**LAPORAN PRAKTIKUM**

**INSTALASI CUDA & PYCUDA**

**PEMROGRAMAN PARAREL**

****

**Anggota Kelompok :**

1. **Hot G.N Imanta S (19102013)**
2. **Eko Yanuarso Budi (19102124)**
3. **Filfimo Yulfis A.H. (19102143)**
4. **M.Thoriq Afif (19102153)**
5. **Wahyu Fajaruloh (19102164)**
6. **M.Eris Pumayoga.A (19102290)**

**KELAS S1-TI-1**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2021**

Pengenalan GPU dan PyCuda

*Graphics Processing Unit* atau lebih dikenal dengan GPU adalah satuan elektronik sirkuit yang mempunyai spesialisasi dalam memproses data untuk memproses gambar dari bentuk polygonal primitives. Meskipun GPU didesain untuk memproses gambar, GPU terus mengalami perkembangan, menjadi lebih kompleks dan efisien dalam melayani secara langsung kebutuhan komunitas dalam melakukan teknik komputasi. Dengan banyaknya cores yang tersedia, vendor GPU mulai mengembangkan *parallel programming*.

Setiap GPU terdiri dari beberapa *processing unit* yang dinamakan *Streaming Multiprocessor* (SM) yang merepresentasikan tingkatan logika dari paralel, dan setiap SM nyatanya terus bekerja secara terus menerus dan mandiri terpisah antara satu dan yang lain. Setiap SM dapat dobagi menjadi bagian yang dinamakan *Stream Processors* (SP), yang mana setiap *core* dapat melakukan eksekusi dan dapat berjalan berurutan dalam menjalankan *thread*. Setiap SP merepresentasikan bentuk terkecil dari logika eksekusi dan merepresentasikan tingkatan dari paralel. Setiap *cores* membuat bagian yang menjalankan instruksi yang sama pada waktu bersamaan. Ini dinamakan model *Single instruction, mutiple data* (SIMD)

Teknik Programming

Seperti yang telah dibahas sebelumnya, GPU terstruktur berbeda dari prosesor pada umumnya; untuk itu, GPU mempunyai masalah yang berbeda dan membutuhkan teknik yang spesifik dalam melakukan programming. Fitur yang paling menonjol dari GPU adalah tingginya jumlah *cores* yang tersedia, yang mana mengizinkan kita untuk mengeksekusi banyak *threads*, yang sebagian disinkronisasi untuk eksekusi operasi yang sama.

CUDA dan OpenCL

Saat ini, solusi yang paling efektif untuk mengeksploitasi daya komputasi yang disediakan oleh GPU adalah dengan sebuah *software* *library* yang dinamakan CUDA (*Compute Unified Device Architecture*) dan OpenCL (*Open Computing Language*).

CUDA dan OpenCL adalah framework yang digunakan untuk mengeksploitasi kartu grafis GPU.

PyCUDA

PyCUDA adalah model programming yang didesain untuk eksekusi umum dari program CPU dan GPU, mengizinkan untuk melakukan bagian berurutan dari CPU dan bagian numerik, yang mana lebih intensif pada CPU

Instalasi PyCUDA

Syarat :

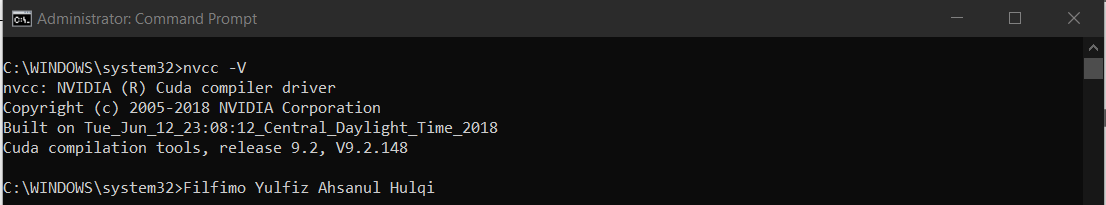
* Memiliki GPU Nvidia
* Jika tidak dapat menggunakan Google Colaboration

Langkah Instalasi :

* Install Python 2.7
* Intall Visual Studio 2015
* Install Visual C++ 9
* Install Nvidia Toolkit 9
* Install PyCuda 🡪 **pip install pycuda**

1. Install CUDA Nvidia dan PyCuda

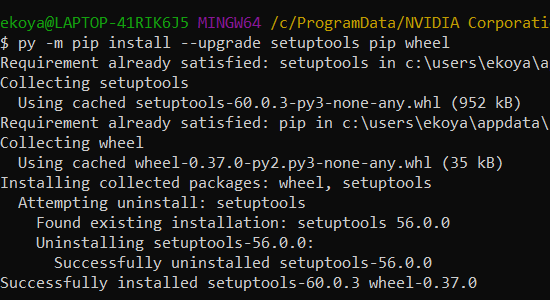
Pastikan *environment* untuk melakukan komputasi GPU telah terinstall dengan menjalankan *command* **nvcc -V**.



Gambar 1 Cek CUDA pada laptop

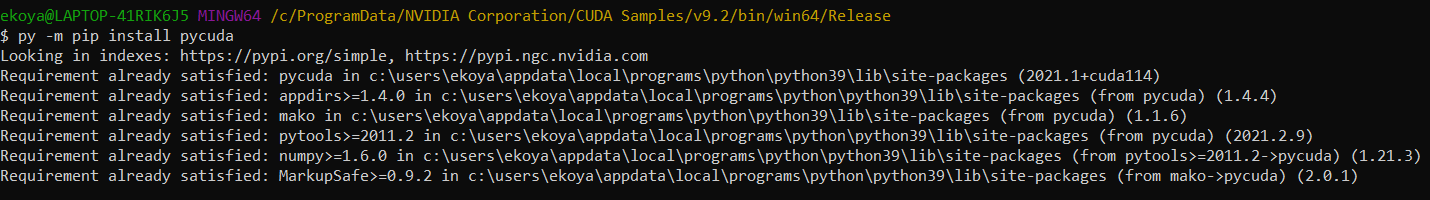
Cek Cuda Compiler Driver, Nvidia CUDA dapat diunduh melalui link [ini](https://developer.nvidia.com/cuda-downloads). Setelah diunduh ikuti langkah-langkah instalasi yang telah diberikan, kemudian dapat melakukan cek versi dan validasi.

Install PyCuda dengan menjalankan **install pycuda -m pip install pycuda**



Gambar 2 Instalasi pip pycuda oleh Eko

Proses intalasi **pip** dilakukan didalam terminal, jika baru pertama kali akan muncul beberapa instuksi, cukup ikuti instruksi yang diberikan untuk menyelesaikan instalasi.



Gambar 3 Instalasi pip pycuda telah selesai

Dengan begini *library* **pycuda**telah berhasil diinstall pada sistem, dan siap untuk digunakan.

1. Program pycuda 1

Source Code :

import pycuda.driver as drv

drv.init()

print("%d device(s) found." % drv.Device.count())

for ordinal in range(drv.Device.count()):

dev = drv.Device(ordinal)

print("Device #%d: %s" % (ordinal, dev.name()))

print(" Compute Capability: %d.%d " % dev.compute\_capability())

print(" Total Memory: %s KB" % (dev.total\_memory()//(10214)))

Diatas adalah kode program untuk mengecek kartu grafis apa yang digunakan pada device kita.

Ouput Program :



Gambar 4 Hasil dari kode program setelah dijalankan

Setelah program dijalankan maka, akan didapatkan hasil seperti diatas, disini kartu grafis yang terdekteksi berjumlah satu, tipe GTX 1650, kemampuan komputasi sebesar 7.5 dan total memori 4GB.

1. Program pycuda 2

Source Code :

import pycuda.driver as cuda

import pycuda.autoinit

from pycuda.compiler import SourceModule

import numpy

import os

a = numpy.random.randn(5,5)

a = a.astype(numpy.float32)

a\_gpu = cuda.mem\_alloc(a.nbytes)

cuda.memcpy\_htod(a\_gpu,a)

mod = SourceModule("""

\_\_global\_\_ void doubleMatrix(float \*a)

{

int idx = threadIdx.x + threadIdx.y\*5;

a[idx]

}

""")

func = mod.get\_function("doubleMatrix");

func(a\_gpu, block(5,5,1))

"block 5,5,1 -> thread x = 5, thread y = 5, thread z = 1"

a\_doubled = numpy.empty\_like(a)

cuda.memcpy\_dtoh(a\_doubled, a\_gpu)

print("Original Matrix")

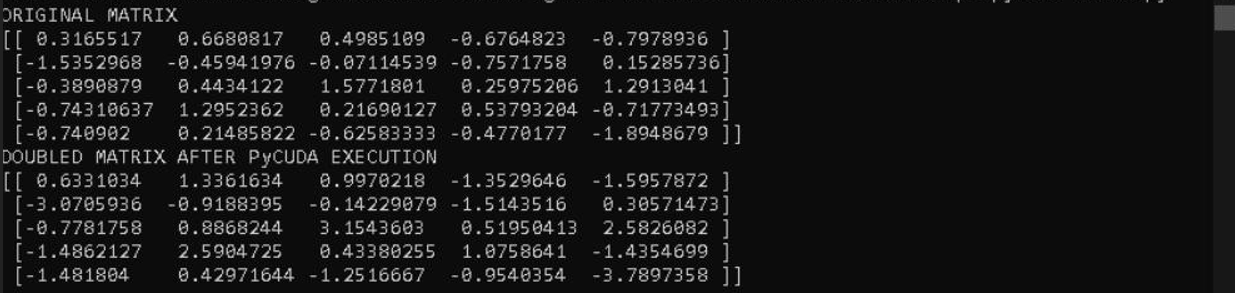
print (a)

print("Doubled Matrix After PyCuda Execution")

print (a\_doubled)

Diatas adalah kode program untuk melakukan operasi matrix menggunakan GPU, disini akan melakukan operasi *doubledMatrix*.

Output Program :



Gambar 5 Hasil dari kode program setelah dijalankan oleh Dimas

Pada bagian ini, pada program saya mengalami *error*, dimana tidak adanya file dengan tipe *dll*, setelah mencari banyak sumber untuk mengatasi *error* tersebut, program tetap tidak dapat menampilkan *output* apapun, disini saya meminta bantuan dari kelompok lain untuk mendapatkan *ouput* dan didapatkan hasil seperti diatas.

Program diatas akan menampilkan matriks awal secara acak, matriks awal tersebutlah yang akan dilakukan operasi *doubledMatrix*, dan hasil dari *doubled* tersebut dapat dilihat dibawah dari hasil matriks awal.