INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DESIGN AND MANUFACTURING KANCHEEPURAM

LAB ASSIGNMENT 5 - REPORT
ON
MATRIX ADDITION
AND
MATRIX MULTIPLICATION
IN CUDA

SUBMITTED BY
AMAR KUMAR
(CED17I029)
TO
DR. NOOR MAHAMMAD

Strategy

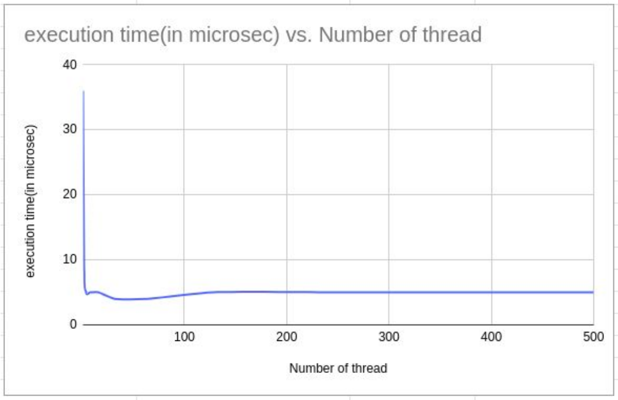
In my program for matrix addition, the instruction which is running in parallel is in the "for" loop i.e **C[i] = A[i] + B[i];**

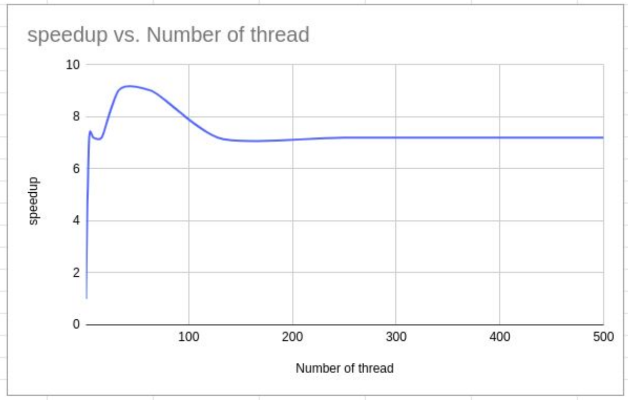
Instead of running the program serially, we can distribute the task of adding matrices between threads so that my program could run parallely and in turn save the execution time.
Therefore , firstly, i converted the matrix in an array and divided the task of adding. For example, if my matrix size is 4*4 then i have converted it in array of size 16. Suppose i have 4 thread then thread1 will execute 1,5,9,13, thread2 will execute 2,6,10,14, thread3 will execute 3,7,11,15 and thread4 will execute 4,8,12,16. In this way matrix addition will happen with the help of threads in CUDA

Graph and tables

https://docs.google.com/spreadsheets/d/1DM3hmeEPAiy6AFi0OVakTaHoM5Bg47a1P4glVVslYhU/edit#gid=0

Question1			
Number of thread	execution time(in microsec)	speedup	parallelization fraction(f)
1	36	1	0
2	9	4	1.5
4	5	7.2	1.148148148
8	5	7.2	0.9841269841
16	5	7.2	0.9185185185
32	4	9	0.917562724
64	4	9	0.9029982363
128	5	7.2	0.8678915136
256	5	7.2	0.8644880174
500	5	7.2	0.8628367847





Calculation of parallelization fraction

T(1) =36 microsecond
Here , for P = 32 the execution time is minimum
T(P) = 4 microsecond
Speedup = $\frac{T(1)}{T(P)}$ = $\frac{36}{4}$ = 9

From Amdahl's Law,
Speedup = $\frac{1}{(f/P) + (1-f)}$ Where , f = Parallelization factor P = Thread Number

So, f = $\frac{(1-T(P)/T(1))}{(1-(1/P))}$
Therefore, f = 0.917562724 which means that approx 91% of the program is parallelizable.

MATRIX MULTIPLICATION

Strategy

n my program for vector addition, the instruction which is running in parallel is in the "for" loop i.e **C[x][y] += A[x*N+k] * B[k*N+y];**

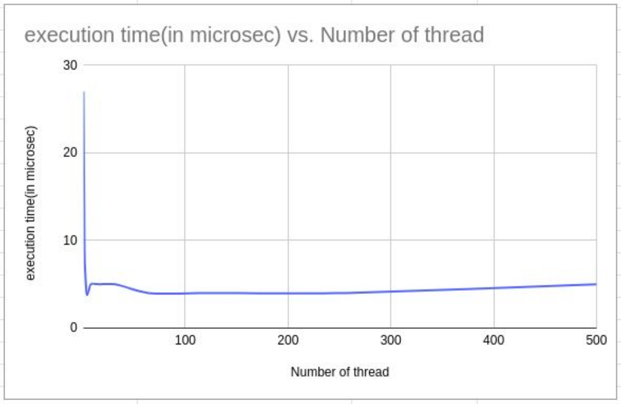
Instead of running the program serially, we can distribute the task of multiplying matrices between threads so that my program could run parallely and in turn save the execution time.
Therefore , firstly, i converted the matrix in an array and divided the task of multiplying among the threads. For example, if my matrix size is 4*4 then i have converted it in array of size 16. Suppose i have 4 thread then thread1 will execute 1,5,9,13, thread2 will execute 2,6,10,14, thread3 will execute 3,7,11,15 and thread4 will execute 4,8,12,16. x and y is the position of element which needs to be calculated. k is varying as the column of A or row of B. In this matrix multiplication is happening.

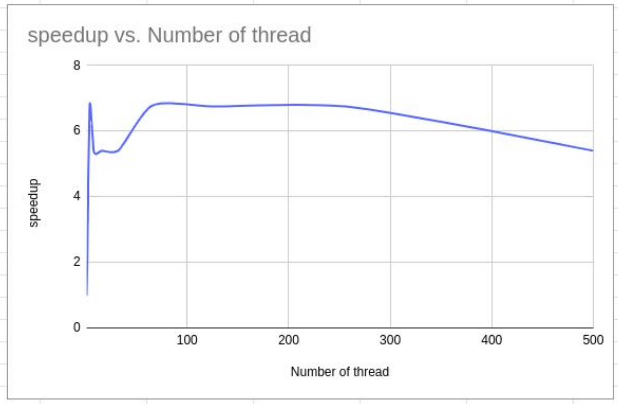
Graph and tables

https://docs.google.com/spreadsheets/d/1DM3hmeEPAiy6AFi0OVakTaHoM5Bg47a1P4glVVslYhU/edit#gid=0

Question2

Number of thread	execution time(in microsec)	speedup	parallelization fraction(f)
1	27	1	0
2	9	3	1.333333333
4	4	6.75	1.135802469
8	5	5.4	0.9312169312
16	5	5.4	0.8691358025
32	5	5.4	0.8410991637
64	4	6.75	0.8653733098
128	4	6.75	0.8585593467
256	4	6.75	0.8551924473
500	5	5.4	0.8164477102





Calculation of parallelization fraction

T(1) =27 seconds
Here , for P = 64 the execution time is minimum
T(P) = 4 seconds
Speedup = $\frac{T(1)}{T(P)}$ = $\frac{27}{4}$ = 6.750.8448150377

From Amdahl's Law,
Speedup = $\frac{1}{(f/P)+(1-f)}$ Where , f = Parallelization factor P = Thread Number

So, f = $\frac{(1-T(P)/T(1))}{(1-(1/P))}$
Therefore, f = 0.8653733098 which means that approx 86% of the program is parallelizable.