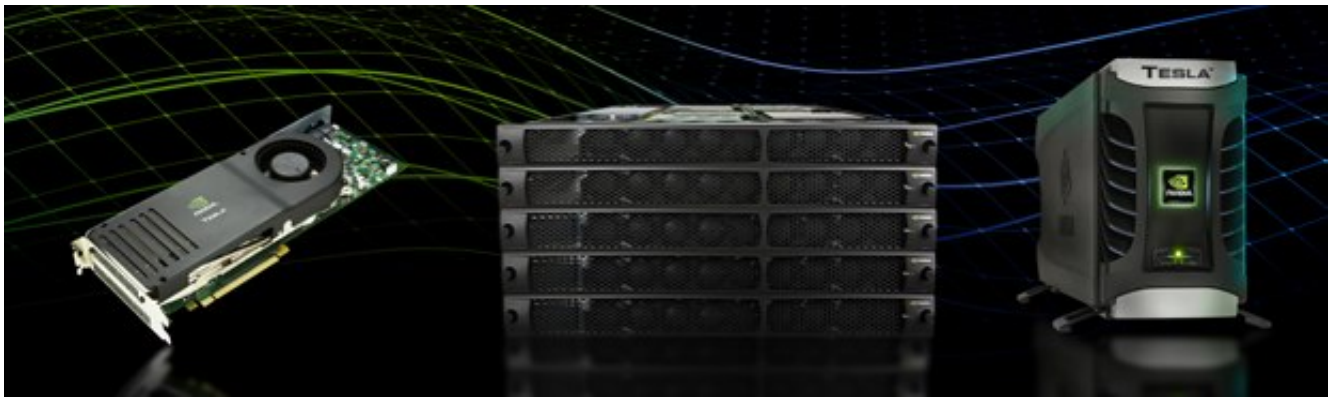
M2-BIG DATA
GPGPU Chapter 2

Exercice



Réalisé par :
Gaby Maroun

Encadré par :
Dr. Etancelin JM

21 octobre 2020

Objectives

Get familiar with GPUs significant figures.

Instructions

Complete the given code Chap2_Ex1.cu in order to display :

- The number of GPU on the machine (use the $cudaGetDeviceCount$ function)
- Foreach device found, retrieve the data informations in the C structure struct $cudaDeviceProp$ using the $cudaGetDeviceProperties$ function.

```cpp
#include <cuda_runtime.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {
  printf("%s Starting...\n\n", argv[0]);

  // Get the device count
  int deviceCount = 0;
  //@TODO@ : Complete here

  // getCount function
  cudaGetDeviceCount(&deviceCount);

  // Loop over devices
  for (int dev = 0; dev < deviceCount; ++dev) {
    // Set the current device
    cudaSetDevice(dev);

    // Fill the data structure with device properties
    struct cudaDeviceProp deviceProp;
    cudaGetDeviceProperties(&deviceProp, dev);
    //@TODO@ : Complete here

    // Display some properties
    printf("\nDevice %d: \"%s\"\n", dev, deviceProp.name);
    //@TODO@ : Complete here
    printf("\nDevice %d: \"%ld\"\n", dev, deviceProp.totalGlobalMem);
    printf("\nDevice %d: \"%d\"\n", dev, deviceProp.multiProcessorCount);
    printf("\nDevice %d: \"%d\"\n", dev, deviceProp.clockRate);
    printf("\nDevice %d: \"%d\"\n", dev, deviceProp.maxThreadsPerBlock);
    printf("\nDevice %d: \"%d\"\n", dev, deviceProp.maxThreadsPerMultiProcessor);

    printf("\nDevice %d: \"%d\"\n", dev, deviceProp.maxGridSize[0]);


    printf("\nDevice %d: \"%d\"\n", dev, deviceProp.maxThreadsDim[0]);

  }
  // finish
  exit(EXIT_SUCCESS);
}
```

Then display some fields of the structure to find :

— the device's name ($name$),
— the total amount of global memory available ($totalGlobalMem$ given in Bytes),
— the number of processors ($multiProcessorCount$),
— the clock rate of the processors ($deviceProp.clockRate$ given in Hz) in GHz,
— the maximum number of threads per block ($maxThreadsPerBlock$),
— the maximum number of threads per multiprocessors ($maxThreadsPerMultiProcessor$),
— the maximum number of blocks in the grid ($maxGridSize$ as a 3D index),
— the maximum threads per block ($maxThreadsDim$).

```
gmaroun@scinfe061:~/Téléchargements$ nvcc 'Chap2_Ex1(1).cu'
gmaroun@scinfe061:~/Téléchargements$ ./a.out
./a.out Starting...


Device 0: "Quadro RTX 4000"

Device 0: "8338604032"

Device 0: "36"

Device 0: "1545000"

Device 0: "1024"

Device 0: "1024"

Device 0: "2147483647"

Device 0: "1024"
```

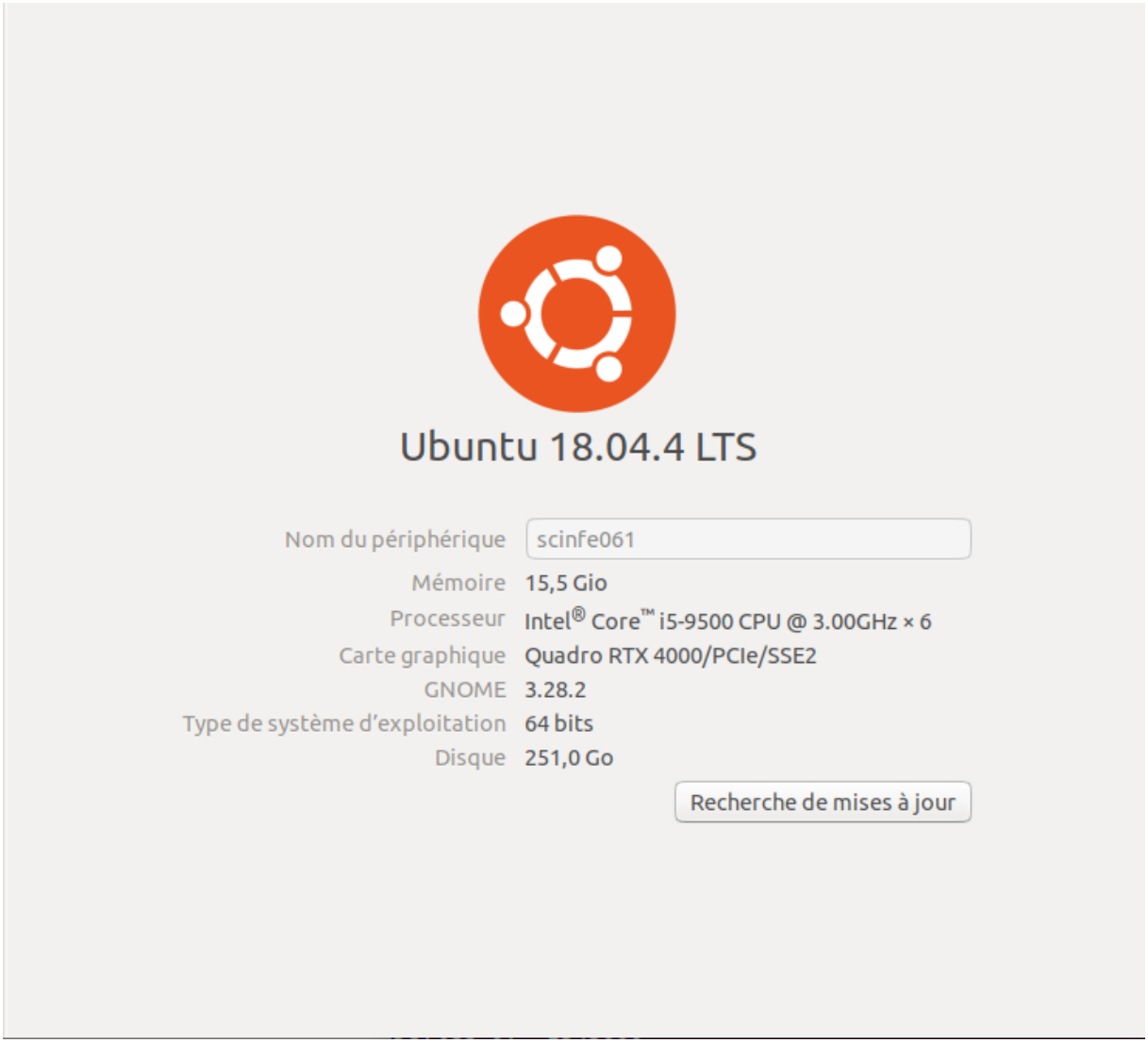
More details on the data structure in the reference documentation : https ://docs.nvidia.com/cuda/cuda-runtime-api/index.html

Compile with $nvcc$ compiler.

Questions

Use your code to answer the following questions :

1. What is the fastest core between the GPU's core and the CPU's core ?



We can deduce from this screenshot that the CPU is at 3GHz while the GPU, and using the $deviceProp.clockRate$ function, shows a frequency of 1.545GHz which indicates the superiority of the GPU in speed over the CPU.

2. Compare the GPU's memory size with the host main memory.

   The GPU has a memory size of 8.3GB, meanwhile the CPU's memory size is way bigger at 15.5GB.

3. What is the theoretical computing power of the GPU ? give the answer in number of operations per second. Compare with the manufacturer information (search on the Internet)

   *FLOPS = sockets * (cores per socket) * (number of clock cycles per second) * (number of floating point operations per cycle).*

   $1(socket)*6(cores)*8,338,604,032(cyclespersecond)*8(single-precisionFLOPspersecond) = 400,252,993,500$ *single-precision FLOPs per second.*

   While it's noted on the Nvidia Website that it's 7.1 TFLOPS for single precision.

4. What is the maximum number of threads that the GPU can handle ?
   We have the maximum number of blocks in the grid which is 2147483647blocks and each block has a maximum number of threads, that is 1024threads, so the total would be

$$2,147,483,647blocks * 1024threads = 2,199,023,255,000$$

La fin.