AGH	Module: **Introduction to CUDA and OpenCL**	Report title: **Report 3**		
	Instructor: **dr hab. inż. Szumlak Tomasz**			
	Name and surname: **Marcin Filipek** **Anca Dărăbuț**	Group: **11**	Date: **6.11.2019**	Grade:

1. nvvp analysis of page faults

A page fault is an exception raised by computer when trying to initialize and use a part of memory which is not ready to be used. The computer tries to assign data, but there are no resources for calculations, even though the page may be accessible to the process. Nevertheless, page faults slow down the process, thus they have to be fixed.

1.1. GPU

The first case is when the unified memory is accessed only by GPU.

```
==40818== Profiling application: ./memout1
==40818== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:  100.00%  34.075ms         1  34.075ms  34.075ms  34.075ms  deviceKernel(int*, int)
      API calls:   89.37%  337.05ms         1  337.05ms  337.05ms  337.05ms  cudaMallocManaged
                   10.37%  39.103ms         1  39.103ms  39.103ms  39.103ms  cudaFree
                    0.13%  473.73us         1  473.73us  473.73us  473.73us  cuDeviceTotalMem
                    0.09%  339.78us        96  3.5390us     838ns  125.57us  cuDeviceGetAttribute
                    0.02%  68.445us         1  68.445us  68.445us  68.445us  cudaLaunchKernel
                    0.01%  49.866us         1  49.866us  49.866us  49.866us  cuDeviceGetName
                    0.01%  36.736us         1  36.736us  36.736us  36.736us  cudaDeviceSynchronize
                    0.00%  10.197us         1  10.197us  10.197us  10.197us  cuDeviceGetPCIBusId
                    0.00%  4.7490us         3  1.5830us     977ns  2.5850us  cuDeviceGetCount
                    0.00%  2.8640us         2  1.4320us  1.0480us  1.8160us  cuDeviceGet
                    0.00%  1.2570us         1  1.2570us  1.2570us  1.2570us  cuDeviceGetUuid

==40818== Unified Memory profiling result:
Device "GeForce GTX 1060 6GB (0)"
   Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
     399         -         -         -           -  34.81760ms  Gpu page fault groups
```

1.2. CPU

Secondly, the unified memory is accessed only by the CPU.

```
==40858== Profiling application: ./memout2
==40858== Profiling result:
No kernels were profiled.
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
      API calls:   96.59%  347.49ms         1  347.49ms  347.49ms  347.49ms  cudaMallocManaged
                    3.16%  11.368ms         1  11.368ms  11.368ms  11.368ms  cudaFree
                    0.13%  454.18us         1  454.18us  454.18us  454.18us  cuDeviceTotalMem
                    0.09%  332.37us        96  3.4620us     838ns  123.34us  cuDeviceGetAttribute
                    0.02%  75.219us         1  75.219us  75.219us  75.219us  cuDeviceGetName
                    0.01%  23.537us         1  23.537us  23.537us  23.537us  cudaDeviceSynchronize
                    0.00%  10.826us         1  10.826us  10.826us  10.826us  cuDeviceGetPCIBusId
                    0.00%  5.0980us         3  1.6990us  1.0470us  3.0030us  cuDeviceGetCount
                    0.00%  2.8640us         2  1.4320us  1.0480us  1.8160us  cuDeviceGet
                    0.00%  1.2570us         1  1.2570us  1.2570us  1.2570us  cuDeviceGetUuid

==40858== Unified Memory profiling result:
Total CPU Page faults: 384
```

1.3. GPU and CPU

Thirdly, the unified memory is accessed first by the GPU, then by the CPU.

```
==40890== Profiling application: ./memout3
==40890== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:  100.00%  33.719ms         1  33.719ms  33.719ms  33.719ms  deviceKernel(int*, int)
      API calls:   95.53%  293.22ms         1  293.22ms  293.22ms  293.22ms  cudaMallocManaged
                    4.17%  12.788ms         1  12.788ms  12.788ms  12.788ms  cudaFree
                    0.15%  446.64us         1  446.64us  446.64us  446.64us  cuDeviceTotalMem
                    0.10%  305.28us        96  3.1790us     838ns  99.733us  cuDeviceGetAttribute
                    0.03%  106.93us         1  106.93us  106.93us  106.93us  cudaLaunchKernel
                    0.01%  36.737us         1  36.737us  36.737us  36.737us  cuDeviceGetName
                    0.01%  23.676us         1  23.676us  23.676us  23.676us  cudaDeviceSynchronize
                    0.00%  9.9880us         1  9.9880us  9.9880us  9.9880us  cuDeviceGetPCIBusId
                    0.00%  3.7720us         3  1.2570us     908ns  1.6770us  cuDeviceGetCount
                    0.00%  2.3750us         2  1.1870us     838ns  1.5370us  cuDeviceGet
                    0.00%  1.1180us         1  1.1180us  1.1180us  1.1180us  cuDeviceGetUuid

==40890== Unified Memory profiling result:
Device "GeForce GTX 1060 6GB (0)"
   Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
       5  38.399KB  16.000KB  64.000KB  192.0000KB  24.89600us  Host To Device
     776  169.07KB  4.0000KB  0.9961MB  128.1250MB  10.84451ms  Device To Host
     396         -         -         -           -  34.01741ms  Gpu page fault groups
       3  4.0000KB  4.0000KB  4.0000KB  12.00000KB           -  Memory thrashes
Total CPU Page faults: 389
Total CPU thrashes: 3
```

1.4. CPU and GPU

Last, but not least, the memory is first accessed by the CPU, then by the GPU.

```
==40925== Profiling application: ./memout4
==40925== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:  100.00%  50.354ms         1  50.354ms  50.354ms  50.354ms  deviceKernel(int*, int)
      API calls:   84.51%  325.30ms         1  325.30ms  325.30ms  325.30ms  cudaMallocManaged
                   15.22%  58.572ms         1  58.572ms  58.572ms  58.572ms  cudaFree
                    0.13%  487.35us         1  487.35us  487.35us  487.35us  cuDeviceTotalMem
                    0.09%  350.18us        96  3.6470us     838ns  109.02us  cuDeviceGetAttribute
                    0.03%  119.57us         1  119.57us  119.57us  119.57us  cudaLaunchKernel
                    0.02%  70.819us         1  70.819us  70.819us  70.819us  cuDeviceGetName
                    0.01%  23.676us         1  23.676us  23.676us  23.676us  cudaDeviceSynchronize
                    0.00%  10.267us         1  10.267us  10.267us  10.267us  cuDeviceGetPCIBusId
                    0.00%  4.7500us         3  1.5830us     978ns  2.7240us  cuDeviceGetCount
                    0.00%  2.7240us         2  1.3620us     978ns  1.7460us  cuDeviceGet
                    0.00%  1.2570us         1  1.2570us  1.2570us  1.2570us  cuDeviceGetUuid

==40925== Unified Memory profiling result:
Device "GeForce GTX 1060 6GB (0)"
   Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
     796  164.66KB  4.0000KB  0.9961MB  128.0000MB  12.03318ms  Host To Device
     381         -         -         -           -  50.85418ms  Gpu page fault groups
Total CPU Page faults: 384
```

CPU page faults occur when the host function is launched, whilst GPU page fault groups appear when the kernel of the device is called. Through the host function (**hostFunction**), the CPU tries to access the unified memory and the kernel of the device (**deviceKernel**) is used so that the GPU can access the unified memory.

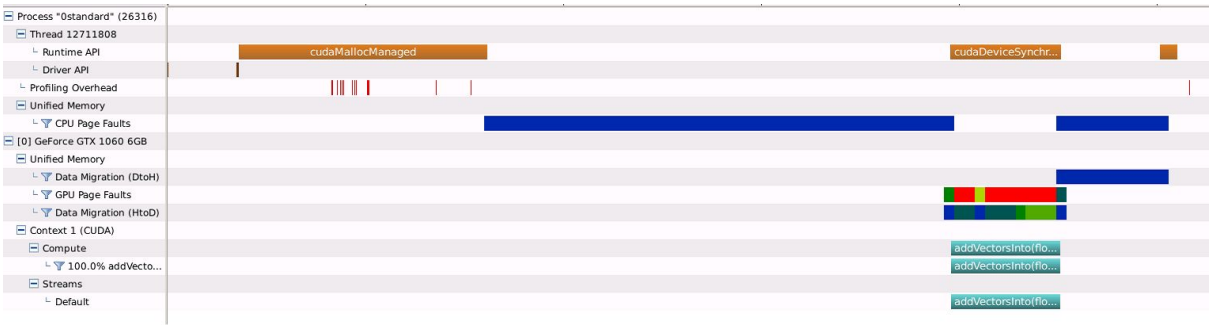
As it can be seen from the nvprof results, when the CPU tries to access the unified memory, the number of page faults varies between 384 and 389 (when GPU accessed the memory first, then the CPU). On the other hand, the number of page fault groups of the GPU decreases from 399 to 396 and 381, when the CPU tries to access the unified memory.

2. memory management analysis with nvvp

In order to study the memory management, we used NVIDIA Visual Profiler to see the time profiling on 4 different cases: classical vector addition, vector addition with asynchronous memory prefetch on the device, initialization of device kernel, asynchronous memory prefetch for the host.

2.1. vector_add_standard.cu

The first one was simple vector addition, the same as used in previous labs. We checked if there were page faults. We got 1536 page faults in total. The reason is that our code is not optimised regarding memory.
Below, there is the time profiling of the **vector_add_standard.cu** file.



It seems that the GPU page faults last as long as the data is sent from the host to device, the vector addition is done and while the synchronization is done.

2.2. vector_add_prefetch_gpu.cu

The next sample is very similar except from three lines with **cudaMemPrefetchAsync** function. Asynchronous prefetch figures out the size of

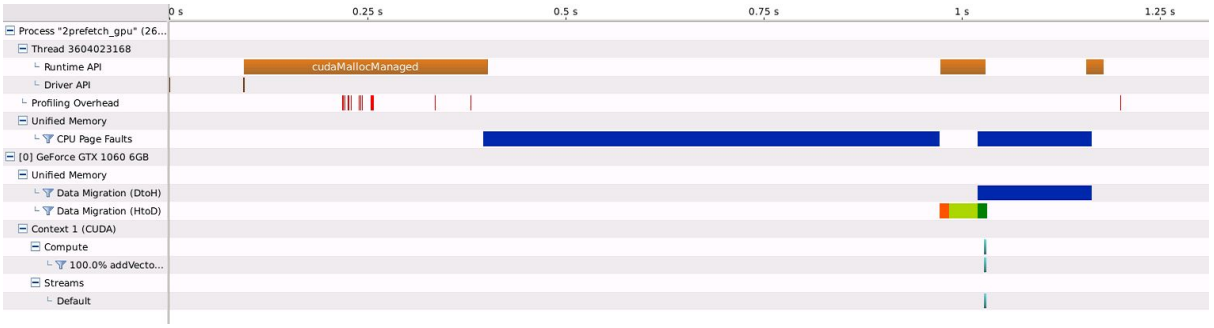
data that is wanted and predicts it. We anticipate and start to move the data as soon as possible, before the kernel does. In this manner, execution is sped up.

We compared times of these sample with previous one and noticed significant difference speed. The first program was much slower, which is caused by the fact that the kernel has to wait until it gets the new data portion. By using prefetching the data was already in the device, so the kernel did not have to wait.

Simple vector add result:
155.38ms addVectorsInto(float, float*, float*, int)*
Prefetched vector add result:
2.5611ms addVectorsInto(float, float*, float*, int)*

It means that over 80% of time was wasted for page faults.

The profiling of **vector_add_prefetch_gpu.cu** file.

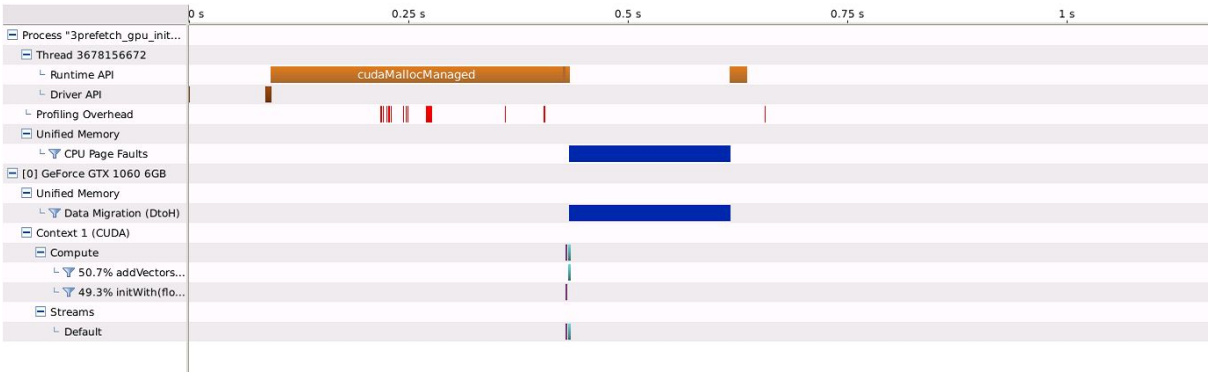


In this case, data requires a smaller amount of time to be sent from host to device, as well as for being processed in vector addition.

2.3. vector_add_prefetch_gpu_init_gpu.cu

Trying to get rid of all page faults, we launched the third sample.
In this case, there are two kernels, one for host and one for the device, thus no data should be shipped to the device, but it is initialised already on the GPU.
Consequently, there is one more kernel which allocates memory and creates data ready to be processed.

The profiling of **vector_add_prefetch_gpu_init_gpu.cu** file.



The amount of time taken by the CPU page faults was considerably decreased. Only one data transfer is left (from the device to the host) and the computation time is divided between the actual vector addition and the initialization of the device kernel (**initWith**).

2.4. vector_add_prefetch_gpucpu_init_gpu.cu

Unfortunately, the program still has to send the result back to the host. It is the last operation which generates page faults, but we can get rid of them in the same way we did in second sample: prefetch the data for CPU.

Time profiling of **vector_add_prefetch_gpucpu_init_gpu.cu** file.

As we can see, there are no page faults anymore. It means that our program does not waste time waiting for data and being idle at that time. Our memory management is optimized.