LAB 6

DUE DATE: Fri 03 Nov 5pm (upload to polylearn)

Names in Group (max 2): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Write a program in Cuda to perform Matrix Matrix multiplication. The requirements for the program:

|  |  |
| --- | --- |
|  |  |

{\displaystyle \mathbf {A} \mathbf {B} ={\begin{pmatrix}a&b&c\\x&y&z\end{pmatrix}}{\begin{pmatrix}\alpha &\rho \\\beta &\sigma \\\gamma &\tau \\\end{pmatrix}}={\begin{pmatrix}a\alpha +b\beta +c\gamma &a\rho +b\sigma +c\tau \\x\alpha +y\beta +z\gamma &x\rho +y\sigma +z\tau \\\end{pmatrix}}\,,}

1. Implement MM in the GPU using Shared Memory. You are provided the code for MM on the GPU not using the shared memory, add the kernel to do this in Shared MemoryMeasure the time it takes for your code to perform matrix multiplication.

The code for shared memory multiplication can be found in the class slides and on the book. The pourpose of this hw is for you to get familiar with one of the most important parallel patterns Matrix Multliplication, and all you can do and SHOULD do to speed it up. SO read pages 90 to 99; I will explain the algorithm in class, but is not trivial and if you don’t read ahead is will be “difficult” to understand.

2. Use the flollowing specific cuda functions to measure time

cudaEvent\_t start, stop;

cudaEventCreate(&start);

cudaEventCreate(&stop);

cudaMemcpy(d\_x, x, N\*sizeof(float), cudaMemcpyHostToDevice);

cudaMemcpy(d\_y, y, N\*sizeof(float), cudaMemcpyHostToDevice);

cudaEventRecord(start);

saxpy<<<(N+255)/256, 256>>>(N, 2.0f, d\_x, d\_y);

cudaEventRecord(stop);

cudaMemcpy(y, d\_y, N\*sizeof(float), cudaMemcpyDeviceToHost);

cudaEventSynchronize(stop);

float milliseconds = 0;

cudaEventElapsedTime(&milliseconds, start, stop);

**What to turn in**

Upload a report in PDF with the following:

1. Your name
2. Explanation of the matrix multiplication code used if different than the one implemented in book
3. Prove that the code is correct by printing the addition of all values in the result matrix and compare it with the sum of all values in your cpu implementation
4. Table with execution times (only for the matrix multiplication, do not add the reading and writing of the matrix from/to a file)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matrix A and B Size | Cpu Execution time No threads | CPU 4 threads execution time | Execution Time GPU  Non Shared | Execution Time GPU  Shared |
| A. 512,512 |  |  |  |  |
| B.1024,1024 |  |  |  |  |
| C. |  |  |  |  |
| D Irregular matrix for extra credit |  |  |  |  |

1. Run the profiler and write here a summary of what you see is different between both implementations (non shared vs shared) and hopefully you see the improvement (you can take screen captures of the program running).
2. What can you do to improve the performance of the matrix multiplication. You don’t need to implemented just think about it and you are allowed to “google” fast algorithms for m\*m. The purpose of this question is for you to get an idea of all the things people are doing for matrix multiplication and important it is
3. Question:5.2
4. Question:5.6
5. Question:5.8
6. Appendix with your code, just the two kernel functions (non-shared and shared), clean and properly commented (I may ask for a demo of your code working on the computer labs if I don’t understand how your code is able to run).