

# 并行计算实验一报告

## ——OpenMP 及 CUDA 实验环境的搭建

PB20111701 叶升宇

PB20111689 蓝俊玮

### 说明

以下为 PB20111701 叶升宇的实验环境。

## OpenmMP 环境搭建

### 下载 CLion

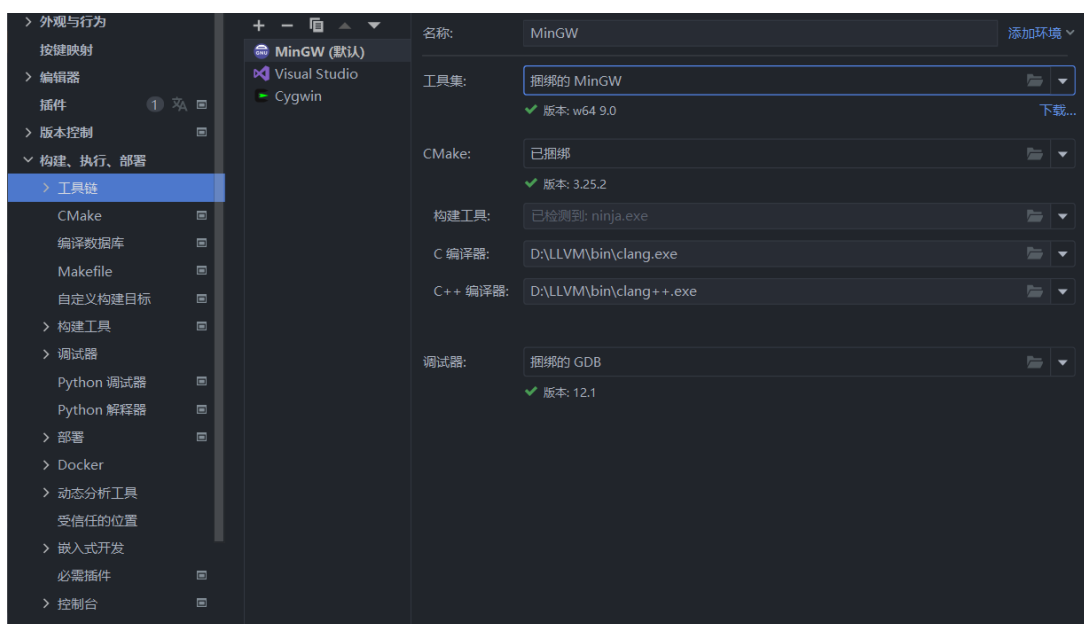
本学期并行计算实验使用的 IDE 为 CLion，下载链接如下 [CLion](#)。

### 下载 MinGW64

使用 MinGW64 作为工具集，下载链接如下 [MinGW64](#)。

### 配置 LLVM + clang

前往 [LLVM](#) 官网下载，c 编译器使用 LLVM clang，c++ 编译器使用 LLVM clang++，构建工具使用 Ninja(而非通常的 Make)。整体工具链配置完毕后如下：



### 配置 CMake

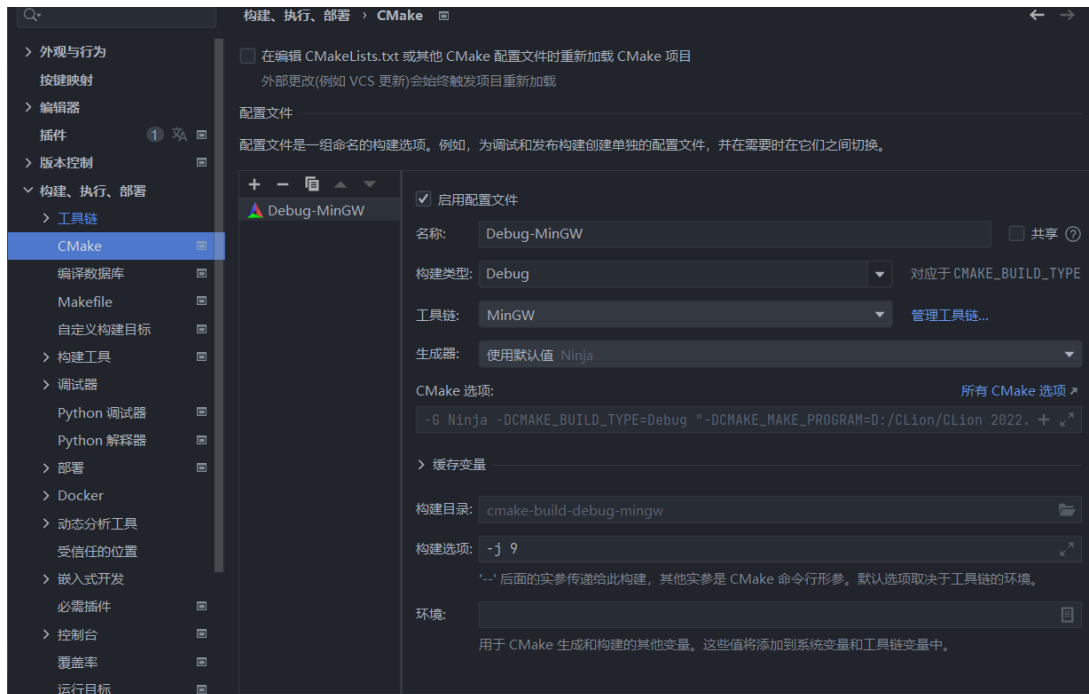
编写 CMakeLists.txt 如下：

- 使用 c++ 14 标准；
- 用 -fopenmp 选项开启 OpenMP 支持

```
cmake_minimum_required(VERSION 3.10)
project(sort)
set(CMAKE_CXX_STANDARD 14)
find_package(OpenMP REQUIRED)
set(SOURCE_FILES main.cpp)
```

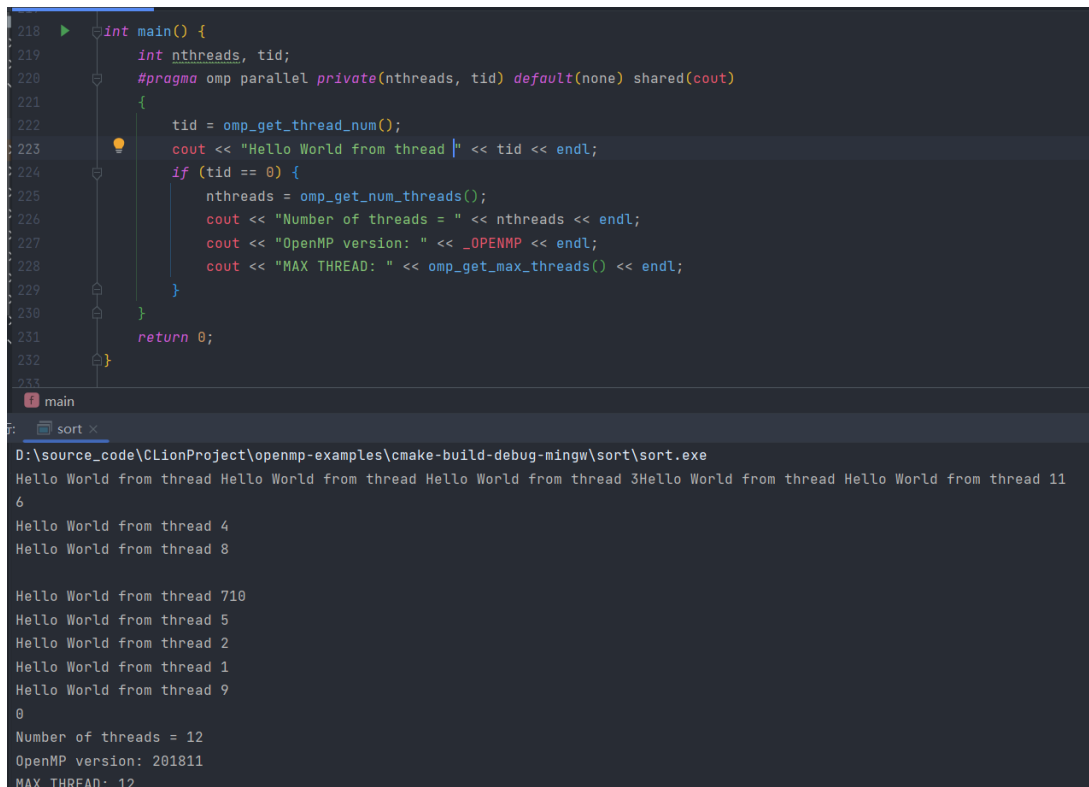
```
add_executable(sort ${SOURCE_FILES})
set(CMAKE_C_COMPILER "clang")
set(CMAKE_CXX_COMPILER "clang++")
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -fopenmp")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -fopenmp")
```

Cmake 相关配置如下：



## 验证配置成功

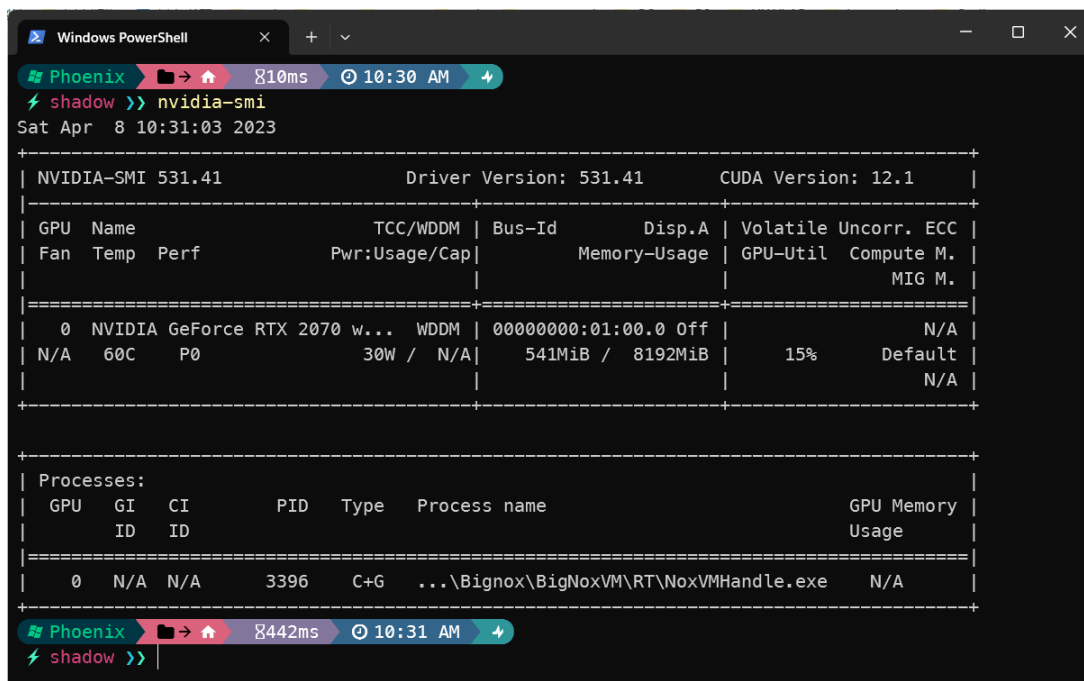
打印简单的 OpenMP 相关参数，来验证配置成功：



## CUDA 环境搭建

### NVIDIA 驱动的安装

依然使用 CLion 作为 IDE，配 OpenMP 时已经完成，不再赘述。下面检查驱动是否安装：在 Windows Terminal 中输入 `nvidia-smi`：



```
Windows PowerShell
Phoenix 810ms 10:30 AM
shadow >> nvidia-smi
Sat Apr 8 10:31:03 2023

+-----+
| NVIDIA-SMI 531.41                  Driver Version: 531.41      CUDA Version: 12.1   |
+-----+-----+
| GPU Name                               TCC/WDDM | Bus-Id      Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf              Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                           | MIG M.         |
+-----+-----+
|  0  NVIDIA GeForce RTX 2070 w...  WDDM  00000000:01:00.0 Off |                  N/A | |
| N/A   60C   P0              30W /  N/A |    541MiB /  8192MiB |      15%    Default |
|                                           |                  N/A |
+-----+-----+

+-----+
| Processes:                         |
| GPU   GI    CI          PID    Type    Process name                        GPU Memory |
| ID                                         |              | Usage       |
+-----+-----+
|  0   N/A   N/A         3396     C+G     ...\\Bignox\\BigNoxVM\\RT\\NoxVMHandle.exe  N/A       |
+-----+

Phoenix 8442ms 10:31 AM
shadow >>
```

#### 说明

这里我是已经配好了 CUDA 环境，所以驱动和 CUDA 版本都有显示。

没有安装驱动的情况下，需要到 [英伟达官网](#) 选取对应显卡的版本下载安装。

### CUDA 安装

进入 [CUDA Toolkit 官网](#) 进行下载并安装，我选择的是 12.1 版本，同时在系统变量中加入 CUDA 路径：

CUDA_PATH	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1
CUDA_PATH_V12_1	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1

以及在 PATH 中加入 CUDA 的 BIN 路径：

```
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1\bin
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1\libnvvp
```

### 验证安装成功

下面通过运行 `deviceQuery.exe`、`bandwidthTest.exe`，`result = PASS` 说明安装成功：

```
Windows PowerShell
Phoenix C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1\extras\demo_suite 80ms 10:55 AM
shadow >> .\deviceQuery.exe
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1\extras\demo_suite\deviceQuery.exe Starting...

CUDA Device Query (Runtime API) version (CUDA static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA GeForce RTX 2070 with Max-Q Design"
  CUDA Driver Version / Runtime Version      12.1 / 12.1
  CUDA Capability Major/Minor version number: 7.5
  Total amount of global memory:             8192 MBytes (8589606912 bytes)
  (36) Multiprocessors, ( 64) CUDA Cores/MP: 2304 CUDA Cores
  GPU Max Clock rate:                       1125 MHz (1.13 GHz)
  Memory Clock rate:                        5501 Mhz
  Memory Bus Width:                         256-bit
  L2 Cache Size:                           4194304 bytes
  Maximum Texture Dimension Size (x,y,z)    1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers
  Total amount of constant memory:           zu bytes
  Total amount of shared memory per block:   zu bytes
  Total number of registers available per block: 65536
  Warp size:                                32
  Maximum number of threads per multiprocessor: 1024
  Maximum number of threads per block:      1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                     zu bytes
  Texture alignment:                        zu bytes
  Concurrent copy and kernel execution:     Yes with 2 copy engine(s)
  Run time limit on kernels:                 Yes
  Integrated GPU sharing Host Memory:        No
  Support host page-locked memory mapping:  Yes
  Alignment requirement for Surfaces:        Yes
  Device has ECC support:                    Disabled
```

```
Windows PowerShell
Phoenix C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1\extras\demo_suite 8356ms 10:55 AM
shadow >> .\bandwidthTest.exe
[CUDA Bandwidth Test] - Starting...
Running on...

Device 0: NVIDIA GeForce RTX 2070 with Max-Q Design
Quick Mode

Host to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)      Bandwidth(MB/s)
  33554432                   12418.1

Device to Host Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)      Bandwidth(MB/s)
  33554432                   11766.2

Device to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)      Bandwidth(MB/s)
  33554432                   295113.6

Result = PASS
```

## 配置 CMake

在 CLion 中, 选择项目为 CUDA 可执行文件, 然后编写 CMakeLists.txt 如下:

```
cmake_minimum_required(VERSION 3.10)
project(cuda_examples CUDA)
set(CMAKE_CUDA_STANDARD 14)
```

```
find_package(CUDA REQUIRED)
add_executable(cuda_examples main.cu)
```

这里以老师 PPT 上的乘积求和程序为例，输出结果如下，也验证了 CUDA 配置成功：

```
57     cudaMemcpy( dst: d_a, src: a, count: size, kind: cudaMemcpyHostToDevice);
58     cudaMemcpy( dst: d_b, src: b, count: size, kind: cudaMemcpyHostToDevice);
59
60     Dot <<< gridDim: 1, blockDim: N >>>( a: d_a, b: d_b, c: d_c); // single block multi threads
61     cudaMemcpy( dst: c, src: d_c, count: sizeof(int), kind: cudaMemcpyDeviceToHost);
62
63     int sumHost = 0;
64     for (int i = 0; i < N; i++) {
65         sumHost += a[i] * b[i];
66     }
67     printf( Format: "sum Calculated on Host = %d\n", sumHost);
68     printf( Format: "Device to Host: a * b = %d\n", *c);
69
70     free( Block: a);
71     free( Block: b);
```

main

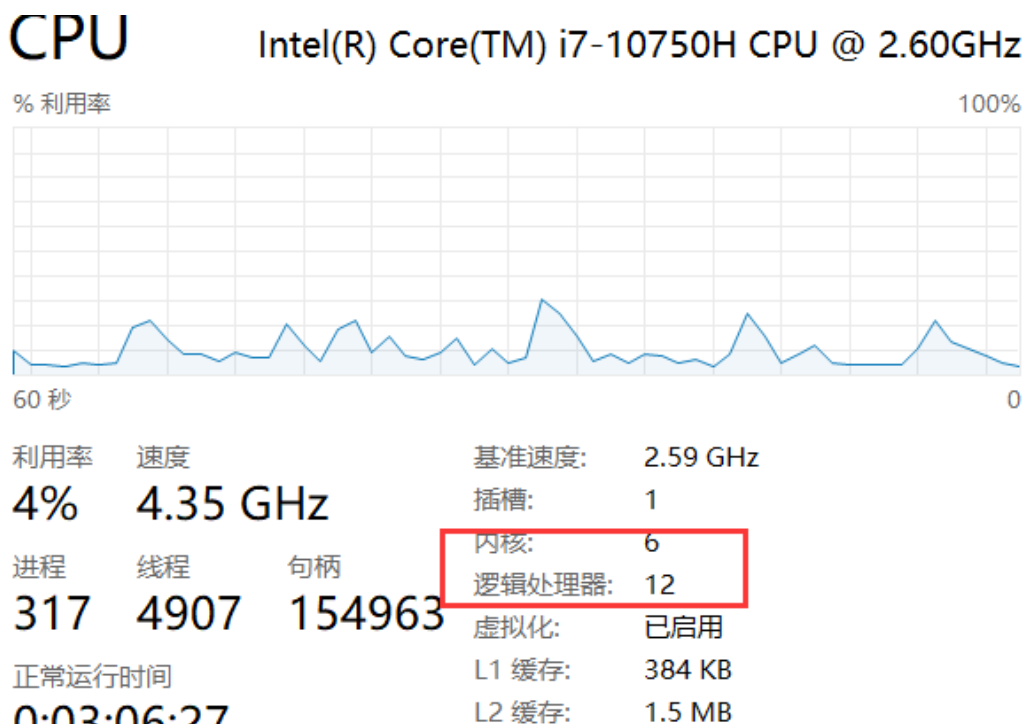
运行: cuda\_examples x

Array a[N]:  
1 7 4 0 9 4 8 8 2 4  
Array b[N]:  
5 5 1 7 1 1 5 2 7 6  
sum Calculated on device 151  
sum Calculated on Host = 151  
Device to Host: a \* b = 151  
进程已结束,退出代码0

## 其它设备参数

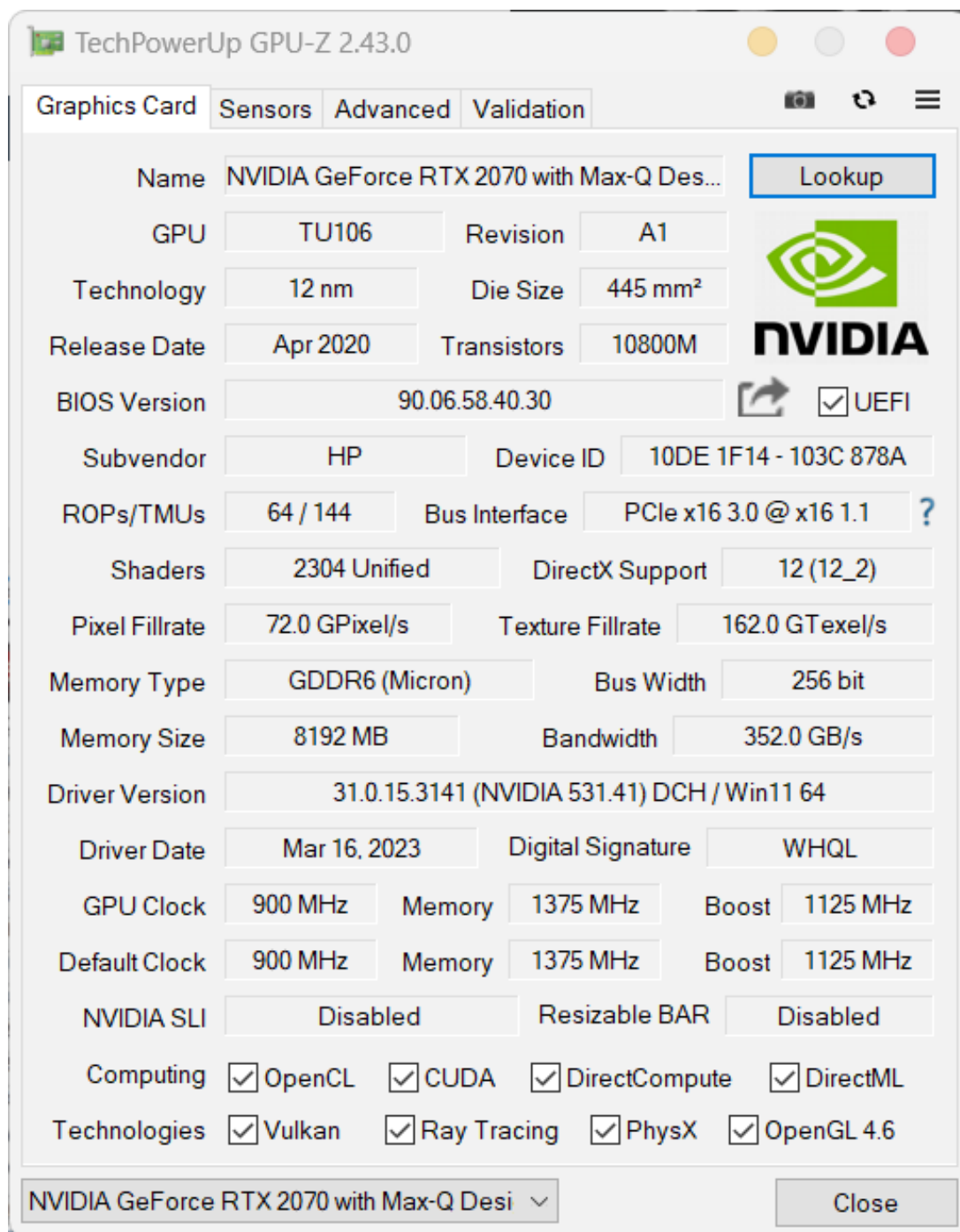
### CPU 参数

通过任务管理器来查看， 我的电脑是 6 核 12 线程的：



## GPU 参数

利用课程群中提供的 GPU-Z 查看 GPU 参数如下：



### 说明

实际上，在先前配置 CUDA 时候通过 deviceQuery.exe 也可以来获取相关参数。