Assignment 4 - COL380

Aayush Goyal 2019CS10452

April 2023

Contents

1 Introduction

In this assignment we have to do the matric multiplication as we had previously done. But this time we have to parallelize it using CUDA.

2 Implementation

I took my implementation of Assignment1 and then decided to parallelize the code using CUDA instead of openMP. For this I went through multiple approaches

2.1 Approach 1

In this approach I simple sent all the data from the host to the device and then did the multiplication on the device and then sent the data back to the host. This approach was very slow and took a lot of time. The reason for this is that the data transfer between the host and the device is very slow and hence the overall time taken was very high. So next I was thinking about a way to reduce the time of transiiton between.

2.2 Approach 2

Now in the 2nd approach I allocated all the arrays as a contiguous array. The 2D arrays were stored in the form of 1D array and then easily sent over for multiplication. This approach was faster than the previous one but still not as fast as I wanted it to be. So I did some basic optimizations within this approach.

2.2.1 Optimization 1

I noticed that while writing the output it was taking a lot of time. Since I had already stored the entire matrix in the form of contiguous 1D array so for writing the matrix to the file I just wrote the entire block in one go instead of doing for loops. This seems to have to reduced the time taken by a lot.

3 Final results

Here is the final result of the time taken by the code for different sizes of the matrix. The time is in miliseconds.

n	m	kA	kB	Time (in ms)
24	4	12	12	91
24	8	5	5	73
8	4	4	4	74
4096	8	52450	52450	3984
32768	8	16800	16800	11662
32768	4	65300	65300	26204

Thus the performance has improved significantly and thanks to CUDA we can now do large computations as well. One more thing I noticed while doing the work was that we cannot use cout statements in the code but we can use printf statements. This is because cout statements are not thread safe and hence can cause problems. Overall with the help of CUDA, I was able to parallelize my code very well and run it faster.