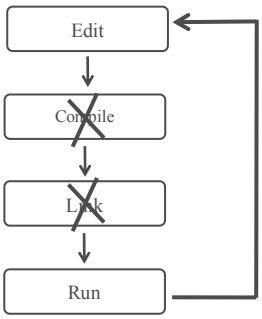
1. Tidak Perlu lagi menuliskan kode untuk melakukan pengujian error, karena python sudah melakukannya secara otomatis.
2. Tidak perlu menuliskan kode secara eksplisit untuk perpindahan data ke memori device.
3. Tidak perlu menuliskan kode untuk membebaskan memori.
4. Penanganan error yang lebih sederhana.
5. Penulisan variabel yang lebih sederhana dibandingkan dengan C, pada python tidak perlu menuliskan tipe data secara eksplisit pada variable karena python sudah mengenali tipe data tersebut secara otomatis.

Penulisan kode C *native* hanya dilakukan ketika membuat *function kernel*, selebihnya *pure* menggunakan sintaks python. Berikut workflow dari CUDA [7] :



Gambar 2.3 *Workflow* CUDA

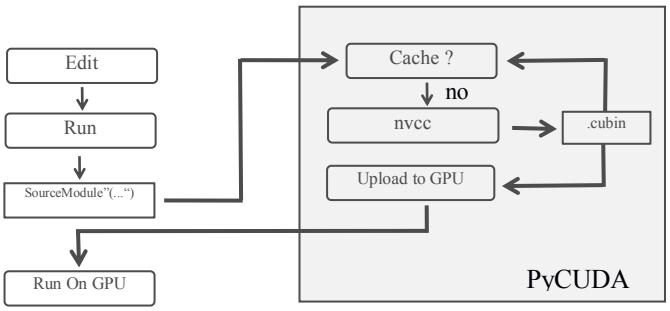
Pada gambar 2.3 di atas menjelaskan ketika program akan dijalankan pada CUDA terdapat beberapa tahapan yang harus dilewati sampai program tersebut berhasil dijalankan yaitu ada proses kompilasi dan *link* terlebih dahulu. Karena python merupakan *scripting language* jadi *workflow*nya adalah sebagai berikut :



Gambar 2.4 *Workflow* python

Terlihat pada gambar 2.4 [7] diatas ketika program python dieksekusi atau dijalankan tidak akan ada proses kompilasi dan *link*, melainkan langsung dieksekusi atau dijalankan.

Selanjutnya adalah *workflow* dari PYCUDA pada gambar 2.5 [7] sebagai berikut :



Gambar 2.5 *workflow* pycuda

Selain PYCUDA ada alternatif lain dalam mengakses GPU dengan API dari python untuk OPENCL yang bernama PYOPENCL [8]. OPENCL bisa diterapkan di kartu grafis NVIDIA, ATI dan juga bisa berjalan diatas komputasi serial. Menurut *paper* Massimo Di Pierro (2014) [9] PYOPENCL merupakan salah satu solusi untuk para scientis yang menggabungkan antara python dengan OPENCL yang bisa bekerja di komputasi CPU, komputasi GPU dan *mobile device*. Sebelumnya sudah dijelaskan performa yang dihasilkan ketika menggunakan CUDA dan OpenCL. Belum ada yang menjelaskan secara rinci mengenai perbandingan performa antara PYCUDA dan PYOPENCL ketika diterapkan pada aplikasi *digital watermarking*. Pada analisis yang dilakukan Brunton dan Zhao (2005) [10] menjelaskan tentang penggunaan komputasi paralel untuk *video watermarking* pada GPU, tapi tidak dijelaskan secara pasti bagaimana teknologi yang digunakan menggunakan CUDA atau OPENCL.

Pada OPENCL *kernel* akan dikompilasi ketika program dijalankan, ini merupakan sifat dari *cross-platform* atau OPENCL bisa dijalankan pada komputasi serial ataupun komputasi paralel tetapi memerlukan kinerja tambahan. Sedangkan CUDA, *kernel* hanya akan jalan pada GPU yang berasal dari kartu grafis NVIDIA [11].

F. Pengolahan citra digital

Pengolahan citra digital adalah suatu disiplin ilmu yang mempelajari hal-hal yang berhubungan dengan manipulasi dan modifikasi citra seperti perbaikan kualitas gambar, transformasi gambar, pemilihan citra ciri (feature images) dan hal lainnya yang berhubungan dengan citra. Adapun perbaikan kualitas gambar seperti peningkatan kontras, transformasi warna, restorasi citra. Sedangkan transformasi gambar seperti rotasi, translasi, skala dan transformasi geometrik [12][13]. Citra adalah suatu representasi (gambaran), kemiripan atau imitasi dari suatu objek. Citra terbagi dua yaitu citra analog dan citra digital. Resolusi adalah banyaknya piksel yang digunakan untuk membentuk sebuah citra digital, semakin tinggi resolusi, citra yang terbentuk akan semakin baik atau semakin bagus kualitas citra yang dihasilkan.

1. Transformasi citra warna menjadi grayscale

Citra warna (*true color*) bisa diubah menjadi citra *grayscale* dengan melakuan perubahan atau modifikasi pada tiap-tiap elemen warna (RGB).

Berikut transformasi citra warna menjadi citra *grayscale* pada gambar 2.6 :

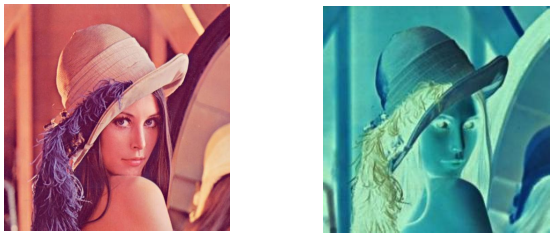
(a) (b)

Gambar 2.6 (a) citra warna (b) citra *grayscale*

2. Transformasi citra warna menjadi citra negatif

Citra negatif atau negasi (*invers*) merupakan citra kebalikan dari sebuah citra asli. Untuk mendapatkan citra negatif caranya adalah dengan mengurangi nilai intensitas piksel dari nilai keabuan maksimumnya.

Berikut transformasi citra warna menjadi citra negatif pada gambar 2.7 :



(a) (b)

Gambar 2.7 (a) citra warna (b) citra *negatif*

3. Transformasi citra warna menjadi citra biner

Citra biner (monokrom) atau citra hitam putih (*black and white*) adalah citra yang terdiri dari dua warna yaitu hitam dan putih. Untuk penentuan nilai hitam dan putih bisa menggunakan operasi ambang batas (*thresholding*) tunggal [12].

Berikut transformasi citra warna menjadi citra biner (*black and white*) pada gambar 2.8 :

(a) (b) (c)

Gambar 2.8 (a) citra warna (b) citra *grayscale* (c) citra *biner*

Citra biner ditransformasikan dari citra warna kemudian citra graysacle dan terakhir citra biner atau *black and white*.

H. Pengukuran

Performa atau peningkatan kecepatan (*speed up*) dihitung dengan cara membandingkan waktu eksekusi program serial dengan waktu eksekusi program paralel. Adapun pengukuran performa dapat dihitung menggunakan persamaan (2.1) [14].

$$S = \frac{T_s}{T_p}$$

S = *Speed Up*
T_s = *Waktu eksekusi program serial* (2.1)
T_p = *Waktu eksekusi program paralel*

III. METODOLOGI

Metode penelitian yang digunakan adalah metode eksperimental, yaitu dengan melakukan pengujian citra melalui pemrograman paralel dengan PYCUDA dan PYOPENCL yang dilakukan pada sebuah perangkat komputer yang mampu menjalankan komputasi GPU dan pemrograman serial yang dijalankan di CPU.

Hasil yang diharapkan nantinya dapat mengetahui kinerja dari komputasi GPU dengan PYCUDA dan PYOPENCL dalam menyelesaikan proses komputasi. Kemudian dua komputasi paralel tersebut akan dibandingkan sehingga menghasilkan perbandingan performa yang akan dibandingkan juga dengan komputasi CPU.

Adapun alat yang digunakan untuk mendukung penelitian ini adalah sebagai berikut :

A. Perangkat Lunak

1) *Sistem operasi Linux Ubuntu 14.04.4 LTS 64-bit*
2) *Python 2.7.6*
3) *CUDA (Compute Unified Device Architecture) Toolkit 6.5*
4) *PyCuda (Python + CUDA) v2015.1.3*
5) *PyOpenCL (Python + OpenCL) v2016.2.1*

6) Sublime Text (editor) 3 build 3114

B. Perangkat Keras

Perangkat keras yang digunakan pada penelitian ini adalah berupa laptop dengan spesifikasi sebagai berikut :

1) Processor intel core i5-2430M CPU @ 2.40GHz × 4
2) NVIDIA Geforce GT520 CUDA 1GB (48 cores)
3) RAM 4GB DDR3
4) Harddisk 500GB

C. Bahan

Citra warna yang akan diuji sebanyak 13 citra dengan ukuran bervariasi dari ukuran kecil sampai besar. Adapun ukuran citra tersebut dijelaskan pada gambar 3.1 :

Ukuran Piksel
400X300
640X480
800X600
1024X768
1600X1200
2272X1704
2816X2112
3264X2448
3648X2736
4096X3072
4480X3360
5120X3840
7216X5412

Gambar 3.1 ukuran piksel

D. Penerapan Metode

Penerapan metode untuk transformasi citra warna menjadi citra *grayscale*, citra negatif dan citra biner adalah sebagai berikut :

1) Metode transformasi citra warna menjadi citra grayscale.

Metode transformasi citra dari citra warna menjadi citra *grayscale* ditunjukan pada persamaan (3.1) [12][13].

$$L = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B$$

Atau

$$L = (R + G + B / 3)$$

L = *Luminosity/Grayscale*
R = *Red* (3.1)
G = *Green*
B = *Blue*

2) Metode transformasi citra warna menjadi citra negatif.

Metode transformasi citra dari citra warna menjadi citra negatif ditunjukan pada persamaan (3.2) [12][13].

$$R' = 255 - R$$
$$G' = 255 - G \quad (3.2)$$
$$B' = 255 - B$$

atau

$$R', G', B' = (255 - R, 255 - G, 255 - B)$$

Kemudian tercipta citra negatif dengan komposisi baru sebagai berikut :

$$N = R'G'B'$$

N = Negatif	R' = *New Red*
R = *Red*	G' = *New Green*
G = *Green*	B' = *New Blue*
B = *Blue*	

3) Metode transformasi citra warna menjadi citra biner.

Metode transformasi citra dari citra warna menjadi citra biner ditunjukan pada persamaan (3.3) [12][13].

$$L = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B$$

IF L < 128 :

value = 0
ELSE

value =255

L = *Luminosity/Grayscale* (3.3)
R = *Red*
G = *Green*
B = *Blue*
Value = nilai baru yang diterapkan ke *channel* RGB

E. Tahapan Perhitungan

Pada bagian tahapan perhitungan, penulis membagi perhitungan menjadi 4 (empat) tahapan yaitu :

1) Waktu yang diperlukan untuk memuat data (load data) dan proses konversi image menjadi array.
2) Waktu yang diperlukan untuk proses modifikasi piksel.
3) Waktu yang diperlukan untuk menyimpan gambar (save image time).
4) Total waktu eksekusi (total execution time) dari tahapan pertama sampai tahapan ketiga.

Untuk menghitung waktu penulis menggunakan fungsi bawaan (*built in*) dari python yaitu fungsi *clock()*. Untuk menggunakan fungsi *clock()* terlebih melakukan *import module* pada python yaitu *module time* dengan perintah *import module*.

IV. HASIL DAN PEMBAHASAN

Operasi citra yang akan dilakukan adalah dengan mengubah citra RGB (*red, green, blue*) menjadi *grayscale*, negatif dan *black and white*. Untuk masing-masing perubahan citra memiliki *algoritme* atau metode tersendiri.

A. Grayscale

Berikut hasil perbandingan rata-rata yang dihasilkan pada operasi citra *grayscale* pada tabel 4.1.

Tabel 4.1 Tabel hasil pengujian citra RGB menjadi citra *grayscale.*

SIZE	CPU (detik)	PYCUDA (detik)	PYOPENCL (detik)
400X300	0,483	0,050	0,055
640X480	1,235	0,053	0,071
800X600	1,905	0,077	0,092
1024X768	3,154	0,111	0,123
1600X1200	7,579	0,233	0,236
2272X1704	15,101	0,420	0,404
2816X2112	23,205	0,619	0,579
3264X2448	31,344	0,802	0,739
3648X2736	41,611	0,975	0,892
4096X3072	58,515	1,192	1,083
4480X3360	74,859	1,398	1,260
5120X3840	201,500	1,759	1,560
7216X5412	355,325	3,220	2,801

Dapat dilihat dari tabel 4.1 perbedaan hasil pengujian antara komputasi serial dengan komputasi paralel, semakin besar ukuran citra yang diuji semakin besar waktu yang diperlukan untuk melakukan operasi citra.

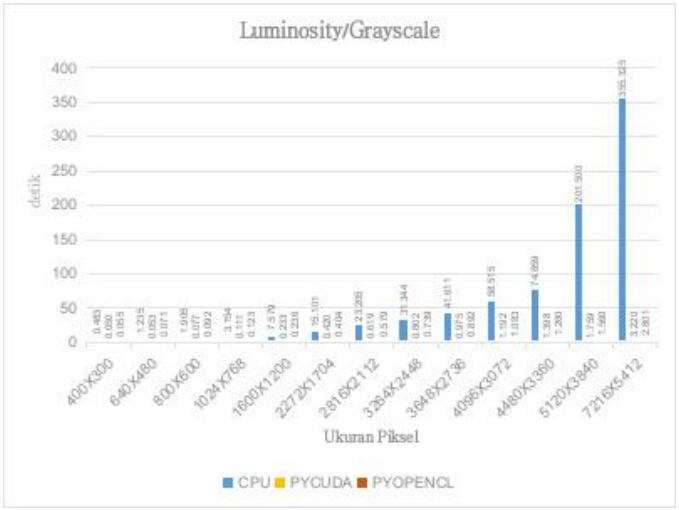
Berikut hasil perhitungan peningkatan kecepatan (*speed up*) dapat dilihat pada tabel 4.2.

Tabel 4.2 Tabel hasil perhitungan peningkatan kecepatan pada citra *grayscale.*

SIZE	SPEED UP	
	CPU/PYCUDA	CPU/PYOPENCL
400X300	9,648	8,861
640X480	23,097	17,474
800X600	24,785	20,805
1024X768	28,390	25,664
1600X1200	32,470	32,140
2272X1704	35,992	37,364
2816X2112	37,479	40,085
3264X2448	39,075	42,443
3648X2736	42,678	46,639
4096X3072	49,086	54,048
4480X3360	53,547	59,404
5120X3840	114,570	129,138
7216X5412	110,342	126,861

Dari hasil tabel 4.2 diatas dapat dilihat peningkatan kecepatan antara komputasi serial dengan komputasi paralel untuk operasi citra *grayscale* mencapai lebih dari 100 kali lipat pada citra dengan ukuran lebih dari 5000 piksel.

Adapun grafik pengukuran hasil pengujian pada operasi citra *grayscale* dapat dilihat pada gambar grafik 4.1.



Gambar 4.1. Gambar grafik hasil pengujian operasi citra *grayscale*

Seperti yang sudah dijelaskan sebelumnya pada tabel 4.1, pada gambar 4.1 terlihat peningkatan atau banyaknya waktu komputasi yang dibutuhkan untuk melakukan suatu operasi citra *grayscale* yang disajikan dalam berntuk grafik. Terlihat jelas komputasi serial memerlukan lebih banyak waktu untuk melakukan proses operasi citra *graysacle* dibandingkan dengan komputasi paralel.

B. Negatif

Berikut Hasil pengujian perubahan citra RGB menjadi citra negatif dapat dilihat pada tabel 4.3.

Tabel 4.3 Tabel hasil pengujian citra RGB menjadi citra *negatif.*

SIZE	CPU (detik)	PYCUDA (detik)	PYOPENCL (detik)
400X300	1,413	0,034	0,052
640X480	3,616	0,078	0,071
800X600	5,593	0,077	0,093
1024X768	9,213	0,110	0,126
1600X1200	22,357	0,233	0,237
2272X1704	51,285	0,419	0,405
2816X2112	132,522	0,615	0,580
3264X2448	260,148	0,798	0,742
3648X2736	275,833	0,969	0,895
4096X3072	299,418	1,185	1,086
4480X3360	418,697	1,393	1,264
5120X3840	393,726	1,747	1,564
7216X5412	1113,445	3,202	2,808

Pada tabel 4.3 menjelaskan hasil pengujian yang telah dilakukan pada operasi citra dari citra RGB menjadi citra negatif. Komputasi serial memerlukan lebih banyak waktu dibandingkan dengan komputasi paralel.

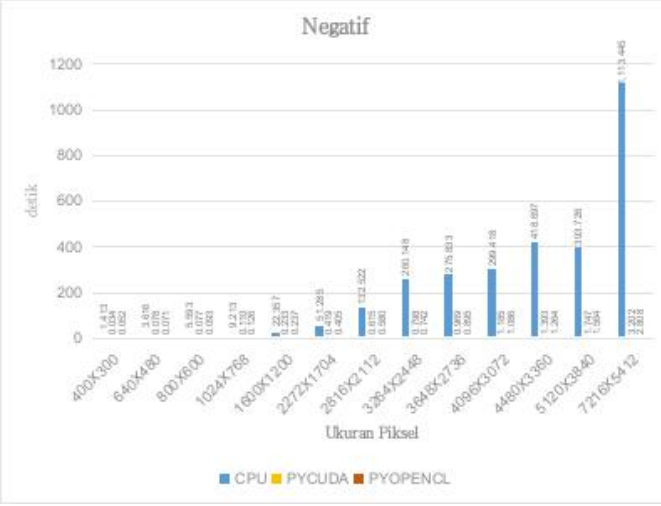
Hasil perhitungan peningkatan kecepatan (*speed up*) pada citra negatif dapat dilihat pada tabel 4.4.

Tabel 4.4 Tabel hasil perhitungan peningkatan kecepatan pada citra *negatif.*

SIZE	SPEED UP	
	CPU/PYCUDA	CPU/PYOPENCL
400X300	41,624	27,255
640X480	46,634	50,722
800X600	72,826	60,335
1024X768	83,529	73,179
1600X1200	95,830	94,255
2272X1704	122,544	126,520
2816X2112	215,377	228,387
3264X2448	325,836	350,533
3648X2736	284,614	308,159
4096X3072	252,609	275,770
4480X3360	300,572	331,195
5120X3840	225,366	251,743
7216X5412	347,745	396,582

Dari hasil tabel 4.4 diatas dapat dilihat peningkatan kecepatan antara komputasi serial dengan komputasi paralel untuk operasi citra negatif mencapai lebih dari 100 kali lipat pada citra dengan ukuran lebih dari 2200 piksel.

Adapun grafik pengukuran hasil pengujian pada operasi citra negatif dapat dilihat pada gambar grafik 4.2.



Gambar 4.2. Gambar grafik hasil pengujian operasi citra *negatif.*

Pada gambar 4.2 terlihat peningkatan atau banyaknya waktu komputasi yang dibutuhkan untuk melakukan operasi citra negatif yang disajikan dalam berntuk grafik. Terlihat jelas komputasi serial memerlukan lebih banyak waktu untuk melakukan proses operasi citra negatif dibandingkan dengan komputasi paralel.

C. Black and White

Berikut Hasil pengujian perubahan citra RGB menjadi citra *black and white* dapat dilihat pada tabel 4.5.

Tabel 4.5. Tabel hasil pengujian citra RGB menjadi citra *black and white.*

SIZE	CPU (detik)	PYCUDA (detik)	PYOPENCL (detik)
400X300	0,484	0,035	0,052
640X480	1,208	0,054	0,072
800X600	1,905	0,078	0,093
1024X768	3,102	0,112	0,123
1600X1200	7,629	0,234	0,237
2272X1704	15,284	0,420	0,405
2816X2112	23,535	0,619	0,580
3264X2448	34,742	0,800	0,741
3648X2736	40,021	0,974	0,894
4096X3072	67,178	1,192	1,085
4480X3360	76,577	1,400	1,264
5120X3840	128,799	1,758	1,565
7216X5412	381,599	3,220	2,809

Pada tabel 4.5 menjelaskan hasil pengujian yang telah dilakukan pada operasi citra dari citra RGB menjadi *citra black and white*. Komputasi serial memerlukan lebih banyak waktu dibandingkan dengan komputasi paralel. Semakin besar ukuran citra maka semakin lama pula waktu yang diperlukan untuk melakukan operasi citra.

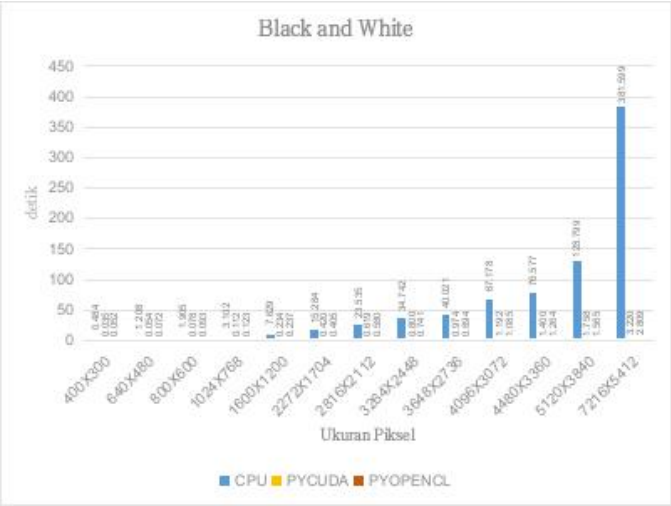
Hasil perhitungan peningkatan kecepatan (*speed up*) pada citra *black and white* dapat dilihat pada tabel 4.6.

Tabel 4.6 Tabel hasil perhitungan peningkatan kecepatan pada citra *negatif.*

SIZE	SPEED UP	
	CPU/PYCUDA	CPU/PYOPENCL
400X300	14,016	9,353
640X480	22,430	16,881
800X600	24,471	20,473
1024X768	27,670	25,157
1600X1200	32,637	32,237
2272X1704	36,378	37,771
2816X2112	38,042	40,601
3264X2448	43,413	46,894
3648X2736	41,079	44,782
4096X3072	56,365	61,892
4480X3360	54,702	60,607
5120X3840	73,264	82,313
7216X5412	118,500	135,863

Dari hasil tabel 4.6 diatas dapat dilihat peningkatan kecepatan antara komputasi serial dengan komputasi paralel untuk operasi citra *black and white* mencapai lebih dari 100 kali lipat pada citra dengan ukuran lebih dari 7000 piksel.

Adapun grafik pengukuran hasil pengujian pada operasi citra *black and white* dapat dilihat pada gambar grafik 4.3.



Gambar 4.3. Gambar grafik hasil pengujian operasi citra *black and white*.

Pada gambar 4.3 terlihat peningkatan atau banyaknya waktu komputasi yang dibutuhkan untuk melakukan sebuah operasi citra *black and white* yang disajikan dalam berntuk grafik. Terlihat jelas komputasi serial memerlukan lebih banyak waktu untuk melakukan proses operasi citra *black and white* dibandingkan dengan komputasi paralel.

Dari ketiga proses operasi citra yang telah dilakuan, komputasi serial memerlukan lebih banyak waktu untuk melakukan operasi citra dibandingkan dengan komputasi paralel. Perbedaan jenis operasi citra sangat berpengaruh terhadap waktu yang dihasikan pada pengujian.

V. KESIMPULAN

Berdasarkan hasil penelitian analisis performa terhadap citra *image* dengan ukuran piksel yang berbeda-beda dan diterapkan terhadap 3 (tiga) jenis perubahan citra yaitu dengan merubah citra image RGB menjadi citra *grayscale*, negatif dan *black and white* dapat diambil kesimpulan sebagai berikut :

1. Jenis perubahan (*grayscale*, negatif dan *black and white*) yang telah dijalankan dan diuji performanya, alokasi waktu yang diperlukan CPU untuk mengolah tiga jenis perubahan tersebut yang paling lama dibandingkan dengan yang memanfaatkan GPU dalam hal ini adalah NVIDIA CUDA. Contoh untuk citra dengan ukuran 1600x1200 ditransformasikan menjadi citra *grayscale* memerlukan waktu 7,579 detik untuk komputasi serial (CPU) 0,233 detik untuk komputasi paralel menggunakan PYCUDA dan 0,236 detik untuk komputasi paralel menggunakan PYOPENCL.
2. Selisih waktu yang dihasilkan oleh CPU dibandingkan dengan GPU sangat besar perbedaannya, bisa mencapai 100 (seratus) kali lipat. Contoh untuk citra dengan ukuran 7216x5412 ditransformasikan menjadi citra *grayscale* kemudian dilakukan perhitungan selisih antara komputasi serial dengan komputasi paralel. Selisih waktu komputasi serial CPU dengan komputasi paralel PYCUDA mencapai 110,342 kali lipat dan Selisih waktu komputasi serial CPU dengan komputasi paralel PYOPENCL mencapai 126,861 kali lipat.
3. Selisih waktu yang dihasilkan oleh pycuda dan pyopencl yang sama-sama memanfaatkan GPU hanya berkisar paling banyak 1 (satu) detik berdasarkan pengukuran. Dengan selisih waktu yang berkisar paling banyak 1 (satu) detik, pyopencl bisa menjadi alternatif bahasa pemorgraman untuk keperluan komputasi paralel.

DAFTAR PUSTAKA

[1] M. A. Gray, “Getting started with gpu programming,” Comput. Sci. Eng., vol. 11, no. 4, pp. 61–64, 2009.

[2] Blaise Barney “What is Parallel Computing ?” [online] 2009. Available : https://computing.llnl.gov/tutorials/parallel_comp/

[3] http://developer.download.nvidia.com/books/cuda-by-example/cuda-by-example-sample.pdf

[4] C.-L. Su, P.-Y. Chen, C.-C. Lan, L.-S. Huang, and K.-H. Wu, “Overview and comparison of OpenCL and CUDA technology for GPGPU,” 2012, pp. 448–451.

[5] S. J. Park, “An Analysis of GPU Parallel Computing,” 2009, pp. 365–369.

[6] https://mathema.tician.de/software/pycuda/

[7] pyCUDA simpler GPU Programming Python.pdf

[8] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih, “PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation,” Parallel Comput., vol. 38, no. 3, pp. 157–174, 2012.

[9] Massimo Di Pierro “Portable Parallel Programs with Python and OpenCL.pdf.”,2014 .

[10] A. Brunton and J. Zhao, “Real-time video watermarking on programmable graphics hardware,” 2005, vol. 2005, pp. 1312–1315.

[11] X. Wang, X. Li, M. Zou, and J. Zhou, “AES finalists implementation for GPU and multi-core CPU based on OpenCL,” 2011, pp. 38–42.

[12] T. Sutoyo, S.Si., M.Kom., Edy Mulyanto, S.Si., M.Kom., Dr. Vincent Suhartono, Oky Dwi Nurhayati, M.T., Wijanarto, M.Kom., “Teori Pengolahan Citra Digtal”. Semarang : ANDI , 2009.

[13] Abdul Kadir dan Adhi Susanto, “Teori dan Aplikasi Pengolahan Citra”, Yogyakarta : ANDI, 2013.

[14] Barry Wilkinson, Michael Allen , ”Parallel Programming Techniques and Application Using Networked Workstation and Parallel Computers – second Edition”, 2005.

[15] Andreas Kl ץockner Easy, Effective, Efficient: GPU Programming in Python with PyOpenCL and PyCUDA