

Phần 1. Cài đặt CUDA và sử dụng cuFFT

1.1. Đối với máy tính có GPU






Bước 1: Kiểm tra tính tương thích của hệ thống

1. Kiểm tra GPU: Xác định xem GPU của bạn có hỗ trợ CUDA hay không:

- Truy cập [NVIDIA CUDA GPUs](#) để xem danh sách GPU hỗ trợ.
- Trên Windows: Mở Command Prompt và chạy nvidia-smi.
- Trên Linux: Chạy nvidia-smi trong Terminal.

2. Hệ điều hành: Đảm bảo hệ điều hành (Windows, Linux hoặc macOS) tương thích với phiên bản CUDA bạn muốn cài đặt.

Click the sections below to expand

	CUDA-Enabled Datacenter Products
	CUDA-Enabled NVIDIA Quadro and NVIDIA RTX
	CUDA-Enabled NVS Products
	CUDA-Enabled GeForce and TITAN Products
	CUDA-Enabled Jetson Products

[Download the CUDA Toolkit](#)


```
Command Prompt
(c) Microsoft Corporation. All rights reserved.

C:\Users\nguye>nvidia-smi
Sun Nov 17 20:40:55 2024

+-----+
| NVIDIA-SMI 551.61                Driver Version: 551.61          CUDA Version: 12.4     |
+-----+-----+-----+-----+-----+-----+
| GPU   Name                               TCC/WDDM | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf              Pwr:Usage/Cap |      Memory-Usage | GPU-Util  Compute M. |
|=====+=====+=====+=====+=====+=====+
|  0  NVIDIA GeForce RTX 3050 ... WDDM | 00000000:01:00.0 On |          1%      N/A |
| N/A   54C    P5              7W /   75W | 1665MiB /  4096MiB |             Default  |
|=====+=====+=====+=====+=====+=====+
|
+-----+
| Processes:                         GPU Memory |
|  GPU   GI    CI          PID    Type   Process name                  Usage |
|=====+=====+=====+=====+=====+=====+
|  0     N/A   N/A         356     C+G    ...crosoft Office\Office16\WINWORD.EXE      N/A |
|  0     N/A   N/A        1028     C+G    ...nt.CBS_cw5n1h2txyewy\SearchHost.exe      N/A |
|  0     N/A   N/A        3276     C+G    ...crosoft\Edge\Application\msedge.exe      N/A |
|  0     N/A   N/A        4260     C+G    ...CBS_cw5n1h2txyewy\TextInputHost.exe      N/A |
|  0     N/A   N/A        4896     C+G    ...ne\Binaries\Win64\EpicWebHelper.exe      N/A |
|  0     N/A   N/A        4940     C+G    ...pdnekdrzrea0\XboxGameBarSpotify.exe      N/A |
|  0     N/A   N/A       10096     C+G    ...oogle\Chrome\Application\chrome.exe      N/A |
|  0     N/A   N/A       10344     C+G    ...5n1h2txyewy\ShellExperienceHost.exe      N/A |
|  0     N/A   N/A       10528     C+G    ...on\128.0.2739.63\msedgewebview2.exe      N/A |
|=====+=====+=====+=====+=====+=====+
|
```

Bước 2: Cài đặt NVIDIA CUDA Toolkit

1. Tải xuống CUDA Toolkit:
- Truy cập trang [CUDA Toolkit](#) và tải xuống phiên bản phù hợp với hệ điều hành và GPU của bạn.

[Home](#) [Blog](#) [Forums](#) [Docs](#) [Downloads](#) [Training](#)

[Downloads](#) [Training](#) [Ecosystem](#) [Forums](#)

CUDA Toolkit

The NVIDIA® CUDA® Toolkit provides a development environment for creating high-performance, GPU-accelerated applications. With it, you can develop, optimize, and deploy your applications on GPU-accelerated embedded systems, desktop workstations, enterprise data centers, cloud-based platforms, and supercomputers. The toolkit includes GPU-accelerated libraries, debugging and optimization tools, a C/C++ compiler, and a runtime library.

Download Now

The Features of CUDA 12

Built-In Capabilities for Easy Scaling

Using built-in capabilities for distributing computations across multi-GPU configurations, you can develop applications that scale from single-GPU workstations to cloud installations with thousands of GPUs.

New Release, New Benefits

CUDA 12 introduces support for the NVIDIA Hopper™ and Ada Lovelace architectures, Arm® server processors, lazy module and kernel loading, revamped dynamic parallelism APIs, enhancements to the CUDA graphs API, performance-optimized libraries, and new developer tool capabilities.

Support for Hopper

Support for the Hopper architecture includes next-generation Tensor Cores and Transformer Engine, the high-speed NVIDIA NVLink® Switch, mixed-precision modes, second-generation Multi-Instance GPU (MIG), advanced memory management, and standard C++/Fortran/Python parallel language

CUDA Toolkit 12.6 Update 2 Downloads

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown. By downloading and using the software, you agree to fully comply with the terms and conditions of the [CUDA EULA](#)

Operating System	<input type="button" value="Linux"/> <input type="button" value="Windows"/>
Architecture	<input type="button" value="x86_64"/>
Version	<input type="button" value="10"/> <input type="button" value="11"/> <input type="button" value="Server 2022"/>
Installer Type	<input type="button" value="exe (local)"/> <input type="button" value="exe (network)"/>

Download Installer for Windows 11 x86_64

The base installer is available for download below.

> CUDA Toolkit Installer

Download (3.0 GB)

Installation Instructions:

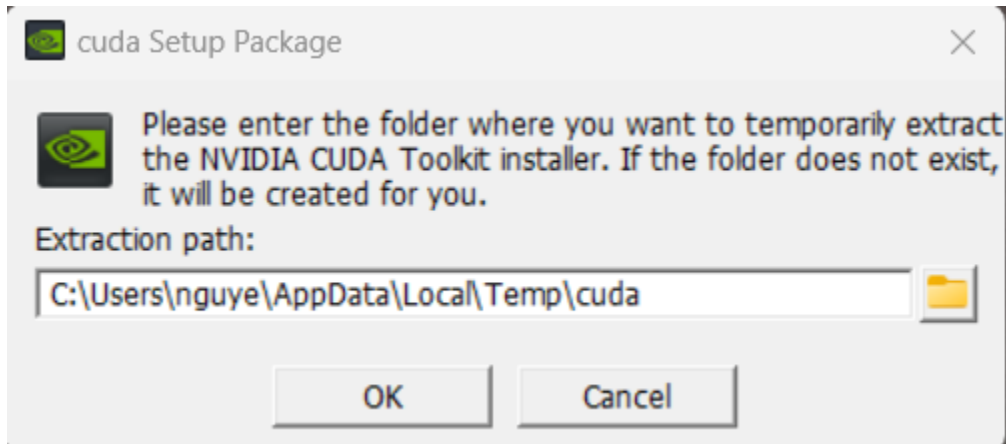
1. Double click cuda_12.6.2_560.94_windows.exe
2. Follow on-screen prompts

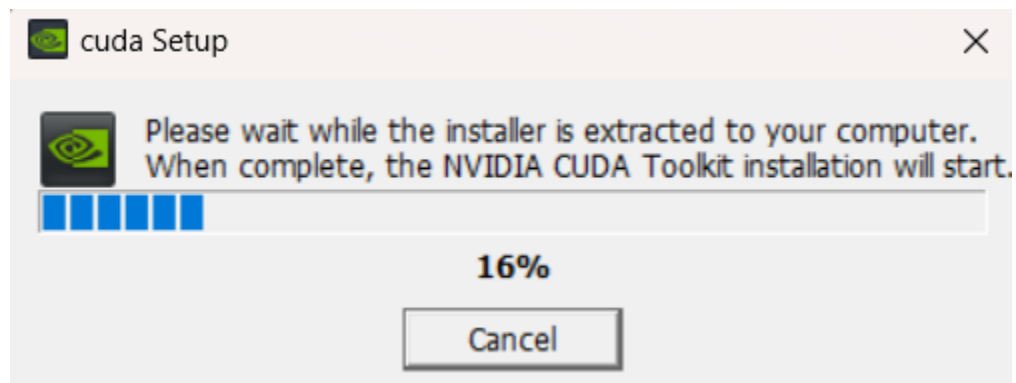
Additional installation options are detailed [here](#).

2. Cài đặt CUDA:

○ Windows:

- Chạy file .exe tải về và làm theo hướng dẫn.
- Trong bước **Installation Options**, chọn **Express (Recommended)**.





NVIDIA CUDA

Version 12.6



- ✓ System Check
- ✓ License Agreement

Options

Install

Finish

Installation options

☒ Express (Recommended)

Installs all CUDA components and overwrites current Display Driver.

☐ Custom (Advanced)

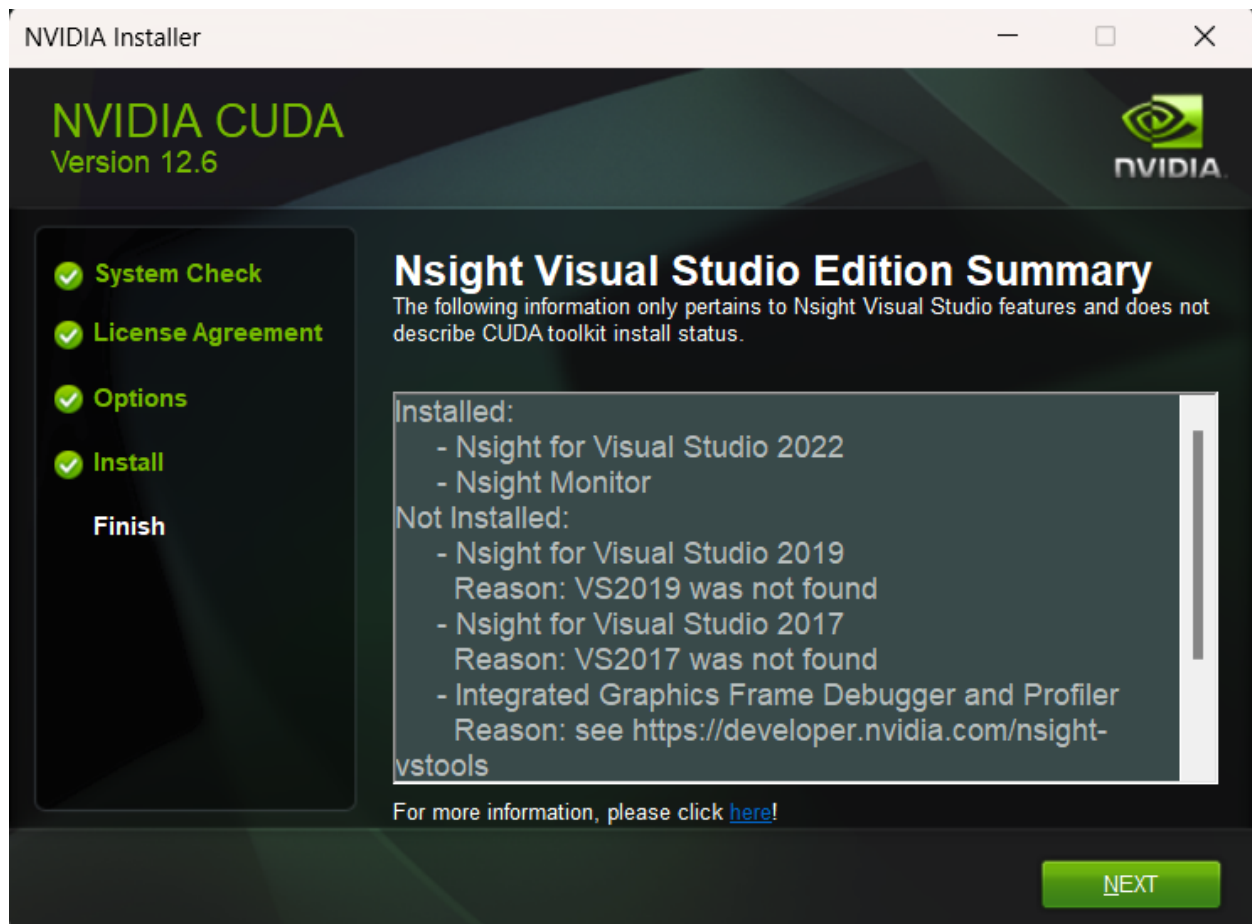
Allows you to select the components you want to install.

Note: Some flashing might occur during the installation.

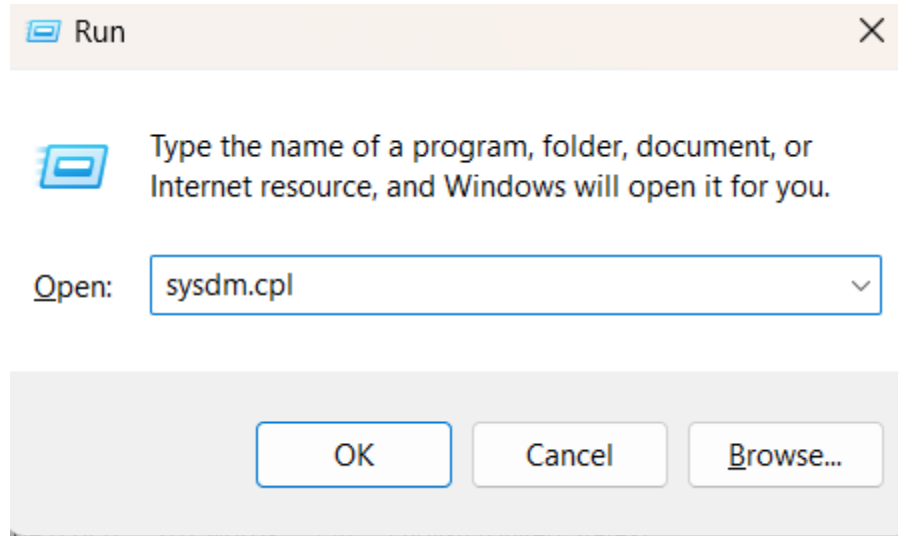
BACK

NEXT

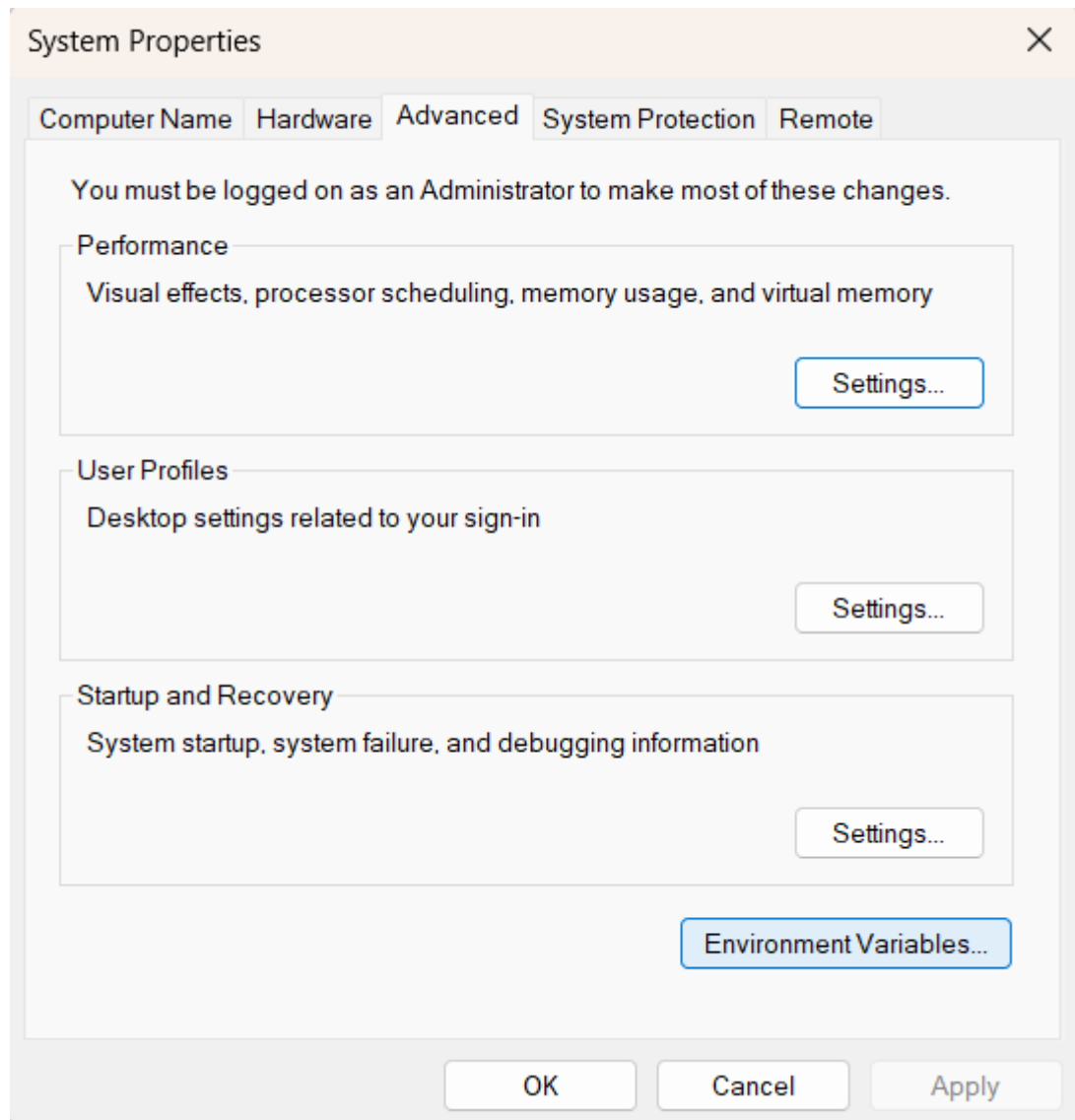
CANCEL



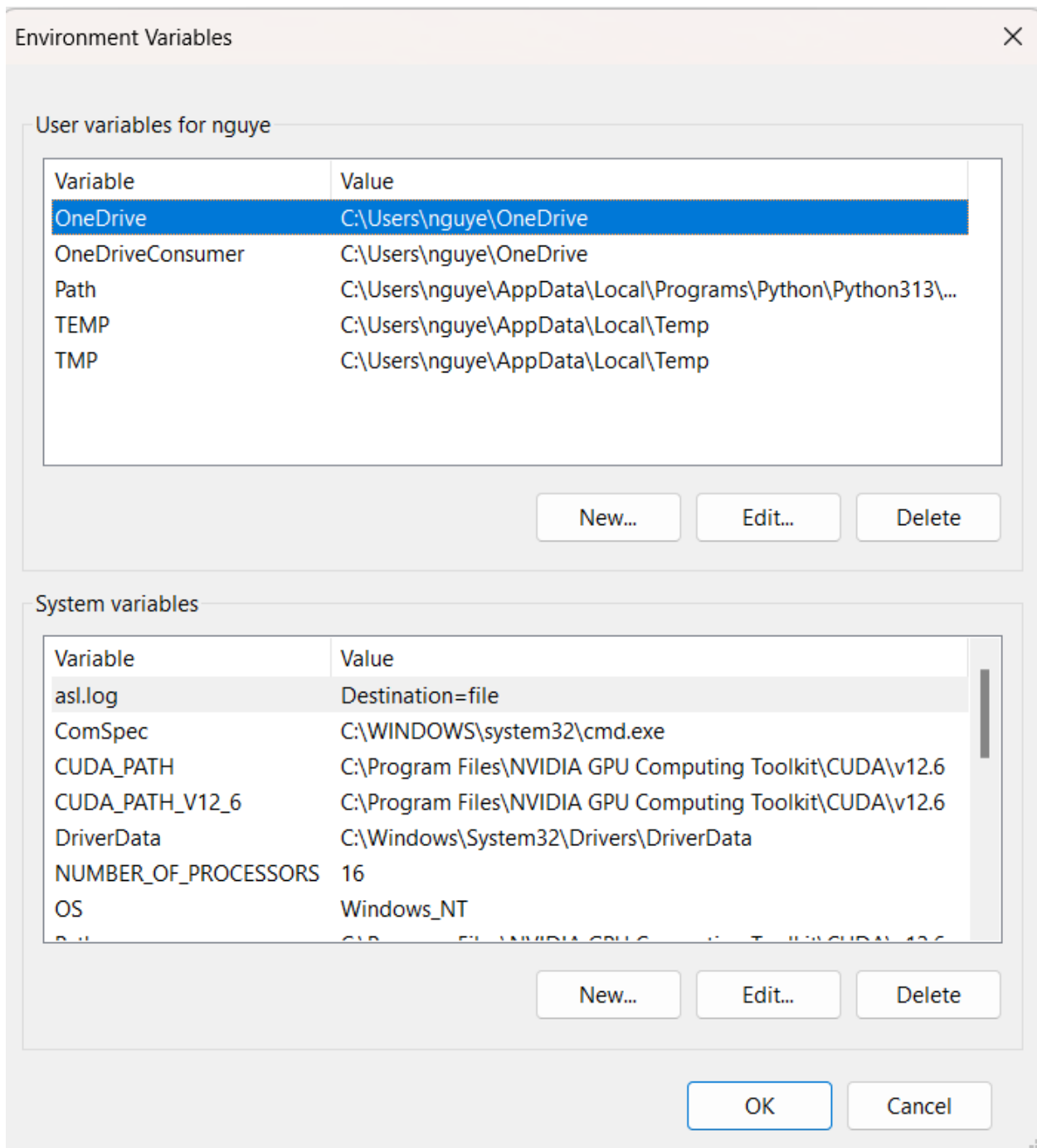
- **Linux:**
 - Cài đặt theo hướng dẫn trên trang [Linux Installation Guide](#).
- 3. **Kiểm tra cài đặt CUDA:**
 - Thêm CUDA vào PATH của hệ thống:
 - **Windows:** Thêm C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\vX.X(version)\bin vào biến môi trường Path.



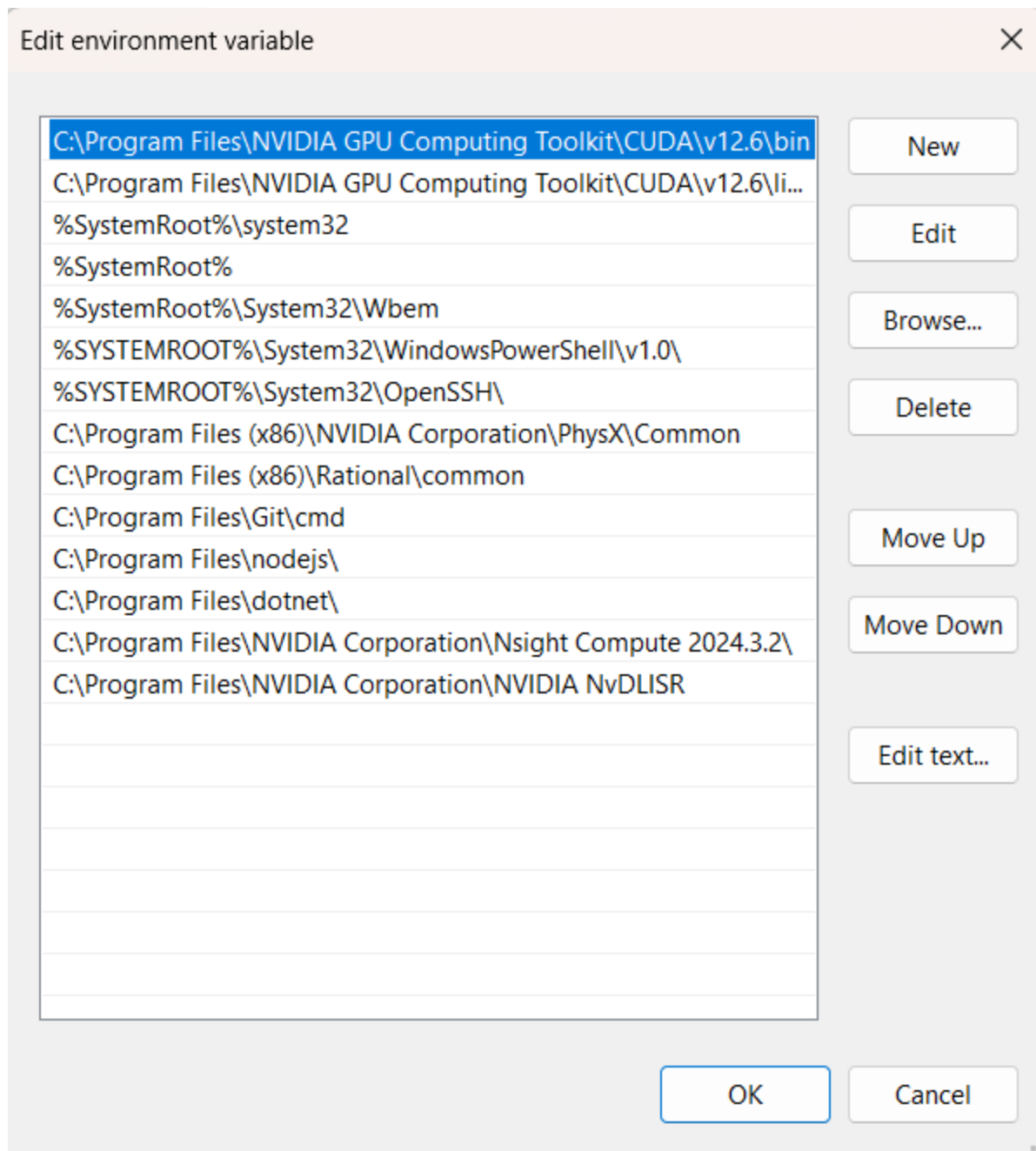
Chạy lệnh sysdm.cpl



Chọn Advanced-> Environment Variables



Trong system variables chọn Path->Edit



Chọn new thêm đường dẫn :

-VSCode 2022

“C:\ProgramFiles\MicrosoftVisualStudio\2022\Community\VC\Tools\MSVC\14.36.32532\bin\Hostx64\x64”

-Các phiên bản thấp hơn

"D:\Program Files\Microsoft Visual Studio 11.0\VC\bin"

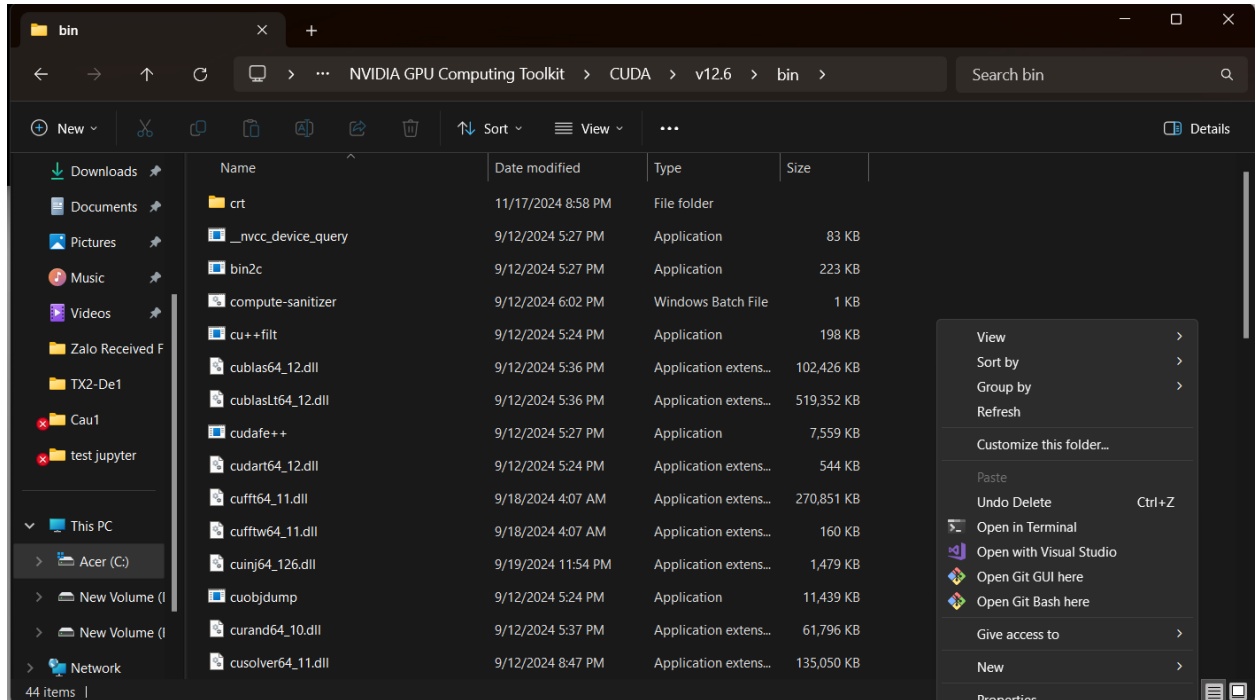
- **Linux:** Thêm dòng sau vào file ~/.bashrc:

```
export PATH=/usr/local/cuda-X.X/bin${PATH:+:${PATH}}
```

```
export
```

```
LD_LIBRARY_PATH=/usr/local/cudaX.X/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

- Kiểm tra bằng lệnh:
nvcc -version



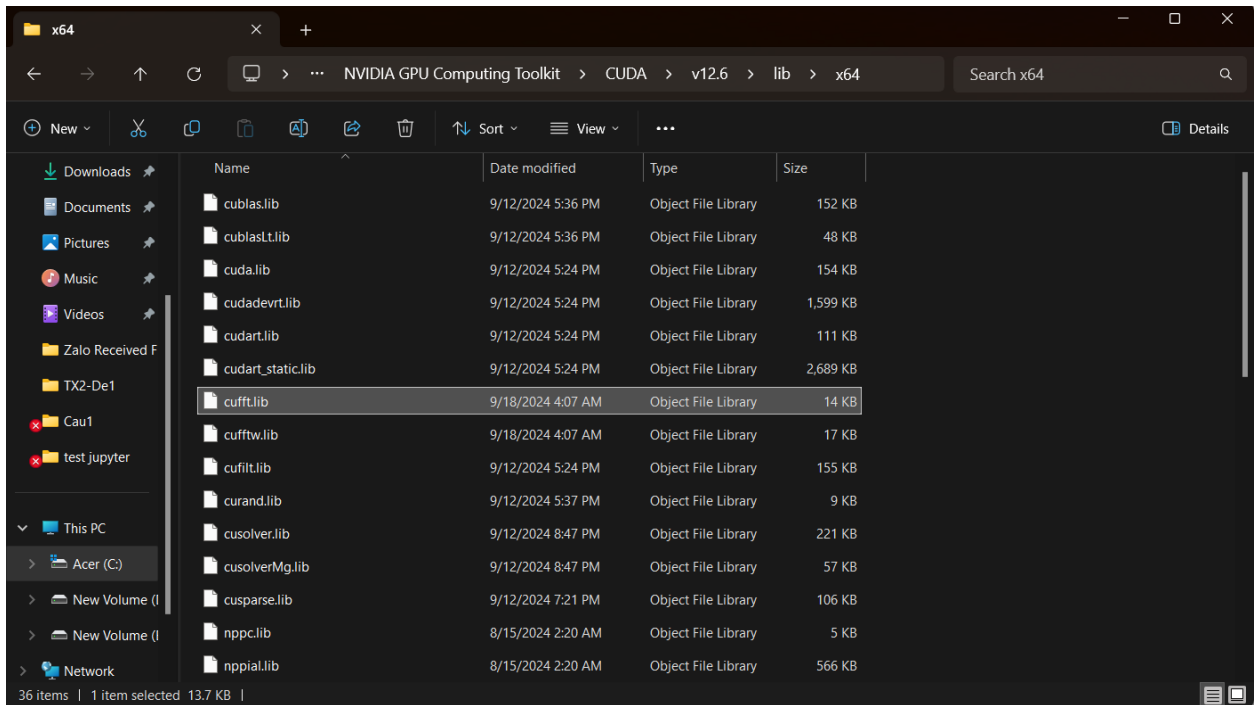
```
MINGW64:/c/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v12.6/bin
nguye@NDT MINGW64 /c/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v12.6/bin
$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2024 NVIDIA Corporation
Built on Thu_Sep_12_02:55:00_Pacific_Daylight_Time_2024
Cuda compilation tools, release 12.6, V12.6.77
Build cuda_12.6.r12.6/compiler.34841621_0

nguye@NDT MINGW64 /c/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v12.6/bin
$
```

Đầu ra hiển thị phiên bản CUDA Toolkit là đúng.

Bước 3. Cài đặt cuFFT

1. cuFFT là một phần của NVIDIA CUDA Toolkit. Khi cài đặt Toolkit, cuFFT đã được tích hợp sẵn trong thư mục lib.
2. Kiểm tra cuFFT:
 - Kiểm tra file cufft.lib (Windows) hoặc libcufft.so (Linux) có nằm trong thư mục:
 - **Windows:** C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\vX.X\lib\x64
 - **Linux:** /usr/local/cuda-X.X/lib64



Bước 4: Chạy chương trình mẫu với cuFFT trong Visual Studio Community 2022

4.1 . Tạo dự án trong Visual Studio

1. Tạo dự án mới:

- Mở Visual Studio 2022.
- Chọn **Create a new project > Console App > C++**.
- Đặt tên cho dự án (ví dụ: cuFFTEExample) và nhấn **Create**.

2. Cấu hình dự án:

- Click chuột phải vào dự án trong **Solution Explorer**, chọn **Properties**.
- Trong **Configuration Properties**, thực hiện các thay đổi sau:

a. Thêm thư mục include:

- Vào **VC++ Directories > Include Directories**.
- Thêm đường dẫn tới thư mục include của CUDA Toolkit:

C:\Program Files\NVIDIA GPU Computing
Toolkit\CUDA\vX.X\include

b. Thêm thư mục thư viện:

- Vào **VC++ Directories > Library Directories**.
- Thêm đường dẫn tới thư mục lib:

C:\Program Files\NVIDIA GPU Computing
Toolkit\CUDA\vX.X\lib\x64

c. Thêm link thư viện cuFFT:

- Vào **Linker > Input > Additional Dependencies**.
- Thêm: `cufft.lib, cuda.lib`

3. Thêm hỗ trợ file CUDA:

- Click chuột phải vào dự án > **Add > New Item > CUDA C++ File (.cu)**.
- Đặt tên file là `fft_example.cu`.

4.2. Viết chương trình mẫu

Copy đoạn mã sau vào file `fft_example.cu`:

```
#include <cufft.h>
```

```
#include <iostream>
```

```
int main() {
```

```
    int N = 256; // Kích thước FFT
```

```
    cufftComplex* data;
```

```
    cudaMallocManaged(&data, sizeof(cufftComplex) * N);
```

```
    // Khởi tạo dữ liệu
```

```
    for (int i = 0; i < N; ++i) {
```

```
        data[i].x = i; // Phần thực
```

```
        data[i].y = 0; // Phần ảo
```

```
    }
```

```
    // Tạo kế hoạch FFT
```

```
    cufftHandle plan;
```

```
    cufftPlan1d(&plan, N, CUFFT_C2C, 1);
```

```

// Thực thi FFT
cufftExecC2C(plan, data, data, CUFFT_FORWARD);

// Xuất kết quả
std::cout << "FFT Result:\n";
for (int i = 0; i < N; ++i) {
    std::cout << "Result[" << i << "] = (" << data[i].x << ", " << data[i].y << ")\n";
}

// Dọn dẹp
cufftDestroy(plan);
cudaFree(data);

return 0;
}

```

4.3. Cấu hình build

1. Chuyển **Solution Configuration** sang **Release** (hoặc **Debug** nếu muốn gỡ lỗi).
2. **Kiểm tra công cụ build CUDA:**
 - Đảm bảo CUDA Toolkit đã được tích hợp vào Visual Studio khi cài đặt.
 - Nếu cần, mở **Properties > CUDA C/C++ > Common** và đảm bảo:
 - CUDA Toolkit Custom Dir trỏ tới thư mục CUDA Toolkit.

4.4. Build và chạy chương trình

1. Click **Build > Build Solution** (hoặc nhấn Ctrl + Shift + B).
 - Đảm bảo không có lỗi trong quá trình biên dịch.
2. Chạy chương trình:
 - Click **Debug > Start Without Debugging** (hoặc nhấn Ctrl + F5).
 - Kết quả FFT sẽ được in ra console.

4.5. Xử lý lỗi thường gặp

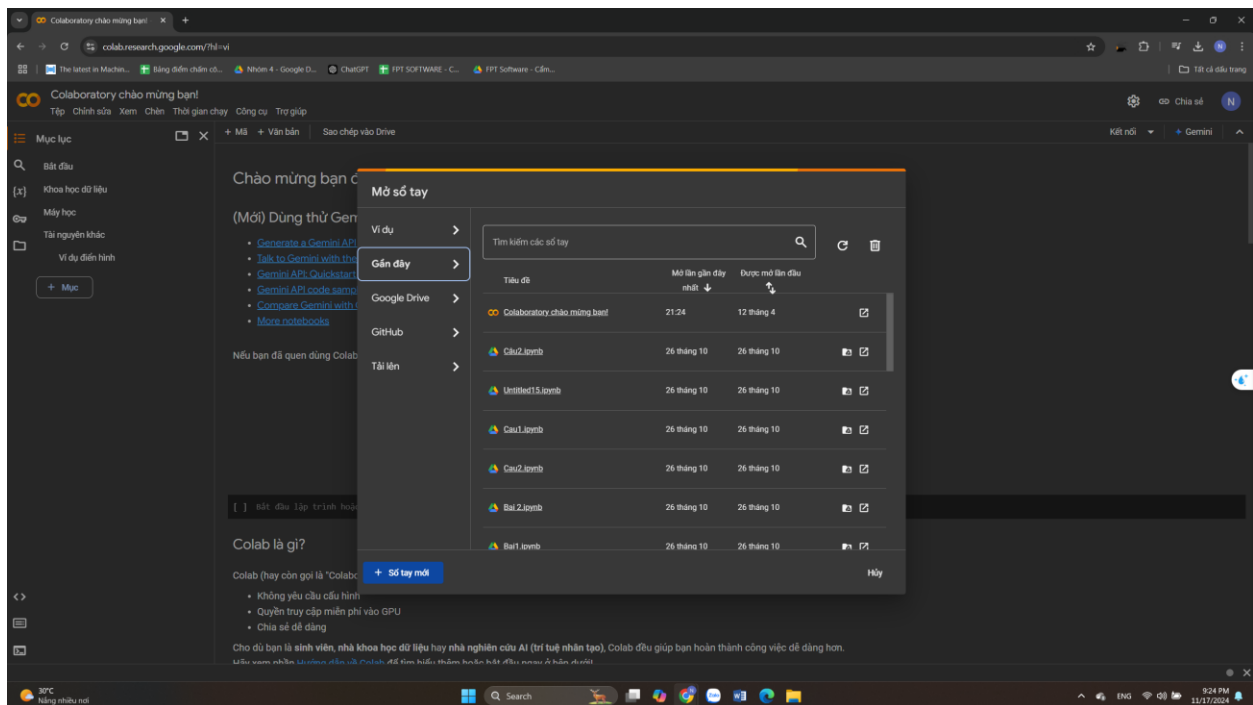
1. **nvcc: command not found hoặc nvcc không được nhận diện:**
 - Đảm bảo bạn đã thêm CUDA Toolkit vào biến môi trường PATH.
2. **Lỗi cufft.lib hoặc cuda.lib không tìm thấy:**

- Kiểm tra lại đường dẫn thư viện trong **VC++ Directories > Library Directories**.
3. **Lỗi runtime về driver:**
- Đảm bảo driver NVIDIA đã được cập nhật lên phiên bản mới nhất tương thích với CUDA Toolkit.

2. Cài đặt CUDA và cuFFT trên Google Colab

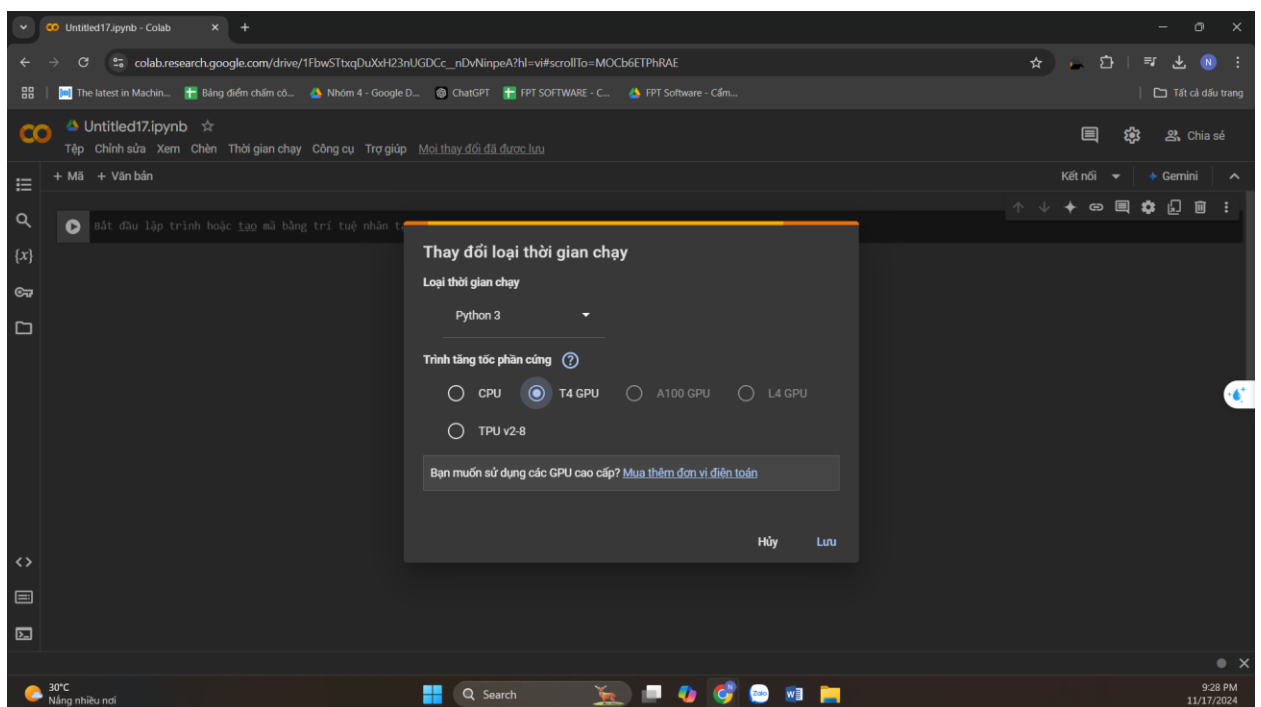
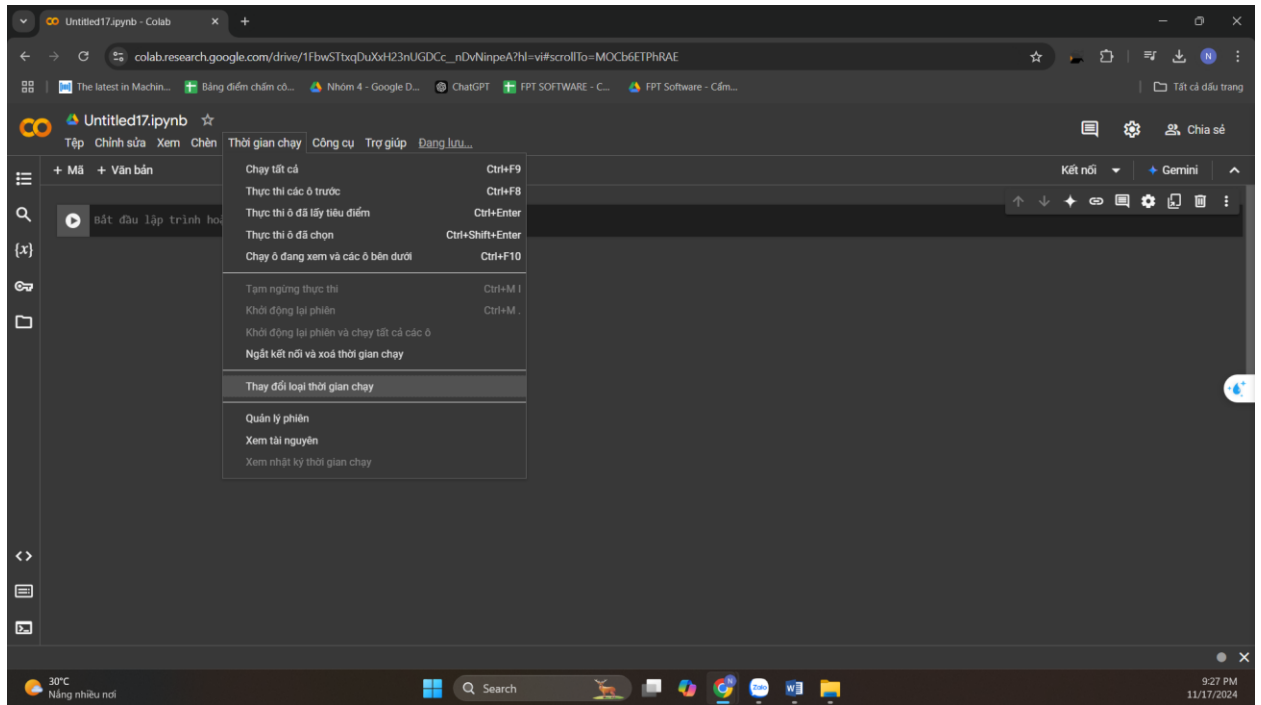
Bước 1: Tạo mới notebook để sử dụng

Truy cập vào trang web: <https://colab.research.google.com/> trong trình duyệt và nhấp chuột vào New notebook.



Bước 2: Cấu hình GPU:

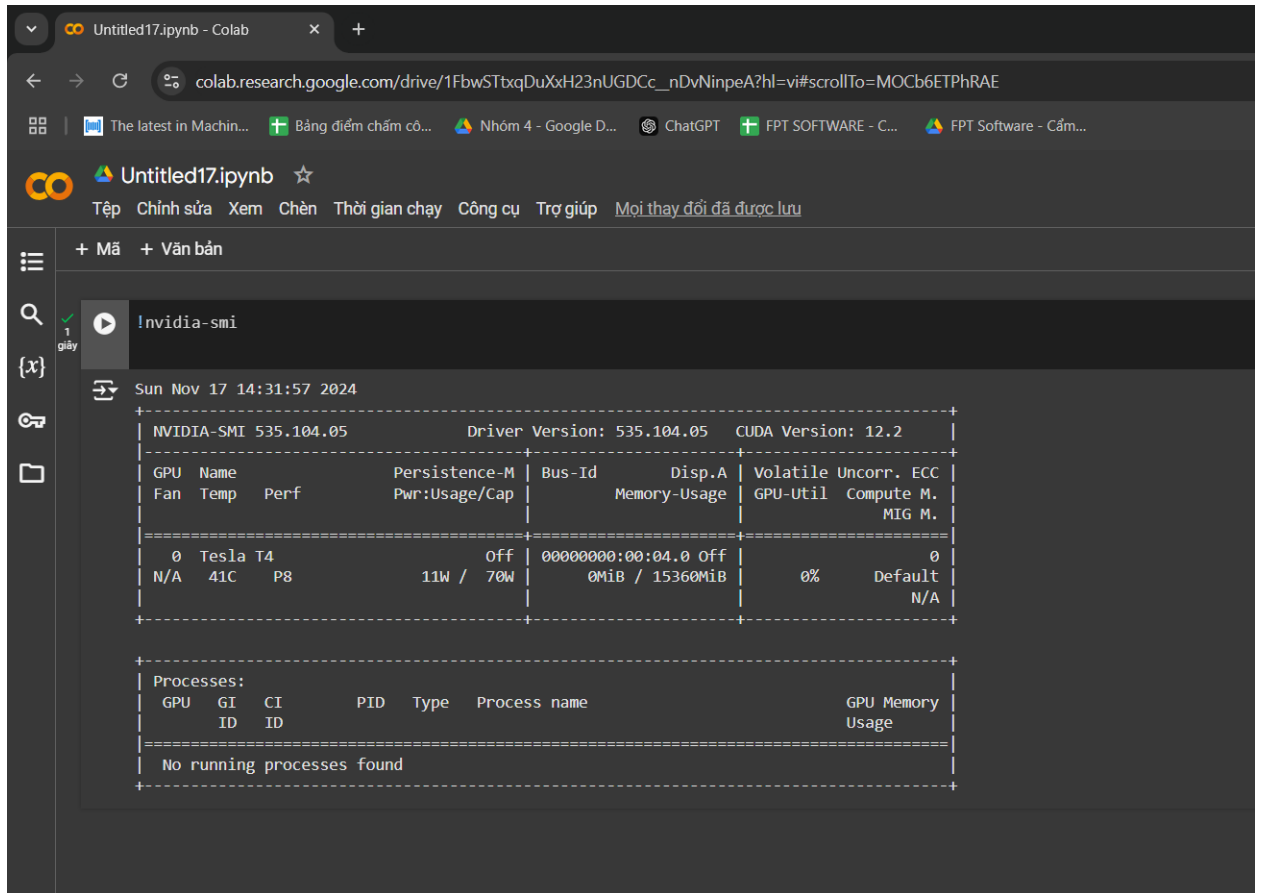
- Vào **Runtime > Change runtime type**.
- Tại mục **Hardware accelerator**, chọn **T4GPU**.
- Nhấn **Save**, sau đó nhấn **Connect** để kết nối.



Bước 3: Kiểm tra GPU và phiên bản CUDA:

- Chạy lệnh: `!nvidia-smi`

Lệnh này hiển thị thông tin về GPU đang sử dụng.



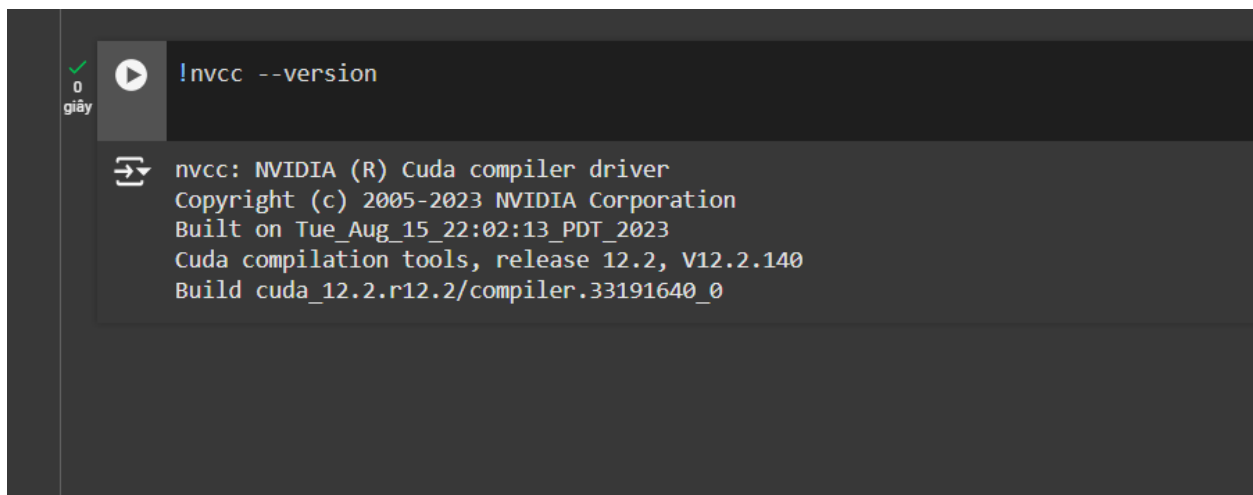
```
!nvidia-smi
```

Sun Nov 17 14:31:57 2024

NVIDIA-SMI 535.104.05			Driver Version: 535.104.05		CUDA Version: 12.2	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile Uncorr. ECC	
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M. MIG M.
0	Tesla T4	Off	00000000:00:04.0	Off	0%	0
N/A	41C	P8	11W / 70W	0MiB / 15360MiB		Default N/A

Processes:						
GPU	GI	CI	PID	Type	Process name	GPU Memory Usage
ID	ID	ID				
No running processes found						

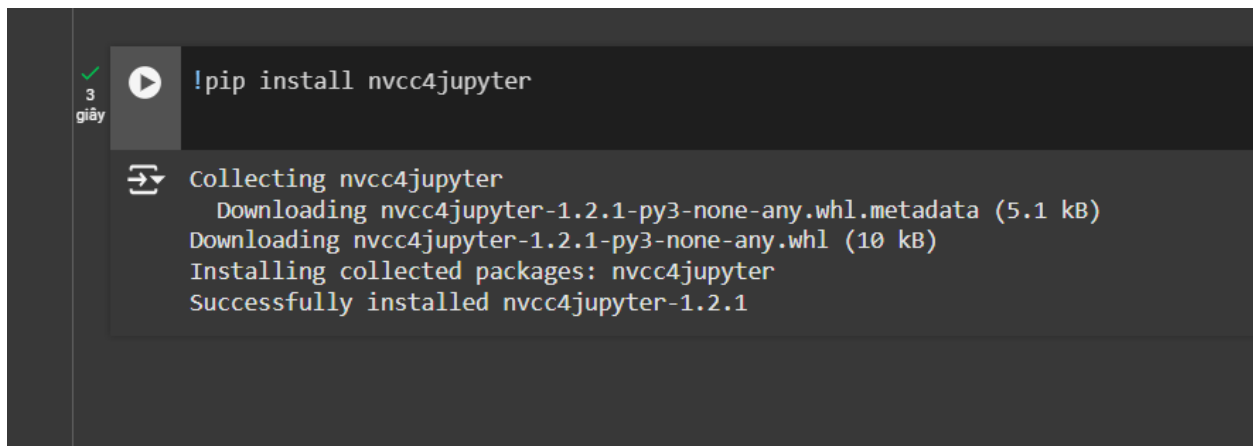
- Kiểm tra phiên bản CUDA: `!nvcc --version`



```
!nvcc --version
```

nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Tue_Aug_15_22:02:13_PDT_2023
Cuda compilation tools, release 12.2, V12.2.140
Build cuda_12.2.r12.2/compiler.33191640_0

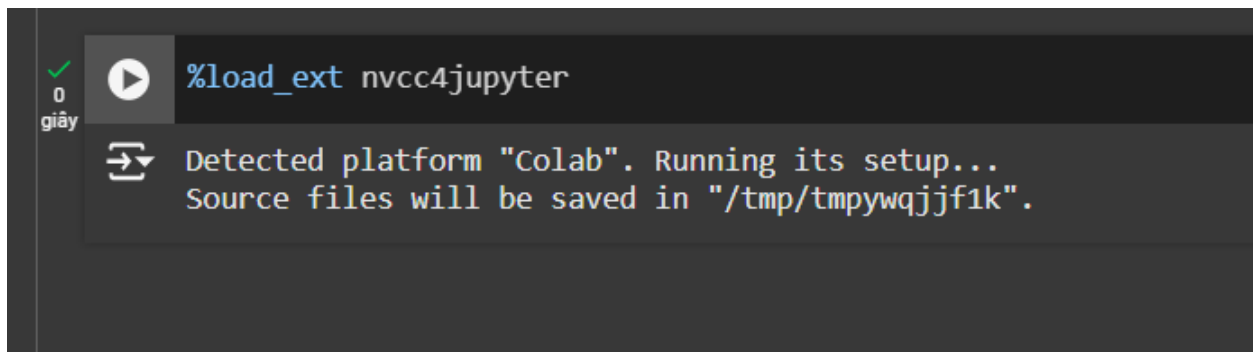
- Cài đặt thư viện nvcc4jupyter để hỗ trợ chạy mã CUDA trong notebook: `!pip install nvcc4jupyter`



A terminal window with a dark background. On the left, a green checkmark and the text '3 giây' are visible. The command '!pip install nvcc4jupyter' is entered. The output shows the collection and installation of nvcc4jupyter-1.2.1, including downloading metadata and the wheel file.

```
✓ 3 giây !pip install nvcc4jupyter  
Collecting nvcc4jupyter  
  Downloading nvcc4jupyter-1.2.1-py3-none-any.whl.metadata (5.1 kB)  
  Downloading nvcc4jupyter-1.2.1-py3-none-any.whl (10 kB)  
Installing collected packages: nvcc4jupyter  
Successfully installed nvcc4jupyter-1.2.1
```

- Tải tiện ích mở rộng để chạy mã CUDA: %load_ext nvcc4jupyter



A terminal window with a dark background. On the left, a green checkmark and the text '0 giây' are visible. The command '%load_ext nvcc4jupyter' is entered. The output indicates that the platform 'Colab' was detected and its setup is running, with source files saved in a temporary directory.

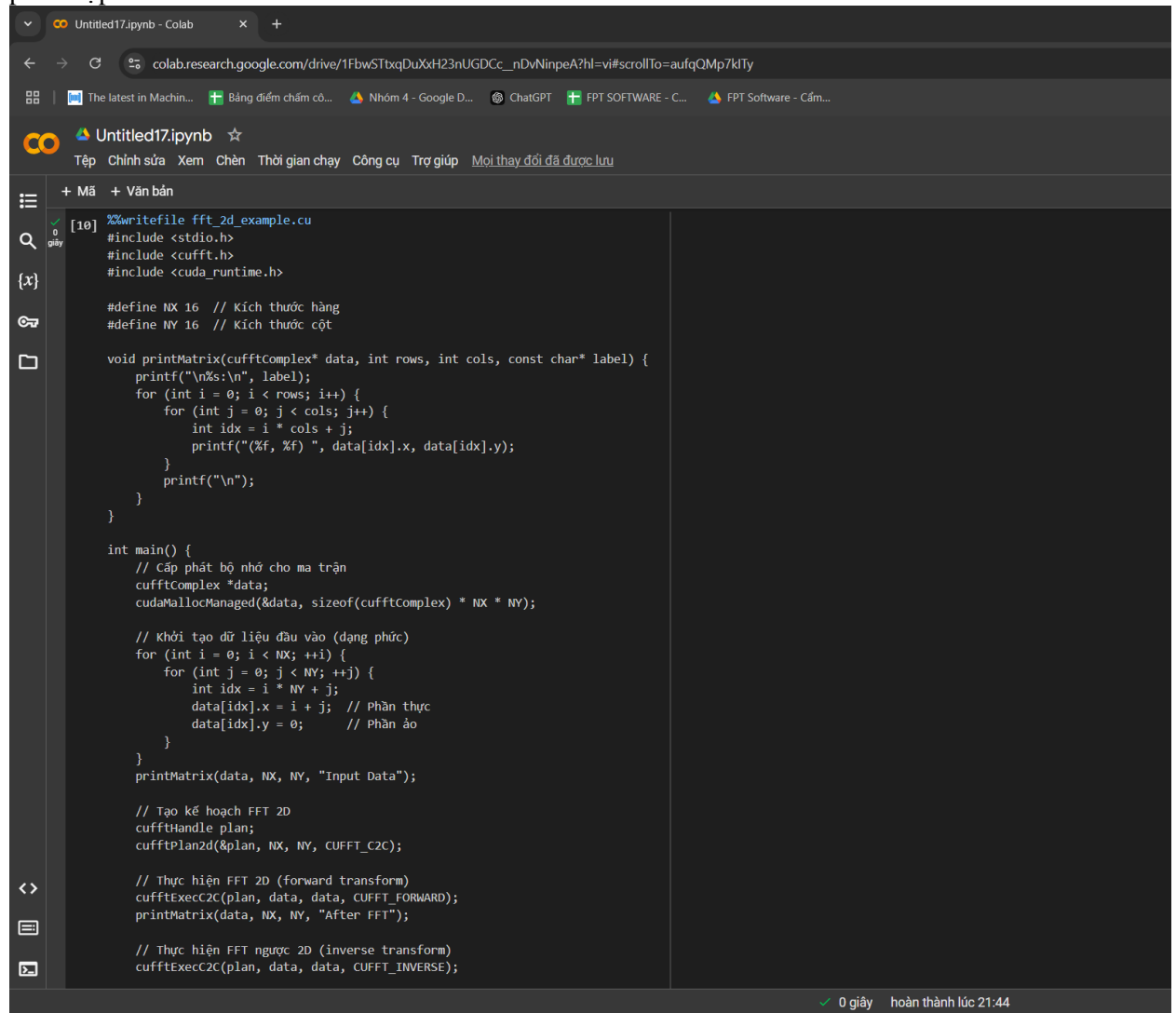
```
✓ 0 giây %load_ext nvcc4jupyter  
Detected platform "Colab". Running its setup...  
Source files will be saved in "/tmp/tmpywqjjf1k".
```

Bước 4: Chạy chương trình ví dụ sử dụng CUDA và cuFFT

1. Chạy mã CUDA bằng tiện ích nvcc4jupyter:

- Trong Google Colab, việc chạy mã CUDA yêu cầu sử dụng nvcc để biên dịch, nhưng cú pháp %%cuda không hỗ trợ trực tiếp liên kết thư viện ngoài, như -lcufft. Do đó, cần tạo file mã nguồn riêng (dùng %%writefile) để biên dịch với các cờ liên kết đầy đủ, đảm bảo các hàm từ thư viện cuFFT được nhận diện đúng trong quá trình liên kết và thực thi chương trình.

Ví dụ: Sử dụng thư viện cuFFT để thực hiện **2D FFT** (Fast Fourier Transform) trên một ma trận phức hợp



The screenshot shows a Google Colab notebook interface. The top bar indicates the notebook is titled 'Untitled17.ipynb'. The browser address bar shows the URL: colab.research.google.com/drive/1FbwSTbxqDuXxH23nUGDCc_nDvNinpeA?hl=vi#scrollTo=aufqQMp7kITy. The notebook has a sidebar with icons for file explorer, search, and other functions. The main area displays a C++ code cell with the following content:

```
[10] %writefile fft_2d_example.cu
#include <stdio.h>
#include <cuFFT.h>
#include <cuda_runtime.h>

#define NX 16 // Kích thước hàng
#define NY 16 // Kích thước cột

void printMatrix(cuFFTComplex* data, int rows, int cols, const char* label) {
    printf("\n%s:\n", label);
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            int idx = i * cols + j;
            printf("(%.2f, %.2f) ", data[idx].x, data[idx].y);
        }
        printf("\n");
    }
}

int main() {
    // Cấp phát bộ nhớ cho ma trận
    cuFFTComplex *data;
    cudaMallocManaged(&data, sizeof(cuFFTComplex) * NX * NY);

    // Khởi tạo dữ liệu đầu vào (dạng phức)
    for (int i = 0; i < NX; ++i) {
        for (int j = 0; j < NY; ++j) {
            int idx = i * NY + j;
            data[idx].x = i + j; // Phần thực
            data[idx].y = 0;     // Phần ảo
        }
    }
    printMatrix(data, NX, NY, "Input Data");

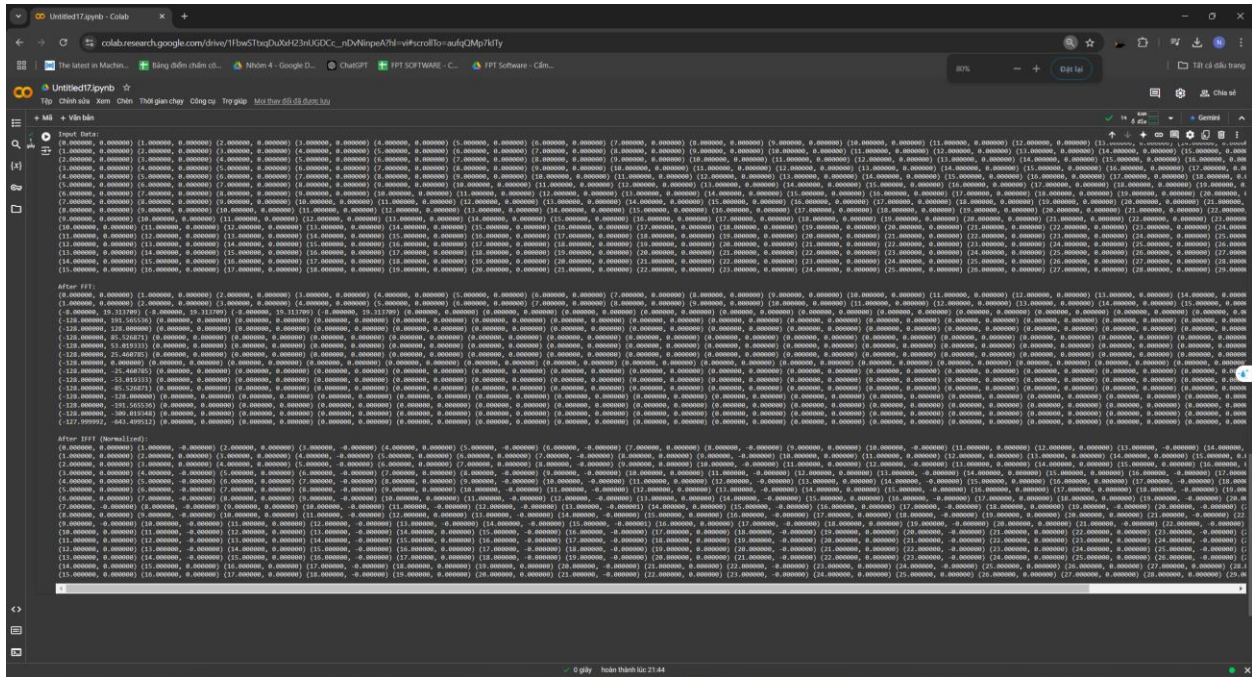
    // Tạo kế hoạch FFT 2D
    cuFFTHandle plan;
    cuFFTPlan2d(&plan, NX, NY, CUFFT_C2C);

    // Thực hiện FFT 2D (forward transform)
    cuFFTExecC2C(plan, data, data, CUFFT_FORWARD);
    printMatrix(data, NX, NY, "After FFT");

    // Thực hiện FFT ngược 2D (inverse transform)
    cuFFTExecC2C(plan, data, data, CUFFT_INVERSE);
```

The bottom status bar shows a green checkmark, '0 giây' (0 seconds), and 'hoàn thành lúc 21:44' (completed at 21:44).

Kết quả:



Phần 2: Lý thuyết cơ bản trong cuFFT

1. Thành phần chính của cuFFT

cuFFT là thư viện NVIDIA hỗ trợ thực hiện các phép biến đổi Fourier nhanh (FFT) trên GPU. Các thành phần chính của cuFFT bao gồm:

a. Kiểu dữ liệu

- Dữ liệu đầu vào/đầu ra: Hỗ trợ các loại dữ liệu phức (complex) và thực (real).
- `cufftComplex` (số phức dạng `float`): Gồm phần thực và phần ảo.
- `cufftDoubleComplex` (số phức dạng `double`).
- `cufftReal` và `cufftDoubleReal`: Dữ liệu thực.

b. Kế hoạch (Plan)

- Kế hoạch FFT: Là cấu trúc định nghĩa cách thực hiện phép biến đổi FFT trên dữ liệu. Kế hoạch này bao gồm kích thước, loại biến đổi và cách ánh xạ dữ liệu GPU.

c. Biến đổi FFT

- Biến đổi FFT Forward (miền thời gian \rightarrow miền tần số).
- Biến đổi FFT Inverse (miền tần số \rightarrow miền thời gian).

2. Các hàm cơ bản trong cuFFT

a. Khởi tạo kế hoạch FFT

1. cufftPlan1d

Tạo kế hoạch cho FFT 1D:

cufftHandle plan;

cufftPlan1d(&plan, nx, CUFFT_C2C, batch)

- `nx`: Số điểm trong FFT.
- `CUFFT_C2C`: Loại biến đổi (Complex-to-Complex).
- `batch`: Số lượng FFT được thực hiện song song.

2. cufftPlan2d và cufftPlan3d

Tạo kế hoạch cho FFT 2D và 3D:

cufftPlan2d(&plan, nx, ny, CUFFT_C2C);

cufftPlan3d(&plan, nx, ny, nz, CUFFT_C2C);

3. cufftPlanMany

Tạo kế hoạch để thực hiện nhiều FFT cùng lúc trên dữ liệu đa chiều hoặc batch:

cufftPlanMany(&plan, rank, n, inembed, istride, idist, onembed, ostride, odist, CUFFT_C2C, batch);

- `rank`: Số chiều của FFT.
- `n`: Mảng chứa kích thước FFT từng chiều.
- `inembed`/'`onembed`': Kích thước mảng dữ liệu đầu vào/đầu ra.

b. Thực hiện FFT

1. FFT Complex-to-Complex (C2C):

cufftExecC2C(plan, input, output, CUFFT_FORWARD);

- CUFFT_FORWARD: FFT chuyển tiếp.
- CUFFT_INVERSE: FFT nghịch.

2. FFT Real-to-Complex (R2C):

cufftExecR2C(plan, input, output);

3. FFT Complex-to-Real (C2R):

cufftExecC2R(plan, input, output);

c. Dọn dẹp

- cufftDestroy: Hủy kế hoạch sau khi sử dụng:

cufftDestroy(plan);

d. Quản lý lỗi

- Các hàm cuFFT trả về mã lỗi. Sử dụng `cufftResult` để kiểm tra, ví dụ:
if (cufftPlan1d(&plan, nx, CUFFT_C2C, batch) != CUFFT_SUCCESS) {
 printf("Failed to create plan\n");
}

3. Quy trình cơ bản khi dùng cuFFT

1. Cấp phát bộ nhớ GPU: Dùng `cudaMalloc` hoặc `cudaMallocManaged`.
2. Tạo kế hoạch FFT: Sử dụng `cufftPlan1d`, `cufftPlan2d`, hoặc `cufftPlan3d`.
3. Thực hiện phép biến đổi: Sử dụng các hàm `cufftExec` phù hợp.
4. Giải phóng tài nguyên: Gọi `cudaFree` và `cufftDestroy`.

4. Ứng dụng thực tế

cuFFT thường được dùng trong:

- Xử lý tín hiệu (biến đổi tín hiệu âm thanh, hình ảnh).
- Phân tích miền tần số trong radar hoặc ảnh y tế.
- Mô phỏng khoa học (ví dụ: giải phương trình vật lý miền tần số).

Phần 3: Lý thuyết về GPU

1. **Grid**: Là tập hợp của nhiều block. Mỗi grid chứa nhiều block, giúp GPU có thể thực thi song song các tác vụ. Grid được chia thành các block và các block này lại chia thành các thread.
2. **Block**: Là nhóm các thread cùng thực thi một tác vụ. Mỗi block có thể chứa tối đa 1024 thread và mỗi thread trong block có thể truy cập bộ nhớ chia sẻ (shared memory). Mỗi block được thực thi trên một multiprocessor (SM) của GPU.
3. **Warp**: Là nhóm các thread (thường là 32 thread) trong một block được xử lý đồng thời trên một SM. Các thread trong một warp luôn thực hiện cùng một lệnh tại cùng một thời điểm.
4. **Thread**: Là đơn vị tính toán cơ bản trong CUDA. Mỗi thread thực hiện một tác vụ riêng biệt, và tất cả các thread trong một block chia sẻ một bộ nhớ cục bộ. Các thread được lập trình viên sử dụng để xử lý song song.
5. **SM (Streaming Multiprocessor)**: Là đơn vị xử lý trong GPU, có khả năng thực thi nhiều thread song song. Mỗi SM chứa một số lượng thread và thực thi các warp.
6. **SMTM (Streaming Multiprocessor Thread Manager)**: Quản lý và phân phối các thread tới các SM trong GPU, đảm bảo rằng các thread được phân bổ hợp lý và hiệu quả trong quá trình xử lý.
7. **Kiến trúc bộ nhớ phân cấp**:
 - o **Global Memory**: Bộ nhớ chính của GPU, nhưng có độ trễ cao. Mọi thread có thể truy cập.
 - o **Shared Memory**: Bộ nhớ chia sẻ giữa các thread trong cùng một block. Có tốc độ truy xuất nhanh hơn so với global memory.
 - o **Local Memory**: Bộ nhớ cục bộ dành riêng cho mỗi thread, thường được sử dụng khi không có đủ register.

- **Constant and Texture Memory:** Các loại bộ nhớ đặc biệt trong GPU, thường dùng cho dữ liệu không thay đổi hoặc dữ liệu hình ảnh.
- 8. **Host-Device:**
 - **Host** là CPU, nơi điều phối chương trình.
 - **Device** là GPU, nơi thực thi các tính toán song song.
 - Dữ liệu cần được chuyển từ host (CPU) sang device (GPU) và ngược lại để thực thi các tác vụ trên GPU.
- 9. **Cấu trúc chương trình CUDA:**
 - Một chương trình CUDA bao gồm các kernel, là các hàm được thực thi song song trên GPU.
 - Các kernel được khai báo với từ khóa `__global__` và được gọi từ CPU (host).
 - Các thread được nhóm thành blocks, và blocks được nhóm thành grid.
- 10. **Hàm gửi host -> device và ngược lại:**
 - **Hàm gửi từ host sang device:**
 - `cudaMemcpy(destination, source, size, cudaMemcpyHostToDevice):` Chuyển dữ liệu từ bộ nhớ của CPU sang GPU.
 - **Hàm gửi từ device về host:**
 - `cudaMemcpy(destination, source, size, cudaMemcpyDeviceToHost):` Chuyển dữ liệu từ bộ nhớ GPU về bộ nhớ của CPU.

Phần 4: Các bài toán mẫu sử dụng cuFFT

Bài toán 1: Biến đổi Fourier nhanh (FFT) 1D trên một tín hiệu

Bài toán 2: Biến đổi Fourier nhanh 2D trên ảnh

Bài toán 3: Biến đổi Fourier nhanh 3D

Bài toán 4 : Biến đổi Fourier nhanh và lọc tín hiệu

Mô tả: Thực hiện FFT trên một tín hiệu, áp dụng lọc tần số (cắt tần số cao) và thực hiện biến đổi ngược (IFFT).

Bài toán 5 : Biến đổi Fourier nhanh để phân tích tần số tín hiệu

Link : <https://github.com/NguyenDuyThai1307/cuFFT-example>