GPU Information Report - Labwork 2

Le Minh Hoang

Keywords:

Abstract. No abstract for the labwork 2, I guess.

1 Introduction

This report states the work completed in Labwork 2, which focuses on GPU communication using Numba.

This report consists of the following sections:

- GPU device name
- Multiprocessor core count
- Memory configuration

2 Results and Analysis

2.1 GPU Device Information

The GPU information can be easily acquired using the `cuda.detect()` function from Numba CUDA. The result are returned as following

```
Found 1 CUDA devices
id 0             b'Tesla T4'                              [
   SUPPORTED]
                      Compute Capability: 7.5
                           PCI Device ID: 4
                              PCI Bus ID: 0
                                    UUID: GPU-1376e51e-3a62-e0b5-
                                       df6b-acec1af4496f
                                Watchdog: Disabled
             FP32/FP64 Performance Ratio: 32
Summary:
  1/1 devices are supported
```

The detected device is a Tesla T4 GPU, which is the only one in free Google Colab sessions with plenty of runtime. This GPU features Compute Capability 7.5.

2.2 Multiprocessor Count

2.2.1 MULTIPROCESSOR_COUNT Attribute

The number of streaming multiprocessors (SMs) can be retrieved programmatically as shown below:

```
device = cuda.get_current_device()
multi_processor = getattr(device, 'MULTIPROCESSOR_COUNT')
print(multi_processor)
```

Output:

```
40
```

2.2.2 Core count

The way of getting the core count according to a Stackoverflow thread `https://stackoverflow.com/questions/63823395/how-can-i-get-the-number-of-cuda-cores-in-my-gpu-using-python-and-numba`, we should get the `device.compute_capability` first, and then search for the whole internet to get the number of cores per SM (Streaming Multiprocessor), and then multiply them together.

```
compute_cap = device.compute_capability
print(compute_cap)

(7.5)

# 7.5 -> https://developer.nvidia.com/cuda-gpus -> 7.5 seems to be
    ampere/Turing
# https://en.wikipedia.org/wiki/Ampere_(microarchitecture) -> 64
   cores/sm
core_per_sm = 64
total_cores = multi_processor * core_per_sm
print(total_cores)

2560
```

2.3 Memory count

We need to get the `current_context` from the cuda session and get the total memory that of the GPU we got allocated to.

```
memory_info = cuda.current_context().get_memory_info()
total = getattr(memory_info, 'total')
print(str(total/1024/1024) + "MB")

15095.0625MB
```

That's 15GB VRAM.