AGH University of Science and Technology

Laboratory report 2

Authors: Pedro Alexandre Simões dos Reis,
Iñaki Urrutia Sánchez
Subject: Introduction to CUDA and OpenCL
Year: 2019/2020

Contents

1 Managing page fault exceptions

1.1 Introduction

A page fault is a type of exception raised when a process trys to accede to a memory page that is not currently mapped by the memory management unit. A wrong management of the page fault exceptions can negatively affect performance.

1.2 Comparison between CPU and GPU

This simple piece of code show us a improvement of the time wasted by page faults exceptions when we use the GPU instead of the CPU.

```
//a is an array of integers in both cases
//we assign 1 to every position of a


//CPU version
void hostFunction(int *a, int N)
{
  for (int i = 0; i < N; ++i)
  {
    a[i] = 1;
  }
}


//GPU version
__global__
void deviceKernel(int *a, int N)
{
  int idx = threadIdx.x + blockIdx.x * blockDim.x;
  int stride = blockDim.x * gridDim.x;

  for (int i = idx; i < N; i += stride)
  {
    a[i] = 1;
  }
}
```

And the results are the following:

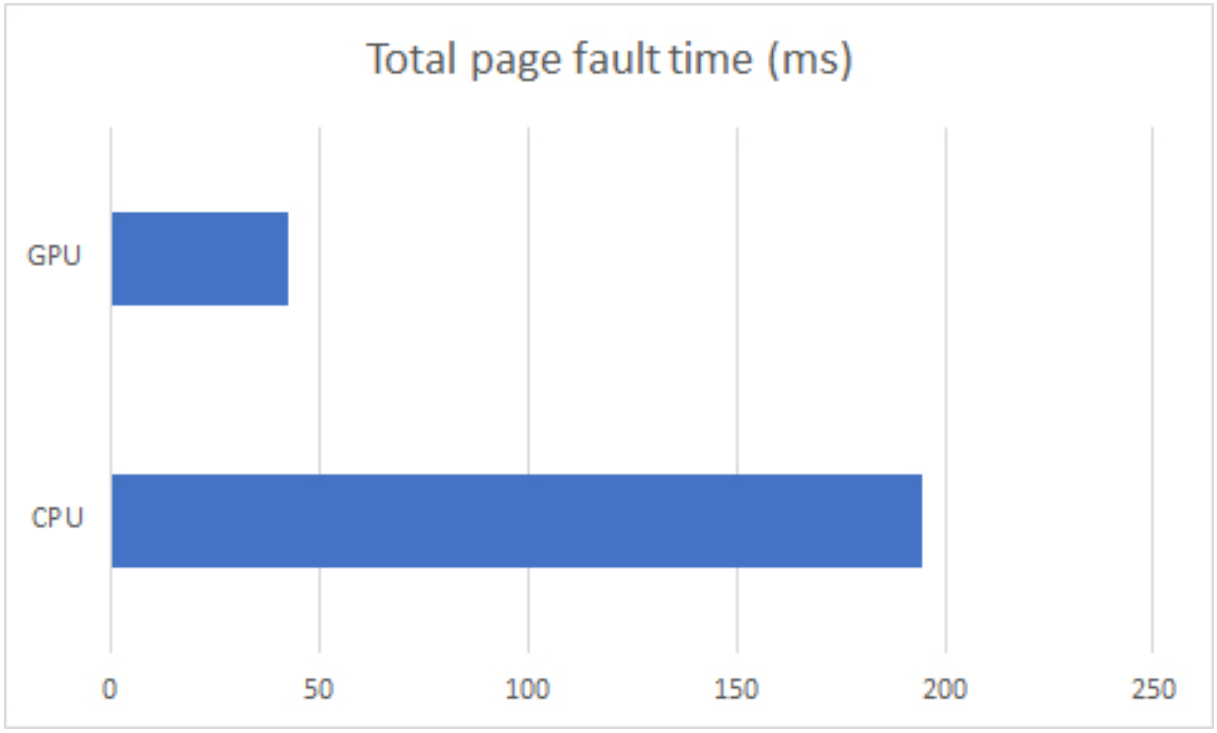

Figure 1: Time wasted on page faults on GPU vs time wasted on page faults on CPU (less is better)

2 Prefetching

2.1 Introduction

Prefetching is a technique for speeding up fetch operations by beginning a fetch operation whose result is expected to be needed soon. This can be done in CUDA with the function *cudaMemPrefetchAsync (const void* devPtr, size_t count, int dstDevice, cudaStream_t stream = 0)*

2.2 Example

The following code is used to check the impact of using prefetching in terms of performance:

```
__global__
void addVectorsInto(float *result, float *a, float *b, int N)
{
  int index = threadIdx.x + blockIdx.x * blockDim.x;
  int stride = blockDim.x * gridDim.x;
  for(int i = index; i < N; i += stride)
    result[i] = a[i] + b[i];
}
```

And the results verify that prefetching can improve performance:

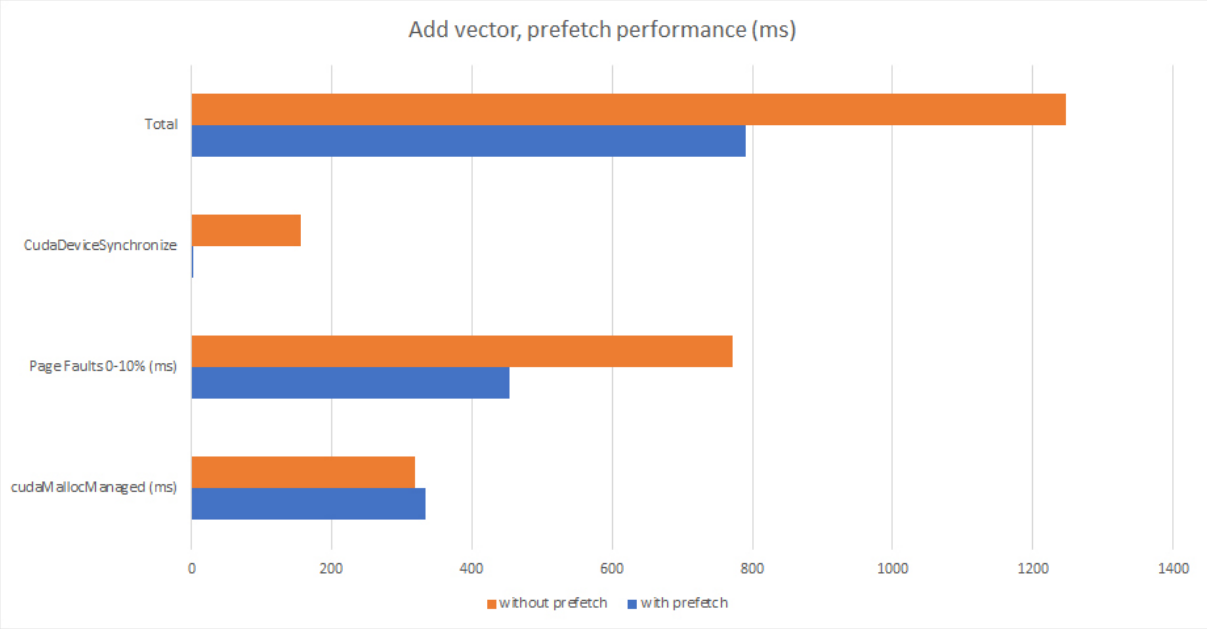

Figure 2: Time wasted on page faults on GPU using prefetch vs time wasted on page faults on GPU without prefetching (less is better)

3 Initializing on the CPU vs initializing on the GPU

An other way to improve our GPU projects can be doing the initialization of large structures of data on the GPU instead of doing it on the CPU as usual (the bigger the structure is, the most performance we gain). This is it because of the parallelism provided by the GPU.

The following graph shows the impact on the performance of this technique (the code use to test it is the same as the previous section).

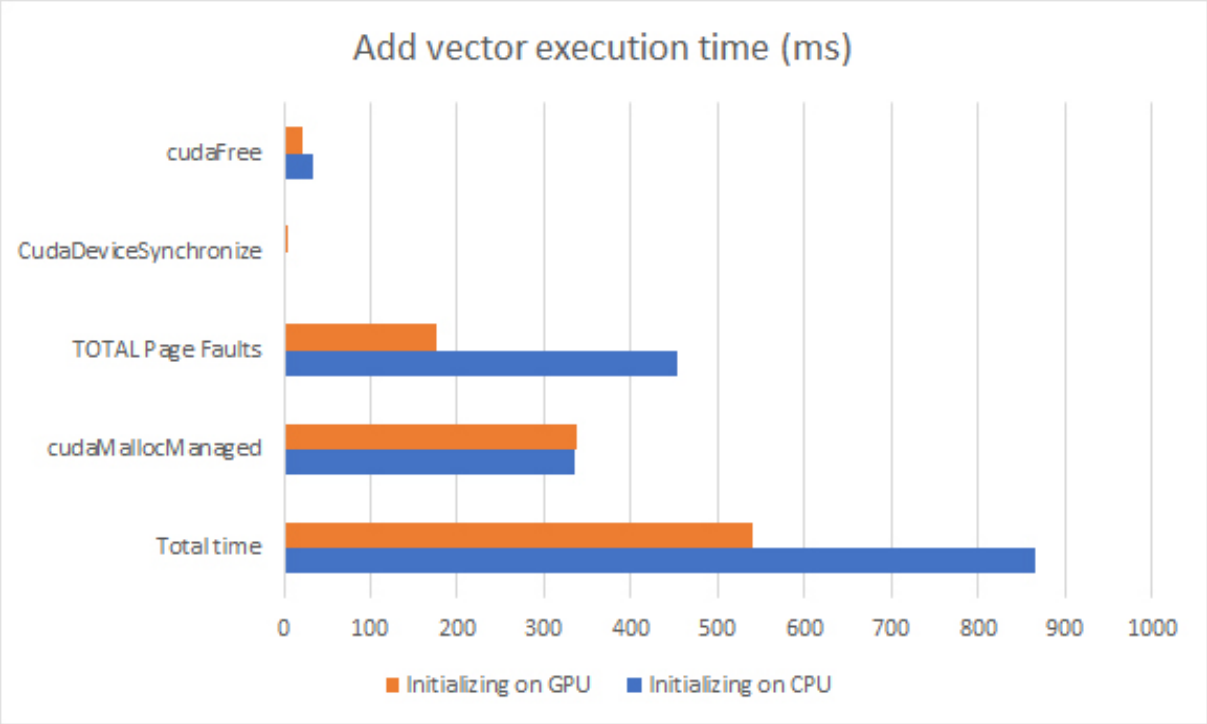

Figure 3: Comparison between times initializing on CPU and initializing on GPU (less is better)

As we can see in the graph, the execution time is around a 33% faster when we initialize the data on the GPU.

4 References

- NVIDIA CUDA Runtime API
- EVERYTHING YOU NEED TO KNOW ABOUT UNIFIED MEMORY - Nikolay Sakharnykh