Introduction to CUDA and OpenCL

Lab 3 Kacper Kapuściak

Introduction

In the fourth laboratories, I’ve focused on using and understanding unified memory. I’ve also developed a simple wrapper for easier error handling and implemented a naive matrix multiplication algorithm. Later on, I’ve made measurements that checks how much does unified memory and prefetching data to device affects performance. The results are quite interesting.

All operations were executed on Nvidia RTX 2060 graphics card.

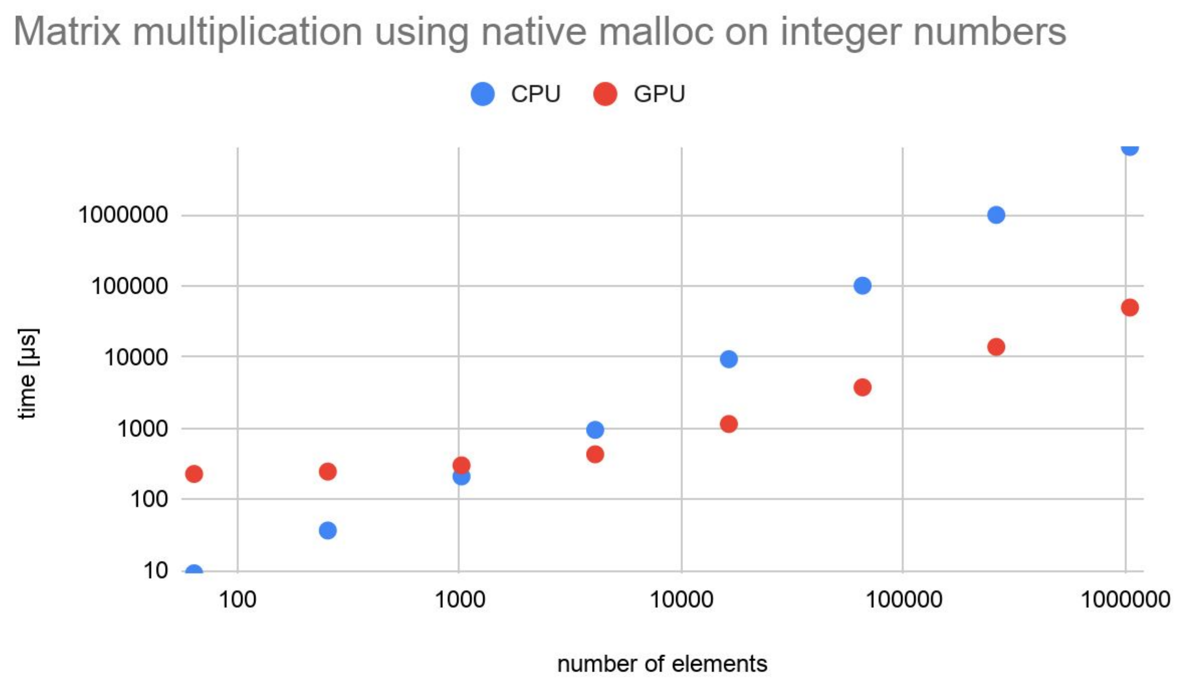
Unified Memory

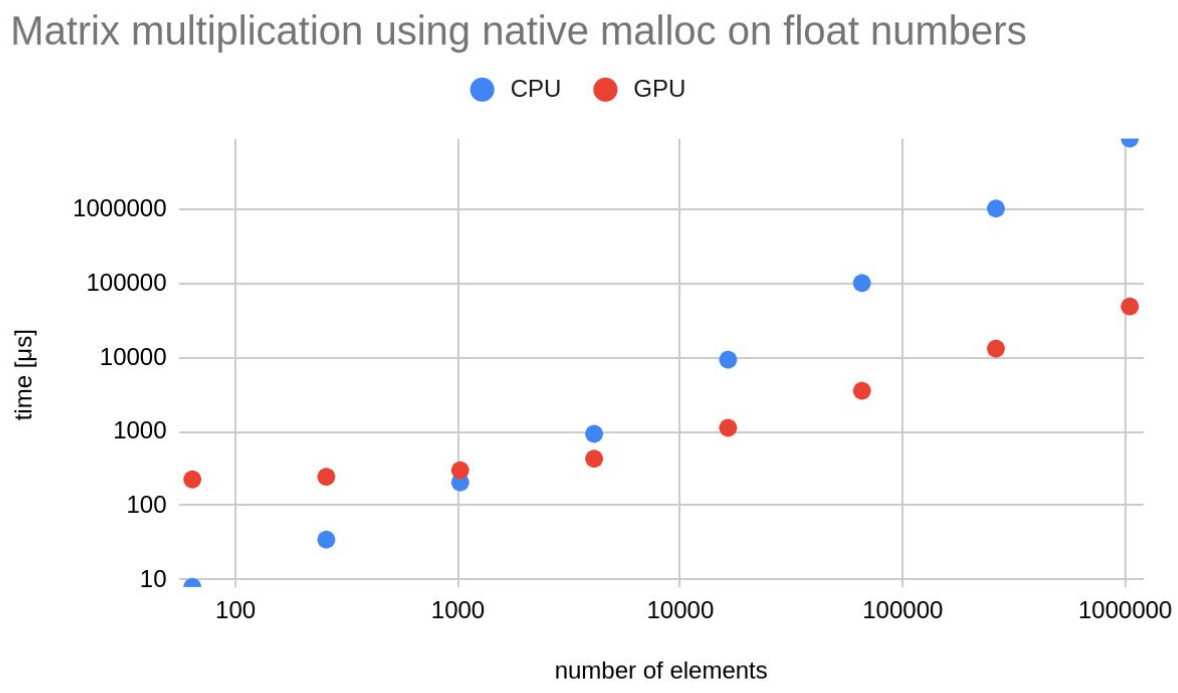
But what exactly is unified memory? According to Nvidia Developer Blog - “Unified Memory is a single memory address space accessible from any processor in a system.” and it is a really good definition. It greatly simplifies CUDA development and makes the code easier to maintain. Using it is as simple as calling the ‘’cudaMallocManaged” function and then freeing the memory with ”cudaFree”. It allocates and frees the memory that is accessible from both GPU and CPU! We no longer have to manually allocate memory with “malloc” and then copy the memory back and forth from GPU. The code is both simpler and easier to use.

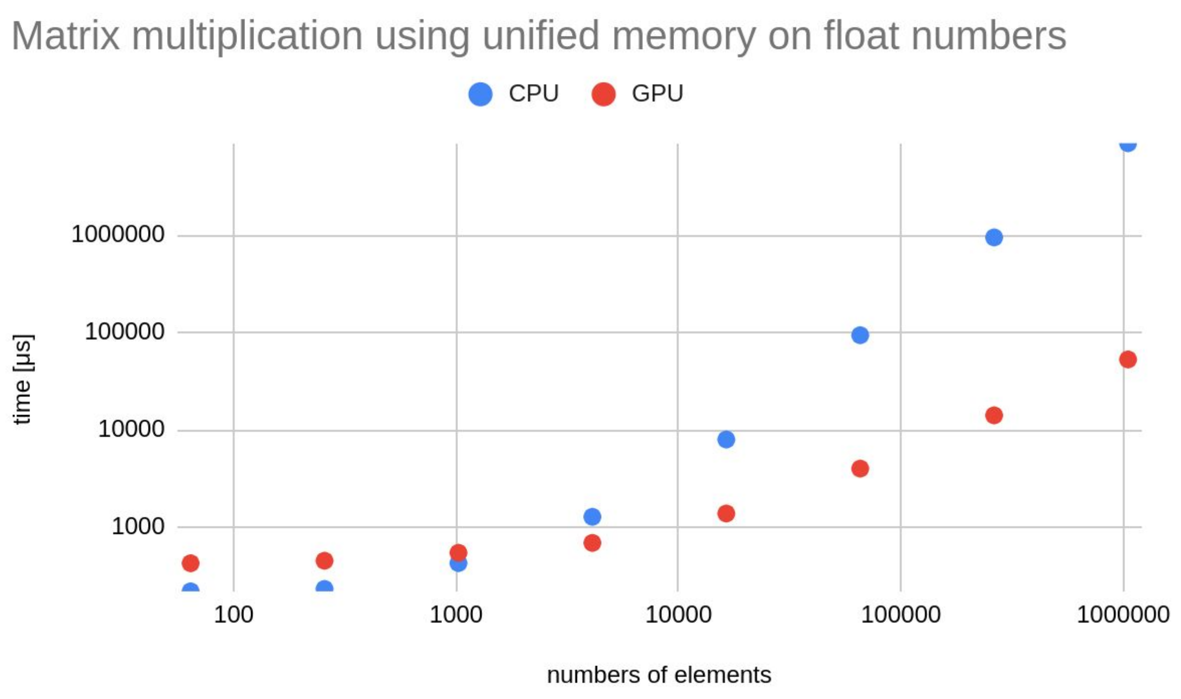
But what about the speed?

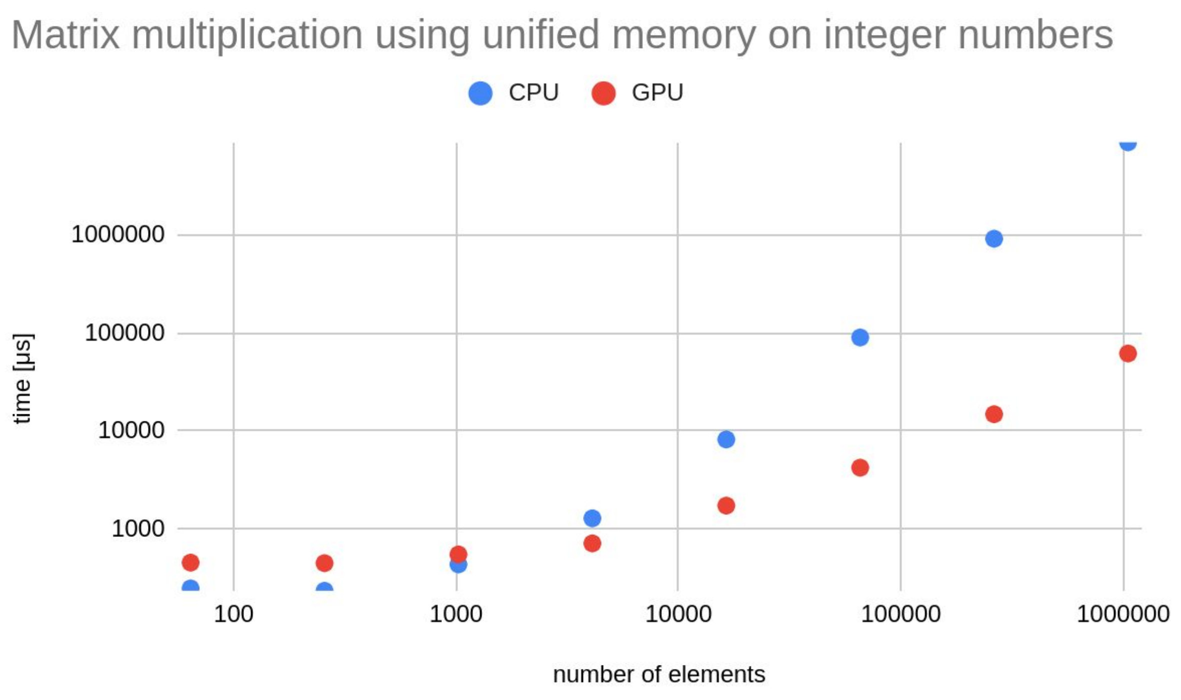
I did measurements for matrices starting from 8x8 and then doubled that number i.e. next matrix was 16x16 etc. For each case, I’ve made 10 measurements and then took the average time of the execution.

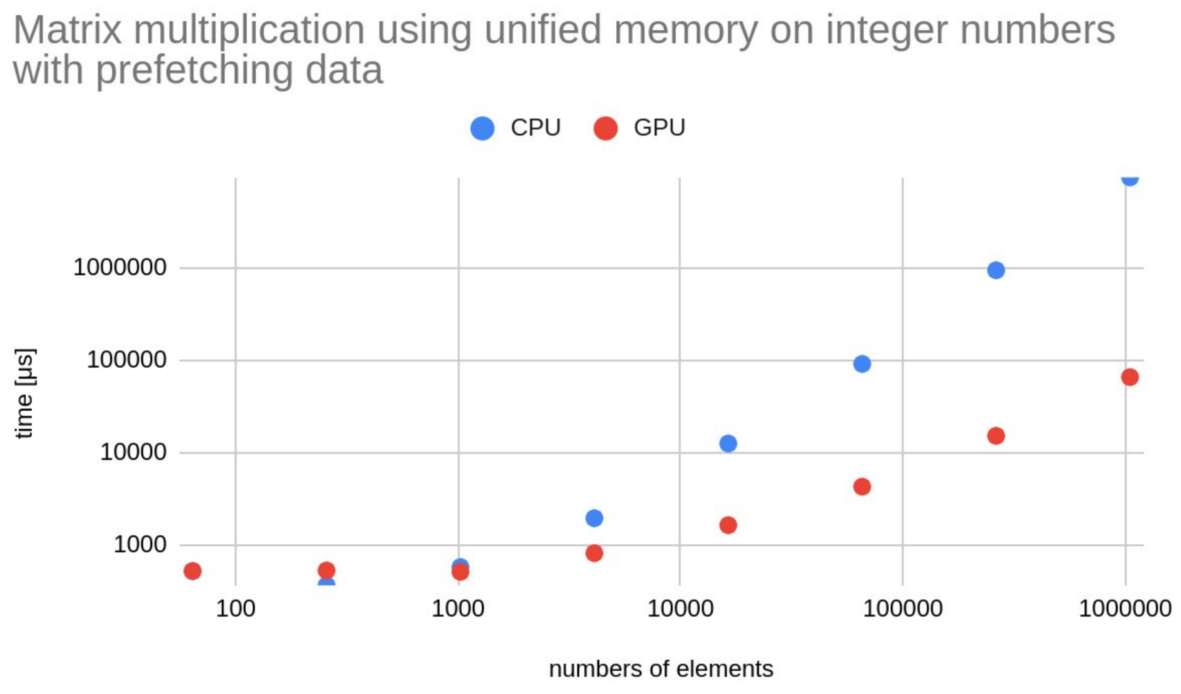
Measurements






Matrix multiplication using unified memory on float numbers
CPU
GPU
time [µs]
1000000
100000
10000
1000
100
1000
10000
100000
1000000
numbers of elements


Matrix multiplication using unified memory on integer numbers
CPU
GPU
time [µs]
1000000
100000
10000
1000
100
1000
10000
100000
1000000
number of elements



Conclusion

My first thought is that the matrix multiplication on GPU is highly effective starting from relatively small matrices. Starting from 1024 elements in a matrix that is 32x32 efficiency skyrocketed for GPU.

As we can see for unified memory not only we have some additional overhead for CPU but all measurements are slower. But that was what I was expecting. Higher memory abstraction from hardware always comes at a cost. In average unified memory was up to 15% slower than bare bone malloc.

In my case, prefetching data had no visible effect on overall performance. I must’ve used it in the wrong way.