# Introduction to CUDA and OpenCL

Ilona Tomkowicz, Zofia Pieńkowska

December 21, 2019

## Contents

## 1 Communication with GPU

In order to check details about the GPU used by VM on labolatories (GeForce GTX 1060 6GB) we used a template project deviceQuery.

### 1.1 How to establish connection with device

To etract information about the devices connected to host we use funcion

```
cudaGetDeviceCount(&deviceCount)
```

cudaGetDeviceCount(&deviceCount), which returnes the flag message and changed the passed argument according to the number of devices that were found. Setting onection is done by calling:

```
cudaSetDevice(dev_index);
```

Where dev_index is the index of GPU found (for only 1 GPU it will be 0).

### 1.2 How to fetch information about GPU

To fetch information about GPU, after completing steps in previous point, we can call:

```
cudaDeviceProp deviceProp;
cudaGetDeviceProperties(&deviceProp, dev_index).
```

to get information about device properties or

```
cudaDriverGetVersion(&driverVersion);
cudaRuntimeGetVersion(&runtimeVersion);
```

to check driver and runtime version. After that we can extract specific data like memory size, clock rate, etc using these functions parameters.

## 2 Experiments with data size and grid layout

Experiments with performance were made using template vectorAdd. This template originally

- allocates memory in host and device,
- copies the host input vectors A and B into device memory,

- sets grid layout,
- uses created kernel to make caculations,
- copies the result back into host,
- veifies if the result is correct
- frees memory on both, device and host.

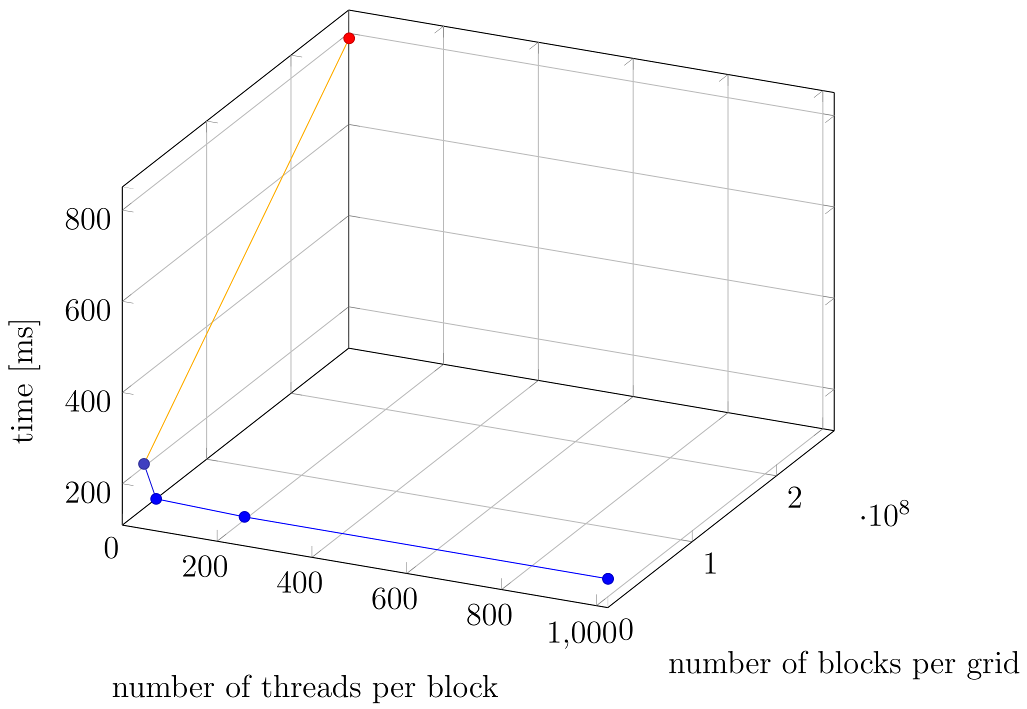
The thread size can change up to 1024, larger threads cannot be built and an error is thrown.

## 2.1 Computation time comparison

For fixed vector size $2^{27}$ and changing grid layout, where blocks and threads size are in a relation with each other in following experiments (changing thread number changes blocks number).

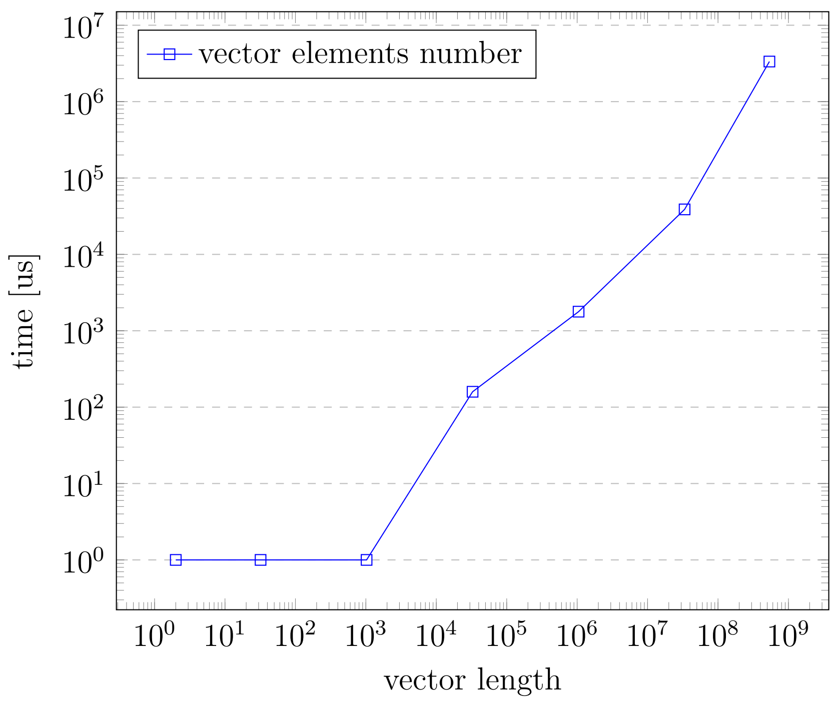
$blocksPerGrid = (vectorElemNumber + threadsPerBlock + 1)/threadsPerBlock$

Computation time performance on different grids



Choosing a size of grid with the largest number of threads and smallest number of blocks per grid the computation performance in relation to vector size was measured. The maximal vector possible to process in this layout was of size $2^{29}$. It can be seen that the worst configuration of grid for a large vector is one thread with a lot of blocks. Choosing any other configuration (from 64 threads up to 1024) have similiar timing.

Computation performance of different vector lengths



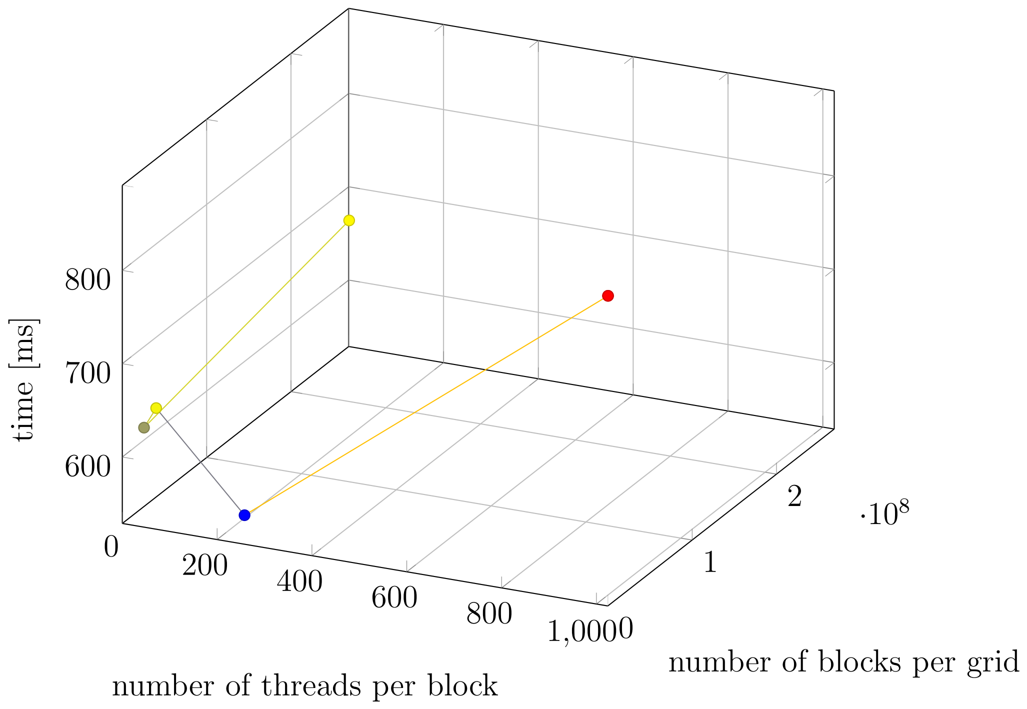
A huge difference can be seen when changing data size when the size of vector is larger than the number of threads per block. When the size of vector exceeds the number of threads per block the difference of time starts to grow immediately.

### 2.2 Data transfer time comparison

Like in the previous experiment, first two plots present resuts for a fixed vector size $2^{27}$ and changing grid layout, where blocks and threads size are in a relation with each other in following experiments (changing thread number changes blocks number).

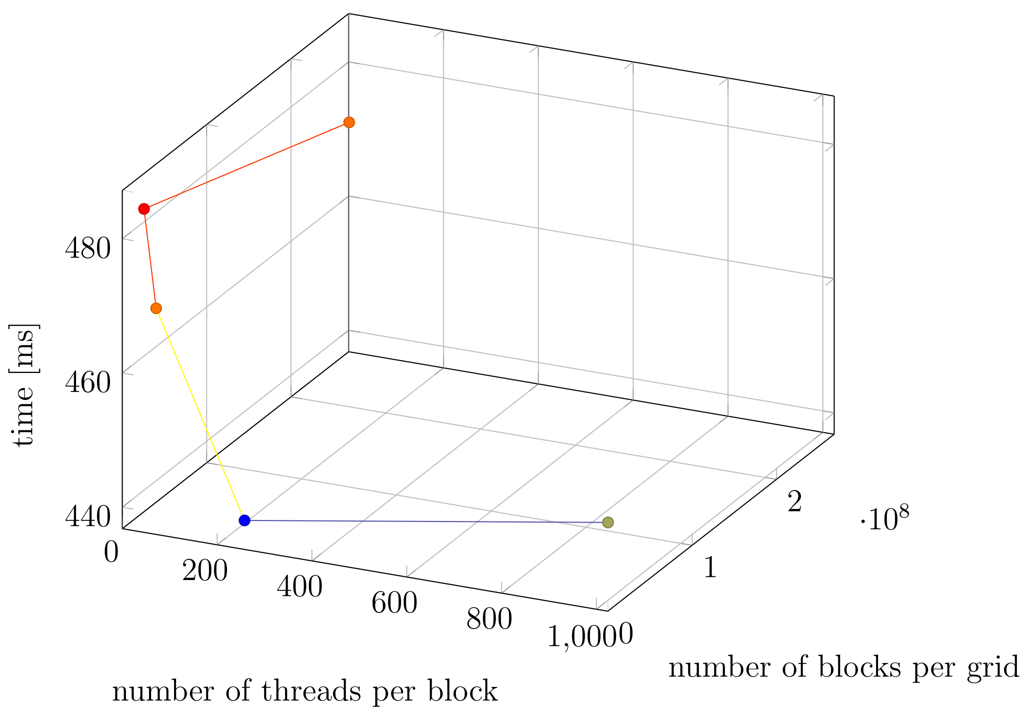
$blocksPerGrid = (vectorElemNumber + threadsPerBlock + 1)/threadsPerBlock$

Host to device transfer perormance on different grids

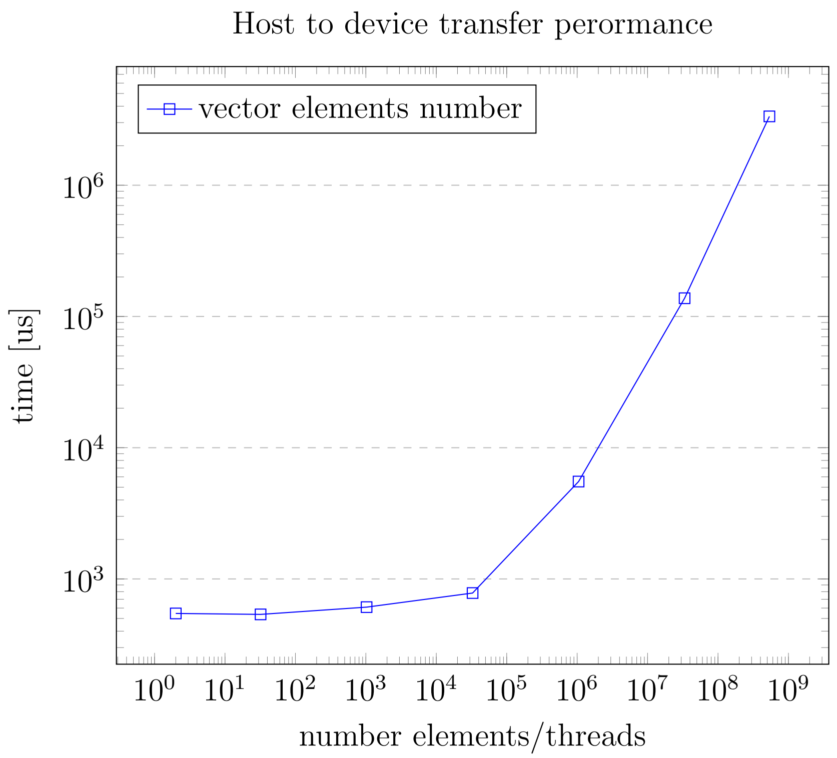


We can observe the best performance for grids with a lot of blocks. Those, which have high number of threads send the data for around 50% more time.

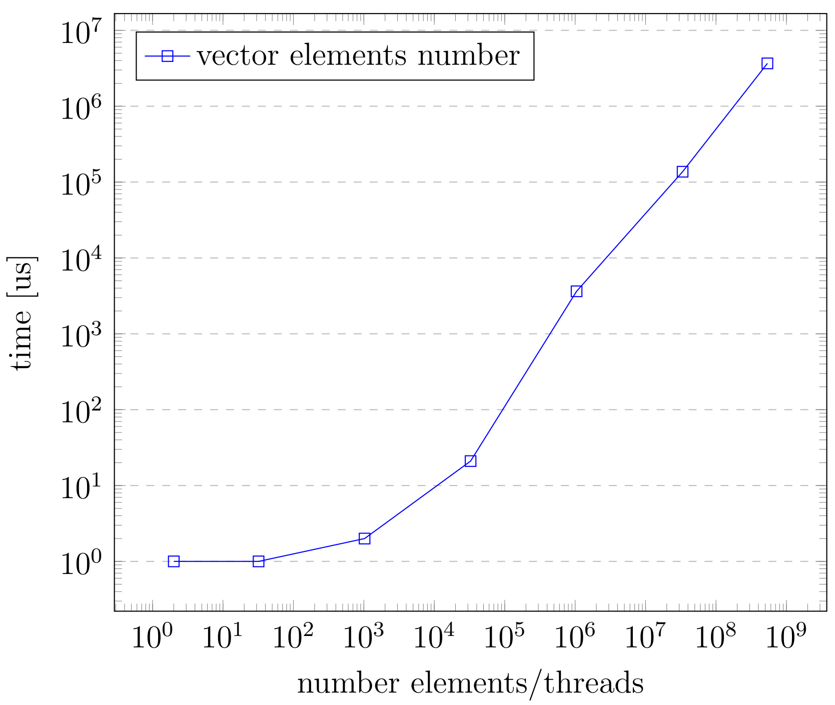
Device to host transfer perormance on different grids



In the back transfer we do not observe any major differences between blocks. They are so small, that the difference can be caused by a normal timing fluctuations observed when running the same code multiple times. Based on this experiment it can be concluded that the transfer from the device to host is not affected by the size of grid in any way.


Host to device transfer perormance
vector elements number
time [us]
number elements/threads
$10^0$ $10^1$ $10^2$ $10^3$ $10^4$ $10^5$ $10^6$ $10^7$ $10^8$ $10^9$
$10^3$ $10^4$ $10^5$ $10^6$

Device to host transfer perormance



In the situation of transfering we can see a very similiar behaviour to the computation one. Until the number of elements do not exceed the size of threads per grid, the transfer time is nearly freezed.

Although transfer times differ for copying from and to the device, the shape of plots look like more or less the same for both operations.

Generally, for all actions on which experiments were conucted, the time change is smaller for low figures and bigger for high figures. It indicates, that using GPU with large number of data transfered at the same time can be less beneficial than granulating this data into subsets and processing them in small portions in GPU. That is why we should try to form our computations in group of smaller calculations, but not too small to aviod too fequent and unnecessary data copying between host and device.