GPU 1

Michał Śledź

12 maja 2020

1 Środowisko testowe

procesor : i5-9600K 6 rdzeni, 3,70-4,60 GHz, 9MB Cache

pamięć : 16 GB RAM, 3200 MHz, CL 16

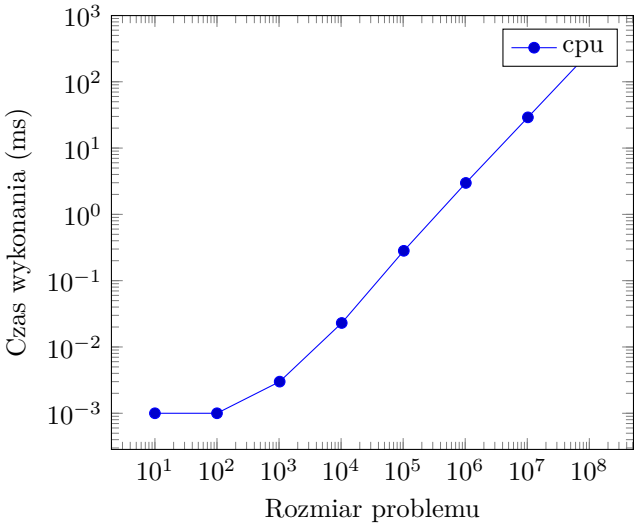
```
./deviceQuery Starting...

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

Detected 1 CUDA Capable device(s)

Device 0: "GeForce RTX 2060"
CUDA Driver Version / Runtime Version          10.2 /
    10.2
CUDA Capability Major/Minor version number:    7.5
Total amount of global memory:                 5931
   MBytes (6219563008 bytes)
(30) Multiprocessors, ( 64) CUDA Cores/MP:     1920
   CUDA Cores
GPU Max Clock rate:                            1770
   MHz (1.77 GHz)
Memory Clock rate:                             7001
   Mhz
Memory Bus Width:                              192-
   bit
L2 Cache Size:
   3145728 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:               65536
    bytes
Total amount of shared memory per block:       49152
    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:
    2147483647 bytes
Texture alignment:                             512
    bytes
Concurrent copy and kernel execution:          Yes
    with 3 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
Device supports Unified Addressing (UVA):      Yes
Device supports Compute Preemption:            Yes
Supports Cooperative Kernel Launch:            Yes
Supports MultiDevice Co-op Kernel Launch:      Yes
Device PCI Domain ID / Bus ID / location ID:   0 / 1
    / 0
Compute Mode:
< Default (multiple host threads can use ::
    cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver
    Version = 10.2, CUDA Runtime Version = 10.2,
    NumDevs = 1
Result = PASS

[CUDA Bandwidth Test] - Starting...
Running on...

Device 0: GeForce RTX 2060
Quick Mode
```

```
Host to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
Transfer Size (Bytes)       Bandwidth(GB/s)
32000000                    12.8

Device to Host Bandwidth, 1 Device(s)
PINNED Memory Transfers
Transfer Size (Bytes)       Bandwidth(GB/s)
32000000                    13.1

Device to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
Transfer Size (Bytes)       Bandwidth(GB/s)
32000000                    285.7

Result = PASS

NOTE: The CUDA Samples are not meant for performance
   measurements. Results may vary when GPU Boost is
   enabled.
```

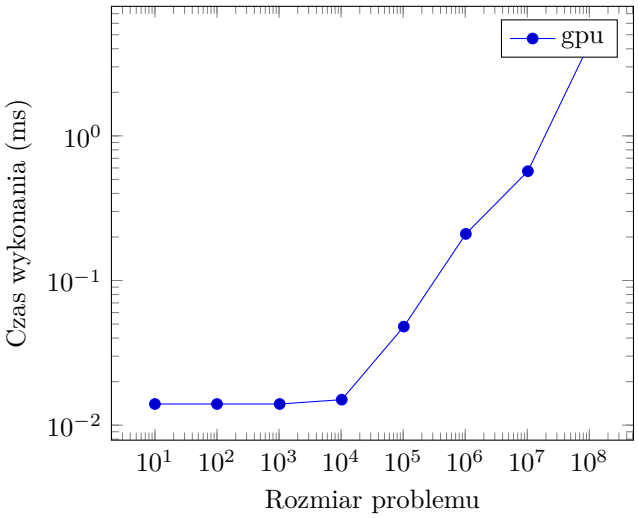
2 Benchmarki

Przyjęto następujące rozmiary problemów: 10, 100, 1024, 10240, 102400, 1024000, 10240000, 102400000. W pomiarach nie są uwzględniane czasy: alokacji/dealokacji pamięci, kopiowania tablic `host->device` oraz `device->host`.



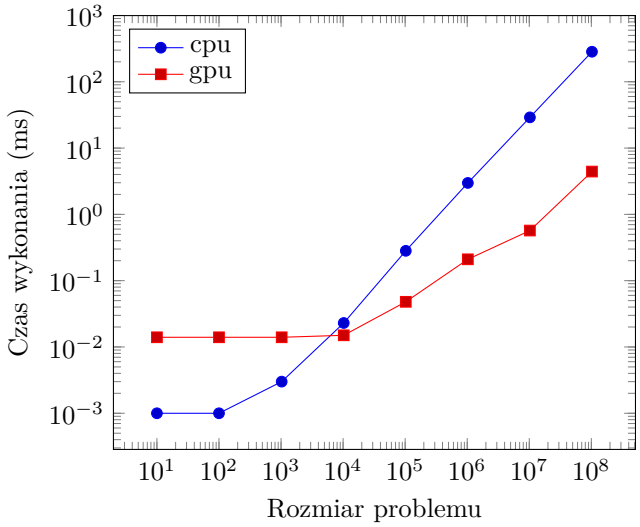
Rysunek 1: Wydajność CPU w zależności od rozmiaru problemu. Wykres w skali logarytmicznej.

Wykres 1 przedstawia wydajność CPU w zależności od rozmiaru problemu. Dodawanie wektorów w wersji dla CPU zostało napisane sekwencyjnie. Wyniki są spodziewane. Czas wykonania na CPU rośnie praktycznie liniowo poza pierwszymi dwoma pomiarami, które są na tyle małe, że różnica w czasie ich wykonania może się mieścić w granicach błędu statystycznego.



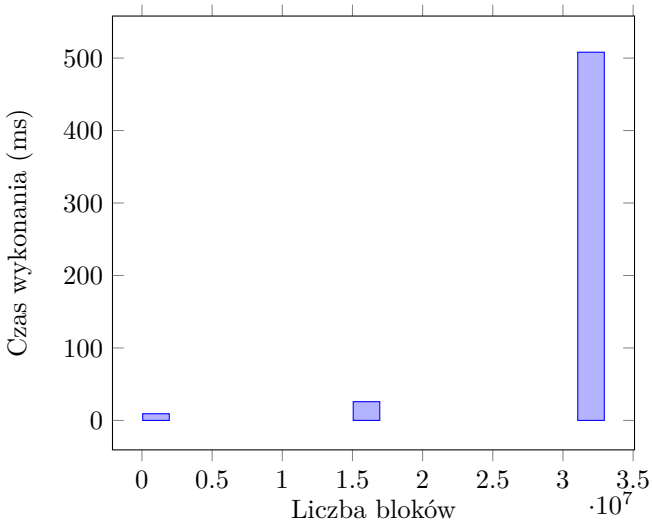
Rysunek 2: Wydajność GPU w zależności od rozmiaru problemu. Wykres w skali logarytmicznej.

Wykres 2 przedstawia wydajność GPU w zależności od rozmiaru problemu. Spodziewałbym się, że w momencie, w którym rozmiar problemu zaczyna być większy od łącznej liczby rdzeni (1920), czas wykonania nieco się zmieni, ale możliwe, że ze względu na małe rozmiary problemu jest to niewidoczne. Zauważalny wzrost czasu wykonania następuje dopiero od rozmiaru problemu równego 102400. Benchmarki na GPU były wykonane zawsze z maksymalną liczbą wątków per blok i minimalną liczbą bloków, która zapewnia pokrycie całego wektora.



Rysunek 3: Wydajność CPU i GPU w zależności od rozmiaru problemu. Wykres w skali logarytmicznej.

Wykres 3 przedstawia wcześniejsze dwa wykresy, nałożone na siebie w skali logarytmicznej. Na początku CPU jest nieco szybsze. Wynika to z większego taktowania procesora. Dla ostatniego rozmiaru problemu wydajność GPU jest dwa rzędy wielkości większa.



Rysunek 4: Wydajność GPU w zależności od ilości bloków.

Wykres 4 przedstawia wydajność GPU w zależności od liczby bloków. Rozmiar problemu był równy 1024000000. Zostały ustalone następujące proporcje wątki/boki: 1024/1000000, 64/16000000, 32/32000000. Widzimy, że najlepiej wypada konfiguracja, w której używamy najmniejszej liczby bloków przy maksymalnej liczbie wątków per blok.