Assignment IV

Adam Orucu

January 26, 2024

Url to code: `https://github.com/adamorucu/gpu-programming-course`

Exercise 1

1. **Assume X=800 and Y=600. Assume that we decided to use a grid of 16X16 blocks. That is, each block is organized as a 2D 16X16 array of threads. How many warps will be generated during the execution of the kernel? How many warps will have control divergence? Please explain your answers.**

   There will be $ceil(800/16) * ceil(600/16) = 1900$ blocks. Since each block has $16 * 16 = 256$ threads there will be $256 * 1900 = 486,400$ threads in total. Given that each warp has 32 threads the number of warps is $486,400/32 = 15,200$.

   There will be extra threads because bottom edge will be out of bounds. $ceil(600/16) * 16 - 600 = 8$, there will be $8 * 800 = 6400$ threads out of bounds. These however will be covered by other. blocks so there will not execute in the if statement so there will not be divergence.

2. **Now assume X=600 and Y=800 instead, how many warps will have control divergence? Please explain your answers.**

   600 is not a multiple of 16, therefore both right edge will be out of bounds. $ciel(800/16) = 50$ blocks will be out of bounds which means that last block in each row will have partially filled rows which will leave them divergent. This means that $50 * 256/32 = 400$ warps will have control divergence.

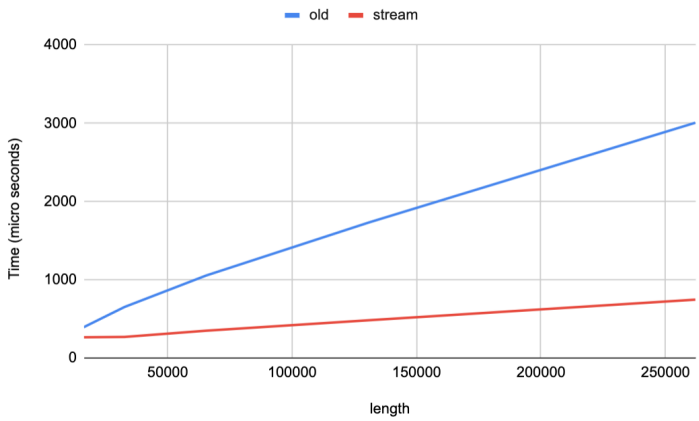
3. **Now assume X=600 and Y=799, how many warps will have control divergence? Please explain your answers.**

   Previous calculation is repeated to get 400 warps. Additionally having 799 pixels on the other axis means that there will be an extra row, which gives extra 38 warps. However these two axes have one overlapping warp therefore the answer is $400 + 38 - 1 = 437$

Exercise 2

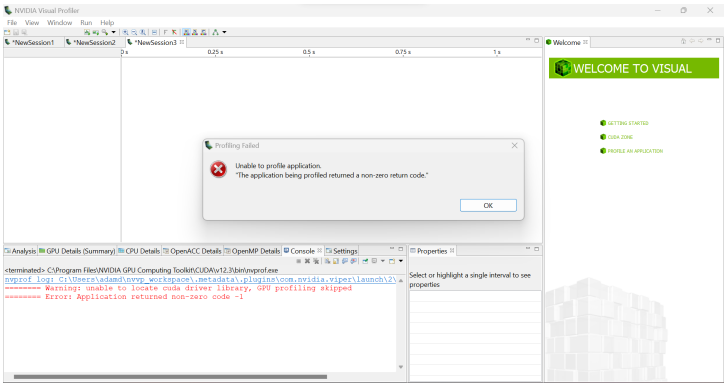
1. **Compared to the non-streamed vector addition, what performance gain do you get? Present in a plot (you may include comparison at different vector length)**

   Experiment for segment size 4096. In my experiments the performance gain increased with the length of the input. The average was around 3X improvement.



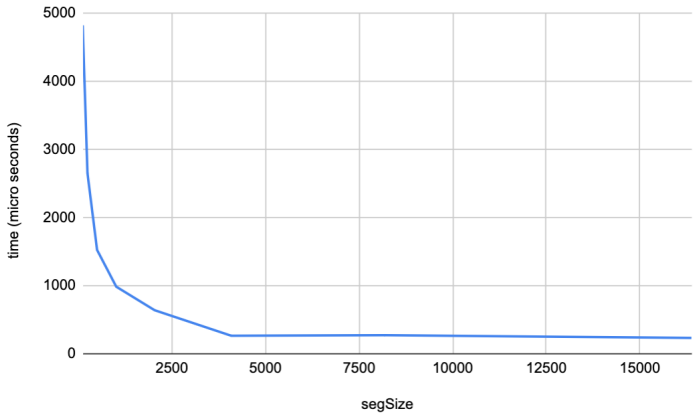
2. **Use nvprof to collect traces and the NVIDIA Visual Profiler (nvvp) to visualize the overlap of communication and computation.**

   I wasn't able to make nvvp work on two different operating systems.



3. **What is the impact of segment size on performance? Present in a plot (you may choose a large vector and compare 4-8 different segment sizes)**

   The execution time gets smaller as segment size grows. Experiment for 65,536 input length.

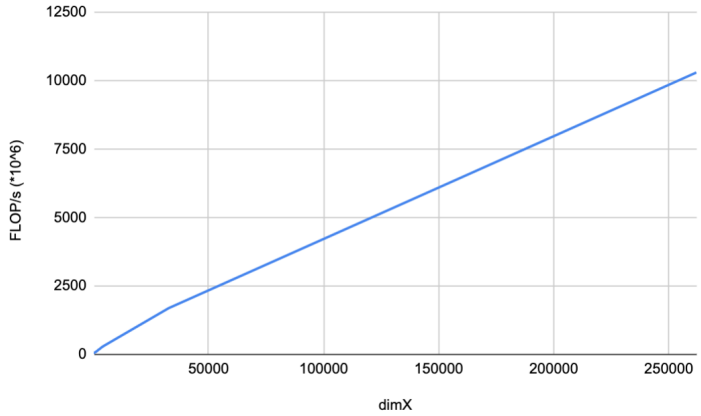


Exercise 3

1. **Run the program with different dimX values. For each one, approximate the FLOPS (floating-point operation per second) achieved in computing the SMPV (sparse matrix**

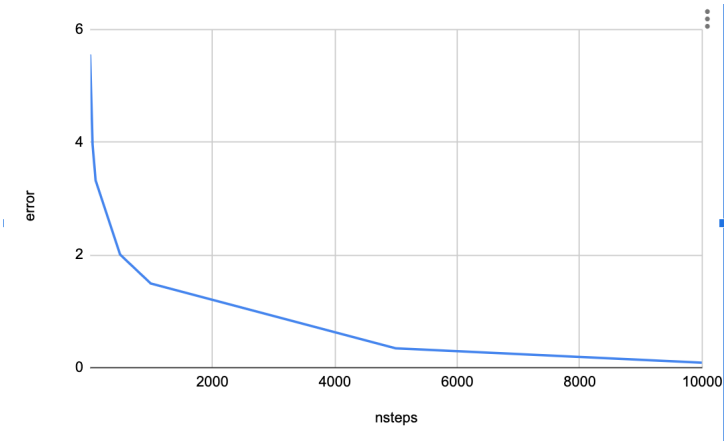
multiplication). Report FLOPS at different input sizes in a FLOPS. What do you see compared to the peak throughput you report in Lab2?

There is a linear relationship between dimX and the number of FLOPs. The peak seems to be around 21,000 * 10**6 FLOPs where it stops increasing with higher dimX.



2. **Run the program with dimX=128 and vary nsteps from 100 to 10000. Plot the relative error of the approximation at different nstep. What do you observe?**

   As expected the approximations are better as the number of steps grows and the error convergece to zero.



3. **Compare the performance with and without the prefetching in Unified Memory. How is the performance impact?**

   There is a clear and significant differnce in the time for initialising the sparce matrix, wether prefetching is used.

```
The X dimension of the grid is 4096
The number of time steps to perform is 5000
Timing - Allocating device memory.             Elasped 211623 microseconds
Timing - Prefetching GPU memory to the host.           Elasped 469 microseconds
Timing - Initializing the sparse matrix on the host.           Elasped 76 microseconds
Timing - Initializing memory on the host.              Elasped 4 microseconds
Timing - Prefetching GPU memory to the device.         Elasped 548 microseconds
Timing - Steping time.         Elasped 885091 microseconds
FLOP/s: 231.320847
The relative error of the approximation is 7.999193
The X dimension of the grid is 4096
The number of time steps to perform is 5000
Timing - Allocating device memory.             Elasped 162686 microseconds
Timing - Initializing the sparse matrix on the host.           Elasped 504 microseconds
Timing - Initializing memory on the host.              Elasped 6 microseconds
Timing - Steping time.         Elasped 729189 microseconds
FLOP/s: 280.777686
 * 10^6The relative error of the approximation is 7.999193
```