GPU Information Report Using Numba

September 28, 2025

1 Introduction

This report presents the results of querying GPU specifications using a Python script leveraging the Numba library's CUDA module. The script retrieves details such as the GPU name, number of streaming multiprocessors (SMs), total CUDA cores, and memory information for available NVIDIA GPUs. The output is analyzed to provide insights into the GPU's capabilities.

2 Python Script

The following Python script uses Numba's CUDA API to detect and display information about available GPUs. It includes a dictionary mapping CUDA compute capabilities to cores per streaming multiprocessor (SM) and calculates the total number of CUDA cores.

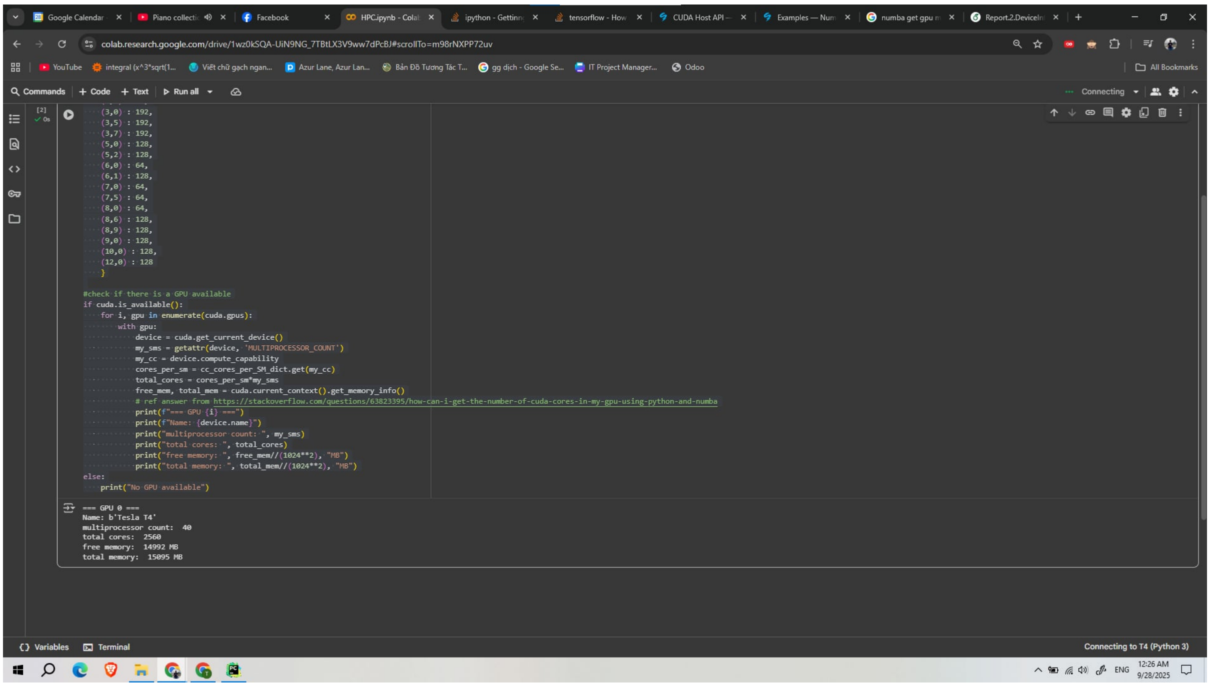
```python
from numba import cuda

cc_cores_per_SM_dict = {
    (2,0) : 32,
    (2,1) : 48,
    (3,0) : 192,
    (3,5) : 192,
    (3,7) : 192,
    (5,0) : 128,
    (5,2) : 128,
    (6,0) : 64,
    (6,1) : 128,
    (7,0) : 64,
    (7,5) : 64,
    (8,0) : 64,
    (8,6) : 128,
    (8,9) : 128,
    (9,0) : 128,
    (10,0) : 128,
    (12,0) : 128
}

# Check if there is a GPU available
if cuda.is_available():
    for i, gpu in enumerate(cuda.gpus):
        with gpu:
            device = cuda.get_current_device()
            my_sms = getattr(device, 'MULTIPROCESSOR_COUNT')
            my_cc = device.compute_capability
            cores_per_sm = cc_cores_per_SM_dict.get(my_cc)
            total_cores = cores_per_sm*my_sms
            free_mem, total_mem = cuda.current_context().get_memory_info()

            print(f"=== GPU {i} ===")
            print(f"Name: {device.name}")
            print("multiprocessor count: ", my_sms)
            print("total cores: ", total_cores)
            print("free memory: ", free_mem//(1024**2), "MB")
            print("total memory: ", total_mem//(1024**2), "MB")
else:
    print("No GPU available")
```

Listing 1: Python script to query GPU information

3 Output

The script was executed on a system with an Tesla T4 GPU. The output is as follows:

```
=== GPU 0 ===
Name: b'Tesla T4'
multiprocessor count:  40
total cores:  2560
free memory:  14992 MB
total memory:  15095 MB
```



colab.research.google.com/drive/1wz0kSQA-UiN9NG_7TBtLX3V9ww7dPcBJ#scrollTo=m98rNXPP72uv
Commands
+ Code
+ Text
Run all
Connecting
Variables
Terminal
Connecting to T4 (Python 3)

```
    (3,0) : 192,
    (3,5) : 192,
    (3,7) : 192,
    (5,0) : 128,
    (5,2) : 128,
    (6,0) : 64,
    (6,1) : 128,
    (7,0) : 64,
    (7,5) : 64,
    (8,0) : 64,
    (8,6) : 128,
    (8,9) : 128,
    (9,0) : 128,
    (10,0) : 128,
    (12,0) : 128
    }

#check if there is a GPU available
if cuda.is_available():
    for i, gpu in enumerate(cuda.gpus):
        with gpu:
            device = cuda.get_current_device()
            my_sms = getattr(device, 'MULTIPROCESSOR_COUNT')
            my_cc = device.compute_capability
            cores_per_sm = cc_cores_per_SM_dict.get(my_cc)
            total_cores = cores_per_sm*my_sms
            free_mem, total_mem = cuda.current_context().get_memory_info()
            # ref answer from https://stackoverflow.com/questions/63823395/how-can-i-get-the-number-of-cuda-cores-in-my-gpu-using-python-and-numba
            print(f"=== GPU {i} ===")
            print(f"Name: {device.name}")
            print("multiprocessor count: ", my_sms)
            print("total cores: ", total_cores)
            print("free memory: ", free_mem//(1024**2), "MB")
            print("total memory: ", total_mem//(1024**2), "MB")
else:
    print("No GPU available")
```

```
=== GPU 0 ===
Name: b'Tesla T4'
multiprocessor count:  40
total cores:  2560
free memory:  14992 MB
total memory:  15095 MB
```

Figure 1: Screenshot of the script's output.

4 Conclusion

The Python script successfully retrieves and displays critical GPU specifications using Numba's CUDA module. The Tesla T4 has 40 streaming multiprocessors, 2560 total cores and 15095 MB total memory.