**ESERCITAZIONE 6**

***Marrazzo Vincenzo***

***Spagna Zito Marika***

**Caratteristiche GPU:**

**GPU Colab:** Tesla T4

**Compute Capability:** 7.5

|  |  |
| --- | --- |
| Technical specifications | Compute Capability |
| Maximum x-dimension of a grid of thread blocks | 231-1 |
| Maximum y-, or z-dimension of a grid of thread blocks | 65535 |
| Maximum number of threads per block | 1024 |
| Maximum number of resident blocks per multiprocessor | 16 |
| Maximum number of resident threads per multiprocessor | 1024 |
| Number of 32-bit registers per multiprocessor | 64K |
| Maximum amount of shared memory per multiprocessor | 64K |
| Maximum amount of shared memory per thread block | 64K |

Ogni SM può gestire fino a 1024 thread ed un numero massimo di 16 blocchi.

Considerando il **blocco 8x8** abbiamo un totale di 64 thread.

**(8x8) =64 < 1024** vincolo della dimensione del blocco soddisfatto.

Siccome soddisfa il massimo numero di blocchi per SM consideriamo tutti e 16 i blocchi.

**16X64= 1024** thread per SM (Otteniamo la piena occupazione dello SM sia per quanto riguarda i thread che per i blocchi)

**STRATEGIA 1**

Controlliamo se questo valore rispetta il vincolo della memoria attraverso l’istruzione **!nvcc -Xptxas -v Ese\_prod\_strategia1.cu**. che ci fornisce il numero di registri utilizzato da ogni thread. Otteniamo il valore **10** e notiamo:

**n° registri per ogni SM 1024 x 10 = 10.240 < 64.000**

**n° registri per ogni thread**

**n° di thread per ogni SM**

**STRATEGIA 2**

Controlliamo se questo valore rispetta il vincolo della memoria attraverso l’istruzione **!nvcc -Xptxas -v Ese\_prod\_strategia2.cu**. che ci fornisce il numero di registri utilizzato da ogni thread. Otteniamo il valore **15** e notiamo:

**n° registri per ogni SM 1024 x 15 = 15.360 < 64.000**

**n° registri per ogni thread**

**n° di thread per ogni SM**

Nessuna info a priori su memoria shared per l’allocazione dinamica.

**STRATEGIA 3**

Controlliamo se questo valore rispetta il vincolo della memoria attraverso l’istruzione! nvcc **-Xptxas -v Ese\_prod\_strategia3.cu**. che ci fornisce il numero di registri utilizzato da ogni thread. Otteniamo il valore 9 e notiamo:

**n° registri per ogni SM 1024 x 9 = 9216 < 64.000**

**n° registri per ogni thread**

**n° di thread per ogni SM**

Nessuna info a priori su memoria shared per l’allocazione dinamica.

**TEMPO SEQUENZIALE**

|  |  |
| --- | --- |
| **N** | **tempo CPU (s)** |
| 1.000.000 | 0,00296 |
| 2.000.000 | 0,005892 |
| 4.000.000 | 0,012064 |
| 8,00E+06 | 0,023971 |
| 1,60E+07 | 0,048648 |

**SPEED-UP STRATEGIA 1**

|  |  |  |
| --- | --- | --- |
| **N** | **tempo GPU(s)** | **Sp** |
| 1.000.000 | 0,008283 | 0,36 |
| 2.000.000 | 0,016114 | 0,37 |
| 4.000.000 | 0,032131 | 0,38 |
| 8,00E+06 | 0,064187 | 0,37 |
| 1,60E+07 | 0,129335 | 0,38 |

**SPEED-UP STRATEGIA 2**

|  |  |  |
| --- | --- | --- |
| **N** | **tempo GPU(s)** | **Sp** |
| 1.000.000 | 0,000282 | 10,50 |
| 2.000.000 | 0,000494 | 11,93 |
| 4.000.000 | 0,000938 | 12,86 |
| 8,00E+06 | 0,001883 | 12,73 |
| 1,60E+07 | 0,003618 | 13,45 |

**SPEED-UP STRATEGIA 3**

|  |  |  |
| --- | --- | --- |
|  | | |
| **N** | **tempo GPU(s)** | **Sp** |
| 1.000.000 | 0,000247 | 11,98 |
| 2.000.000 | 0,000478 | 12,33 |
| 4.000.000 | 0,000856 | 14,09 |
| 8,00E+06 | 0,001657 | 14,47 |
| 1,60E+07 | 0,003361 | 14,47 |