REPORT 3

Introduction to CUDA and OpenCL

AGH-Faculty of Physics and Applied Computer Science

authors:

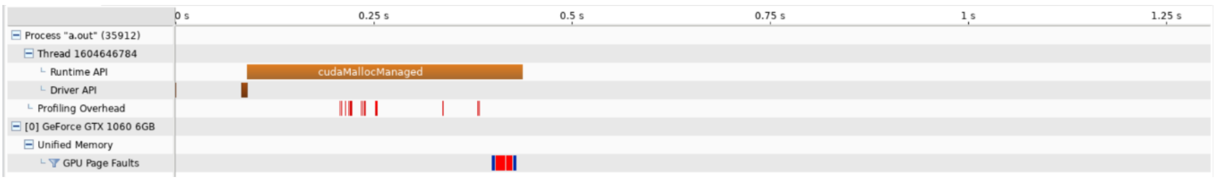
Kinga Pyrek
Aleksandra Rolka

On our last lab classes we conducted experiment to learn more about the behavior of cudaMallocManaged() and we worked with The NVIDIA Visual Profiler.The Visual Profiler is a graphical profiling tool that displays a timeline of our application's CPU and GPU activity.
We used it to show differences in the way of accessing UM (unified memory).

We also looked into a page fault,which is a type of exception raised by computer hardware. It occurs when a running program accesses a memory page that is not currently mapped by the memory management unit into the virtual address space of a process.
It means that the page may be accessible to the process, but requires a mapping to be added to the process page tables, and may additionally require the actual page contents to be loaded from a backing store such as a disk.
When handling a page fault, the operating system generally tries to make the required page accessible at the location physical memory, or terminates the program in case of an illegal memory access.

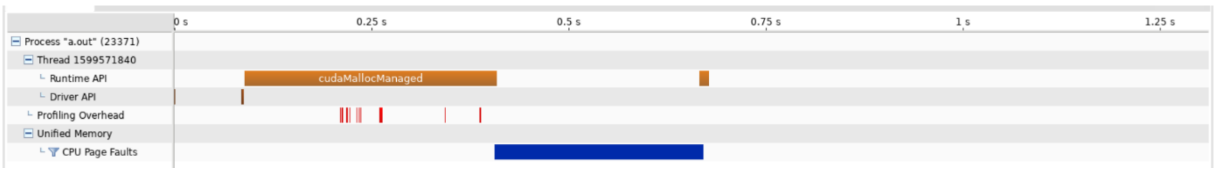
Contrary to what "fault" might suggest, valid page faults are not errors, and are common and necessary to increase the amount of memory available to programs in any operating system that utilizes virtual memory.

There are results of 4 versions of the program:



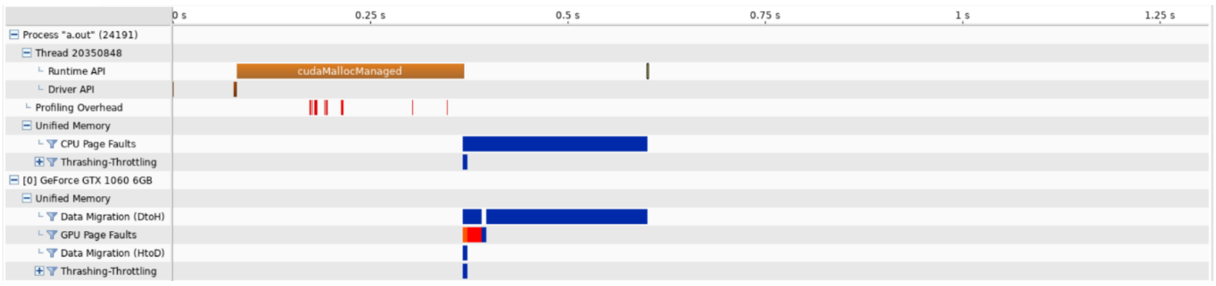
Visual profiling results when unified memory is accessed only by the GPU

Access to unified memory has only GPU,that's why we have only GPU page faults. It doesn't take too much time.



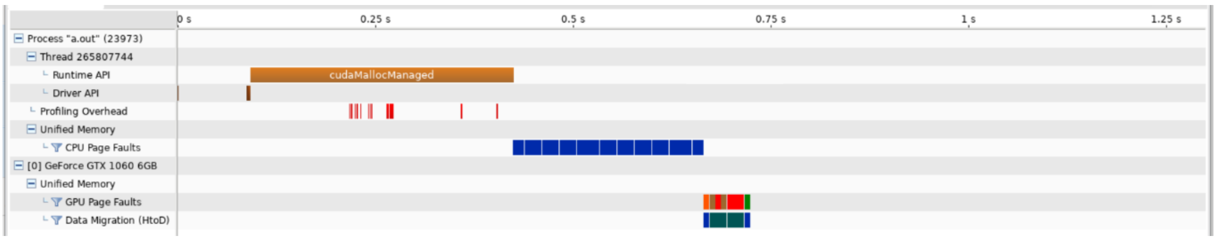
Visual profiling results when unified memory is accessed only by the CPU

This time only CPU has access to UM and there are only CPU page faults, but its time is much longer and it performs right after cudaMallocManaged is done.



Visual profiling results when unified memory is accessed first by the GPU then the CPU

In third case we gave access to UM first to GPU and then to CPU. What's new,that Data Migration(DtoH) appeared. It shows which one has first access and that GPU’s and CPU’s page fault are processed at the same time.



Visual profiling results when unified memory is accessed first by the CPU then the GPU

In last one we have reverse order access to unified memory and in this case GPU’s page fault are processed not at the same time, but after CPU’s page faults are done. We have shortest time of Data Migration, but overall time is longer.

Next we were analyzing files:

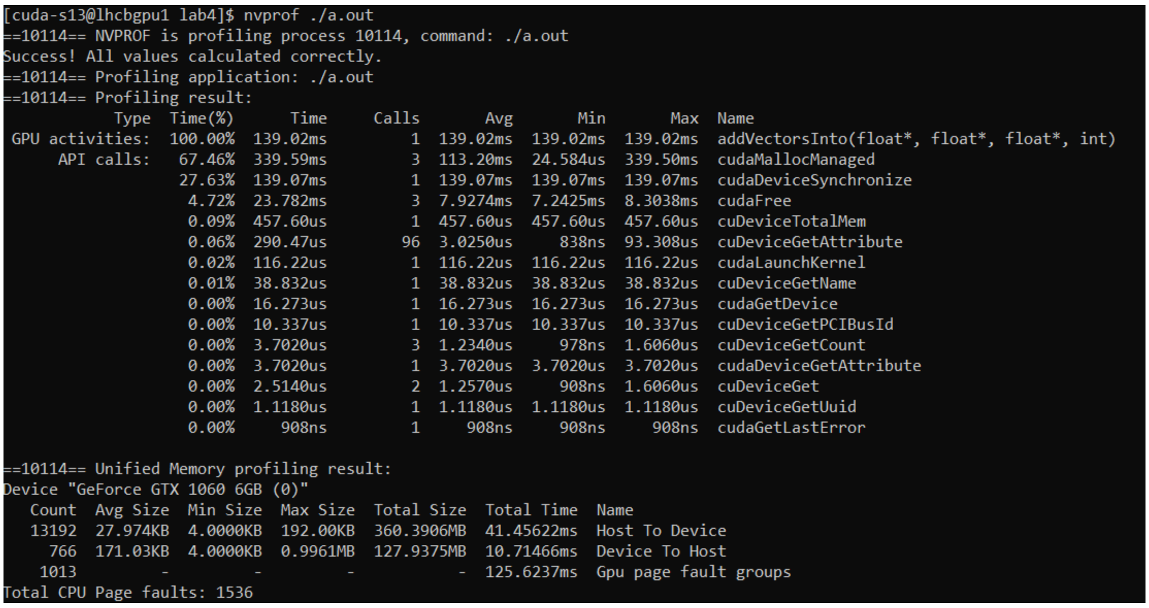
-vector_add_standard
-vector_add_prefetch_GPU
-vector_add_prefetch_GPU_init_GPU
-vector_add_prefetch_GPUCPU_init_GPU

to familiarize ourselves with data prefetching.

To **prefetch** data it means to fetch data in advance. The benefit of prefetching data is to leverage the asynchronous aspect of memory accesses in CUDA. When a memory access operation is executed, it does not block other operations following it as long as they don't use the data from the operation.
As written in (Kernel without prefetching), every addition waits for its data to be loaded from memory. Inside the loop of (Kernel with prefetching), the device first launches a memory load operation for the next iteration and does an addition in parallel. The time for the addition is actually

overlapping with the memory access time. But the increased register usage may lower the number of active warps on an SM.

These are the profiling results we've got:

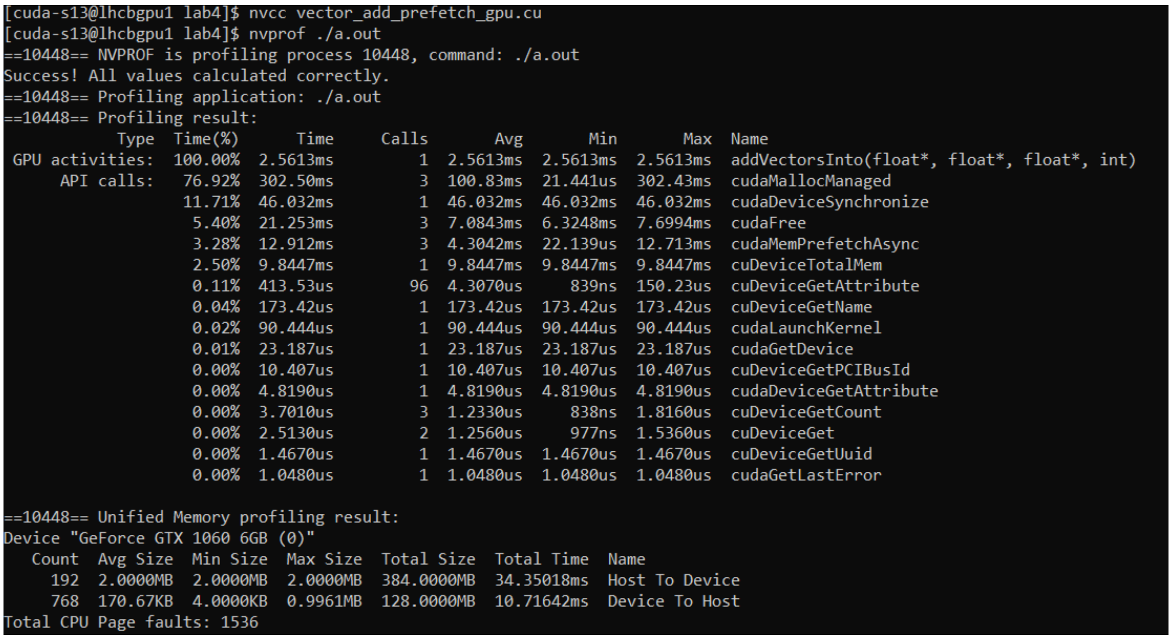


```
[cuda-s13@lhcbgpu1 lab4]$ nvprof ./a.out
==10114== NVPROF is profiling process 10114, command: ./a.out
Success! All values calculated correctly.
==10114== Profiling application: ./a.out
==10114== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:  100.00%  139.02ms         1  139.02ms  139.02ms  139.02ms  addVectorsInto(float*, float*, float*, int)
      API calls:   67.46%  339.59ms         3  113.20ms  24.584us  339.50ms  cudaMallocManaged
                   27.63%  139.07ms         1  139.07ms  139.07ms  139.07ms  cudaDeviceSynchronize
                    4.72%  23.782ms         3  7.9274ms  7.2425ms  8.3038ms  cudaFree
                    0.09%  457.60us         1  457.60us  457.60us  457.60us  cuDeviceTotalMem
                    0.06%  290.47us        96  3.0250us     838ns  93.308us  cuDeviceGetAttribute
                    0.02%  116.22us         1  116.22us  116.22us  116.22us  cudaLaunchKernel
                    0.01%  38.832us         1  38.832us  38.832us  38.832us  cuDeviceGetName
                    0.00%  16.273us         1  16.273us  16.273us  16.273us  cudaGetDevice
                    0.00%  10.337us         1  10.337us  10.337us  10.337us  cuDeviceGetPCIBusId
                    0.00%  3.7020us         3  1.2340us     978ns  1.6060us  cuDeviceGetCount
                    0.00%  3.7020us         1  3.7020us  3.7020us  3.7020us  cudaDeviceGetAttribute
                    0.00%  2.5140us         2  1.2570us     908ns  1.6060us  cuDeviceGet
                    0.00%  1.1180us         1  1.1180us  1.1180us  1.1180us  cuDeviceGetUuid
                    0.00%     908ns         1     908ns     908ns     908ns  cudaGetLastError

==10114== Unified Memory profiling result:
Device "GeForce GTX 1060 6GB (0)"
   Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
   13192  27.974KB  4.0000KB  192.00KB  360.3906MB  41.45622ms  Host To Device
     766  171.03KB  4.0000KB  0.9961MB  127.9375MB  10.71466ms  Device To Host
    1013         -         -         -           -  125.6237ms  Gpu page fault groups
Total CPU Page faults: 1536
```

vector_add_standard's profiling results

This is the simplest version of our program. As we can see there are a lot of page faults(in GPU activity also).

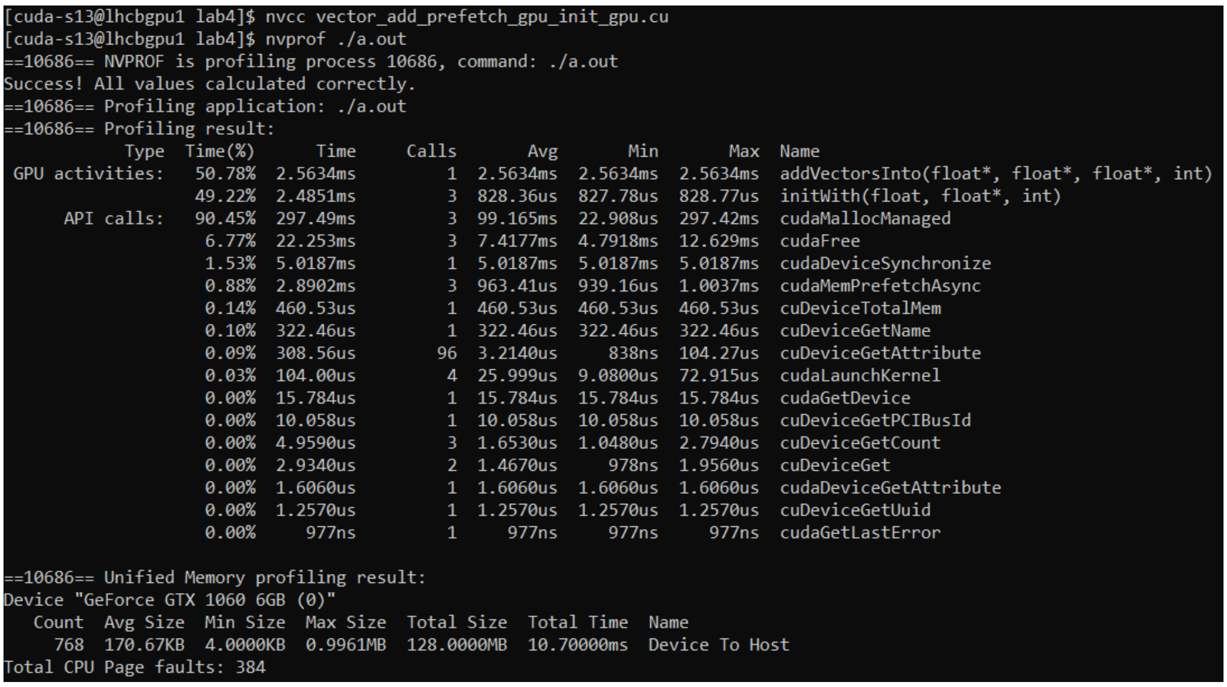


```
[cuda-s13@lhcbgpu1 lab4]$ nvcc vector_add_prefetch_gpu.cu
[cuda-s13@lhcbgpu1 lab4]$ nvprof ./a.out
==10448== NVPROF is profiling process 10448, command: ./a.out
Success! All values calculated correctly.
==10448== Profiling application: ./a.out
==10448== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:  100.00%  2.5613ms         1  2.5613ms  2.5613ms  2.5613ms  addVectorsInto(float*, float*, float*, int)
      API calls:   76.92%  302.50ms         3  100.83ms  21.441us  302.43ms  cudaMallocManaged
                   11.71%  46.032ms         1  46.032ms  46.032ms  46.032ms  cudaDeviceSynchronize
                    5.40%  21.253ms         3  7.0843ms  6.3248ms  7.6994ms  cudaFree
                    3.28%  12.912ms         3  4.3042ms  22.139us  12.713ms  cudaMemPrefetchAsync
                    2.50%  9.8447ms         1  9.8447ms  9.8447ms  9.8447ms  cuDeviceTotalMem
                    0.11%  413.53us        96  4.3070us     839ns  150.23us  cuDeviceGetAttribute
                    0.04%  173.42us         1  173.42us  173.42us  173.42us  cuDeviceGetName
                    0.02%  90.444us         1  90.444us  90.444us  90.444us  cudaLaunchKernel
                    0.01%  23.187us         1  23.187us  23.187us  23.187us  cudaGetDevice
                    0.00%  10.407us         1  10.407us  10.407us  10.407us  cuDeviceGetPCIBusId
                    0.00%  4.8190us         1  4.8190us  4.8190us  4.8190us  cudaDeviceGetAttribute
                    0.00%  3.7010us         3  1.2330us     838ns  1.8160us  cuDeviceGetCount
                    0.00%  2.5130us         2  1.2560us     977ns  1.5360us  cuDeviceGet
                    0.00%  1.4670us         1  1.4670us  1.4670us  1.4670us  cuDeviceGetUuid
                    0.00%  1.0480us         1  1.0480us  1.0480us  1.0480us  cudaGetLastError

==10448== Unified Memory profiling result:
Device "GeForce GTX 1060 6GB (0)"
   Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
     192  2.0000MB  2.0000MB  2.0000MB  384.0000MB  34.35018ms  Host To Device
     768  170.67KB  4.0000KB  0.9961MB  128.0000MB  10.71642ms  Device To Host
Total CPU Page faults: 1536
```

vector_add_prefetch_GPU's profiling results

Here was used cudaMemPrefetchAsync function, which prefetches memory to the specified destination device. As a result of using that we do not have page faults in GPU, but we still have the same number of CPU page faults as in the previous sample.



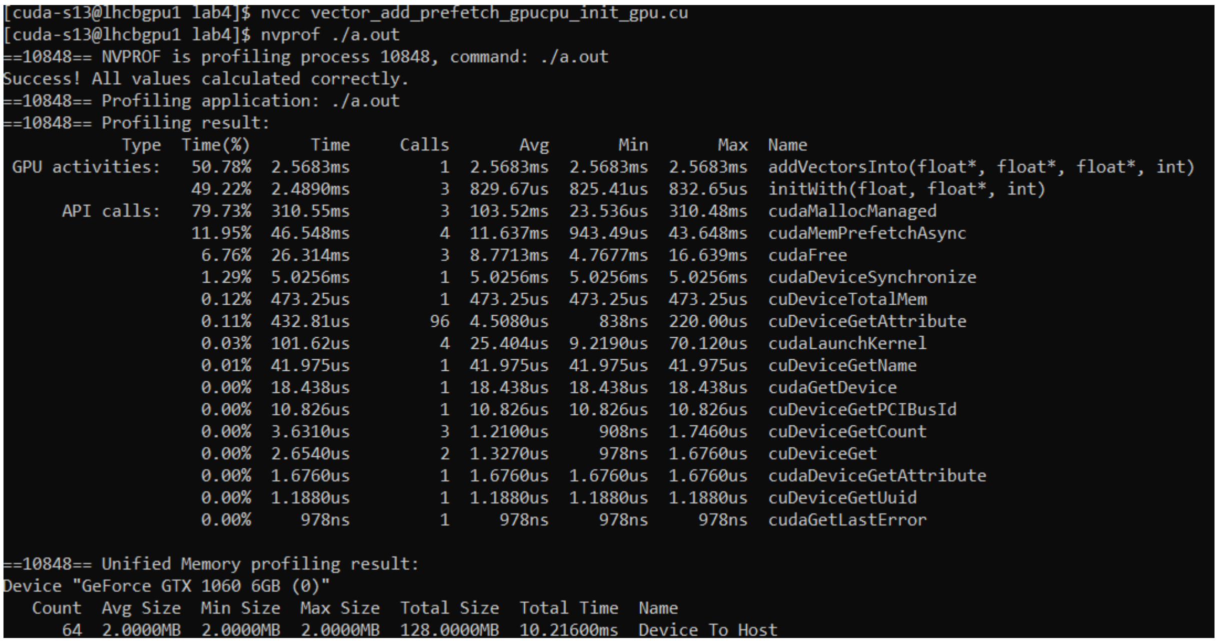
```
[cuda-s13@lhcbgpu1 lab4]$ nvcc vector_add_prefetch_gpu_init_gpu.cu
[cuda-s13@lhcbgpu1 lab4]$ nvprof ./a.out
==10686== NVPROF is profiling process 10686, command: ./a.out
Success! All values calculated correctly.
==10686== Profiling application: ./a.out
==10686== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   50.78%  2.5634ms         1  2.5634ms  2.5634ms  2.5634ms  addVectorsInto(float*, float*, float*, int)
                   49.22%  2.4851ms         3  828.36us  827.78us  828.77us  initWith(float, float*, int)
      API calls:   90.45%  297.49ms         3  99.165ms  22.908us  297.42ms  cudaMallocManaged
                    6.77%  22.253ms         3  7.4177ms  4.7918ms  12.629ms  cudaFree
                    1.53%  5.0187ms         1  5.0187ms  5.0187ms  5.0187ms  cudaDeviceSynchronize
                    0.88%  2.8902ms         3  963.41us  939.16us  1.0037ms  cudaMemPrefetchAsync
                    0.14%  460.53us         1  460.53us  460.53us  460.53us  cuDeviceTotalMem
                    0.10%  322.46us         1  322.46us  322.46us  322.46us  cuDeviceGetName
                    0.09%  308.56us        96  3.2140us     838ns  104.27us  cuDeviceGetAttribute
                    0.03%  104.00us         4  25.999us  9.0800us  72.915us  cudaLaunchKernel
                    0.00%  15.784us         1  15.784us  15.784us  15.784us  cudaGetDevice
                    0.00%  10.058us         1  10.058us  10.058us  10.058us  cuDeviceGetPCIBusId
                    0.00%  4.9590us         3  1.6530us  1.0480us  2.7940us  cuDeviceGetCount
                    0.00%  2.9340us         2  1.4670us     978ns  1.9560us  cuDeviceGet
                    0.00%  1.6060us         1  1.6060us  1.6060us  1.6060us  cudaDeviceGetAttribute
                    0.00%  1.2570us         1  1.2570us  1.2570us  1.2570us  cuDeviceGetUuid
                    0.00%     977ns         1     977ns     977ns     977ns  cudaGetLastError

==10686== Unified Memory profiling result:
Device "GeForce GTX 1060 6GB (0)"
   Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
     768  170.67KB  4.0000KB  0.9961MB  128.0000MB  10.70000ms  Device To Host
Total CPU Page faults: 384
```

vector_add_prefetch_GPU_init_GPU's profiling results

In the third code there is initWith function, which initializes space for data on GPU, due to that the Host to Device transfer is gone.

Even though it is slower because of initializing space on GPU it repays the execution time raise bacause we save the memory. Also total amount of page faults is only 384, so it’s 5 times smaller than in the previous cases.



```
[cuda-s13@lhcbgpu1 lab4]$ nvcc vector_add_prefetch_gpucpu_init_gpu.cu
[cuda-s13@lhcbgpu1 lab4]$ nvprof ./a.out
==10848== NVPROF is profiling process 10848, command: ./a.out
Success! All values calculated correctly.
==10848== Profiling application: ./a.out
==10848== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   50.78%  2.5683ms         1  2.5683ms  2.5683ms  2.5683ms  addVectorsInto(float*, float*, float*, int)
                   49.22%  2.4890ms         3  829.67us  825.41us  832.65us  initWith(float, float*, int)
      API calls:   79.73%  310.55ms         3  103.52ms  23.536us  310.48ms  cudaMallocManaged
                   11.95%  46.548ms         4  11.637ms  943.49us  43.648ms  cudaMemPrefetchAsync
                    6.76%  26.314ms         3  8.7713ms  4.7677ms  16.639ms  cudaFree
                    1.29%  5.0256ms         1  5.0256ms  5.0256ms  5.0256ms  cudaDeviceSynchronize
                    0.12%  473.25us         1  473.25us  473.25us  473.25us  cuDeviceTotalMem
                    0.11%  432.81us        96  4.5080us     838ns  220.00us  cuDeviceGetAttribute
                    0.03%  101.62us         4  25.404us  9.2190us  70.120us  cudaLaunchKernel
                    0.01%  41.975us         1  41.975us  41.975us  41.975us  cuDeviceGetName
                    0.00%  18.438us         1  18.438us  18.438us  18.438us  cudaGetDevice
                    0.00%  10.826us         1  10.826us  10.826us  10.826us  cuDeviceGetPCIBusId
                    0.00%  3.6310us         3  1.2100us     908ns  1.7460us  cuDeviceGetCount
                    0.00%  2.6540us         2  1.3270us     978ns  1.6760us  cuDeviceGet
                    0.00%  1.6760us         1  1.6760us  1.6760us  1.6760us  cudaDeviceGetAttribute
                    0.00%  1.1880us         1  1.1880us  1.1880us  1.1880us  cuDeviceGetUuid
                    0.00%     978ns         1     978ns     978ns     978ns  cudaGetLastError

==10848== Unified Memory profiling result:
Device "GeForce GTX 1060 6GB (0)"
   Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
      64  2.0000MB  2.0000MB  2.0000MB  128.0000MB  10.21600ms  Device To Host
```

vector_add_prefetch_GPUCPU_init_GPU's profiling results

The last version of code is the most efficient one. The memory is initialized on CPU and GPU and we use prefetching data. In the wake of refinements there is no page faults.