INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DESIGN AND MANUFACTURING KANCHEEPURAM

LAB ASSIGNMENT 2 - REPORT
ON
MATRIX ADDITION, MATRIX MULTIPLICATION (Column major order) and MATRIX MULTIPLICATION (Block based approach) IN OPENMP

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

MATRIX ADDITION

Strategy

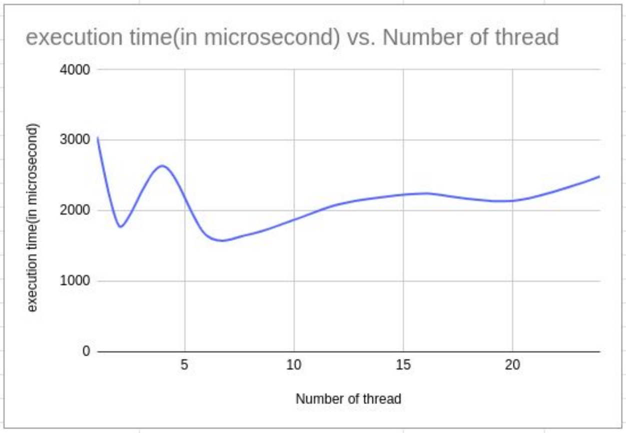
In my program for matrix addition, the instruction which is running in parallel is
c[row][col] = a[row][col] + b[row][col];

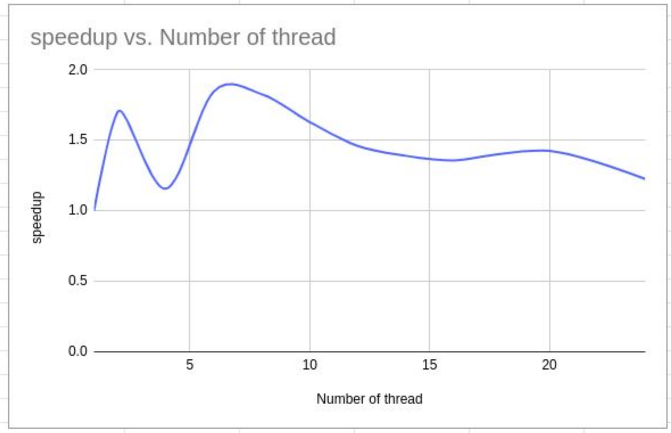
I am first traversing all columns of 1st row, then 2nd row and so on in both the matrices. After that I am adding matrix a and b element wise.
Instead of running the program serially, we can distribute the task of adding elements of matrix among the threads so that my program could run parallely and in turn save the execution time.
Therefore , in #pragma omp parallel block , for loop consists of these granular instructions. We also have shared a,b and c to run the program in parallel.

Graph and tables

https://docs.google.com/spreadsheets/d/1FfjGPtIBFa86WQXOMp2RSFA2-rXr9PHSzDc44sj_CTQ/edit#gid=0

Size of matrix = 500 x 500			
Number of thread	execution time(in microsecond)	speedup	parallelization fraction(f)
1	3049	1	0
2	1788	1.705257271	0.8271564447
4	2637	1.156238149	0.1801683612
6	1651	1.84675954	0.5502131847
8	1669	1.826842421	0.517265614
10	1871	1.629609834	0.4292846471
12	2088	1.460249042	0.3438385164
14	2193	1.390332877	0.3023437697
16	2246	1.357524488	0.2809227069
20	2140	1.424766355	0.3138216154
24	2488	1.225482315	0.1919945242





Calculation of parallelization fraction

T(1) = 3049 microseconds
Here , for P = 6 the execution time is minimum
T(P) = 1651 microseconds
Speedup = $\frac{T(1)}{T(P)}$ = $\frac{3049}{1651}$ = 1.84675954.
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 = 0.5502131847 which means that approx. 55% of the program is parallelizable.

MATRIX MULTIPLICATION
(Column-major order)

Strategy

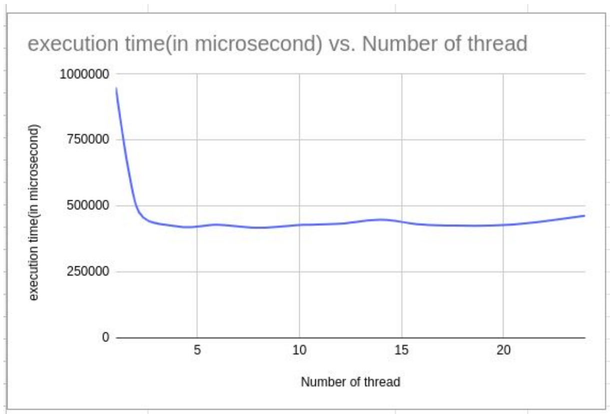
In my program for matrix multiplication, the instruction which is running in parallel is
c[row][col] += a[row][dot] * b[dot][col];

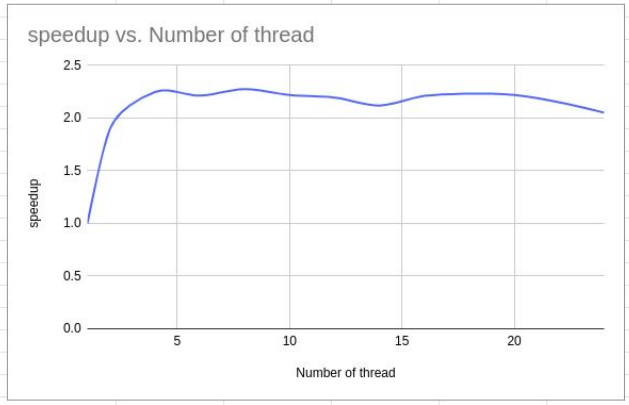
Here, for each row of matrix a, i am traversing each column of matrix b and multiplying elements from row i of matrix a to column j of matrix b. After finishing all column of matrix b, I am going to the next row of matrix a.
Instead of running the program serially, we can distribute the task of multiplying matrices among the threads so that my program could run parallelly and in turn save the execution time.
Therefore , in #pragma omp parallel block , for loop consists of these granular instructions. We also have shared a,b and c to run the program in parallel.

Graph and tables

https://docs.google.com/spreadsheets/d/1FfjGPtIBFa86WQXOMp2RSFA2-rXr9PHSzDc44sj_CTQ/edit#gid=0

Size of matrix A = 500 x 500 ; Size of matrix B = 500 x 500			
Number of thread	execution time(in microsecond)	speedup	parallelization fraction(f)
1	949723	1	0
2	501115	1.89521966	0.9447133533
4	422728	2.246652694	0.7398578322
6	429192	2.212816176	0.6577046149
8	417570	2.274404291	0.6403707788
10	428024	2.21885455	0.6103522349
12	432760	2.194572049	0.5938148664
14	448243	2.118768168	0.5686451572
16	429676	2.210323593	0.5840827273
20	428019	2.21888047	0.5782339748
24	463067	2.050940793	0.5346979662





Calculation of parallelization fraction

T(1) = 949723 microseconds
Here , for P = 8 the execution time is minimum
T(P) =417570 microseconds
Speedup = $\frac{T(1)}{T(P)}$ = $\frac{949723}{417570}$ = 2.274404291.
From Amdahl's Law,
Speedup = $\frac{1}{(f/P)+(1-f)}$ Where , f = Parallelization factor P = Thread Number
f = $\frac{(1-T(P)/T(1))}{(1-(1/P))}$
Therefore, f = 0.6403707788 which means that approx. 64% of the program is parallelizable.

MATRIX MULTIPLICATION
(Block based approach)

Strategy

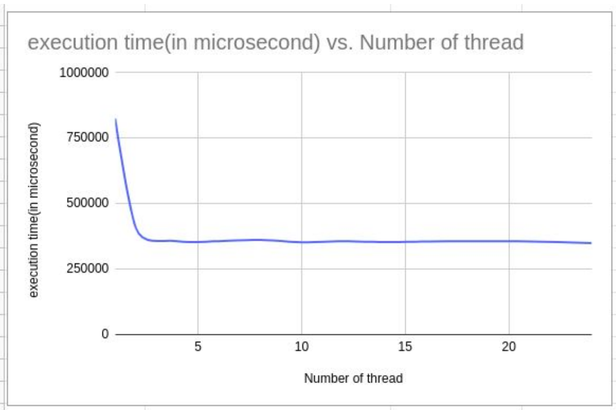
In my program for vector multiplication, the instruction which is running in parallel
c[blockRow][blockCol] = a[blockRow][dot] * b[dot][blockCol];

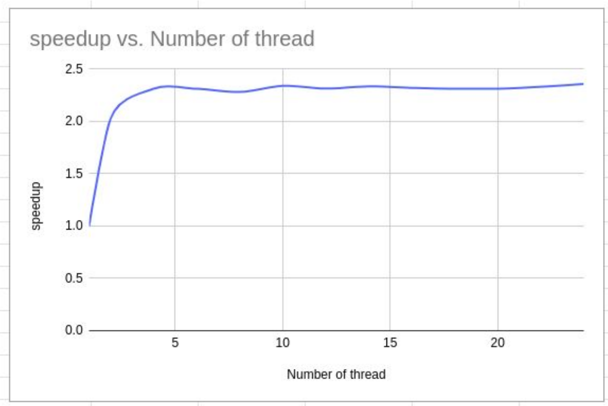
Here, I am traversing using block row and block column for all rows and columns and multiplying elements. Rows and columns are increasing with size of block.
For each block row of matrix a, I am doing the multiplication for each column block of matrix b. After all such block column of matrix b is finished, i am shifting to next block row of matrix a. In this way multiplication is divided in block size and multiplication is being done.
For doing block based approach, i took help from this link
https://www.youtube.com/watch?v=G92BCtfTwOE
Instead of running the program serially, we can distribute the task of multiplying matrices among the threads so that my program could run parallely and in turn save the execution time.
Therefore , in #pragma omp parallel block , for loop consists of these granular instructions. We also have shared a,b and c to run the program in parallel.

Graph and tables

https://docs.google.com/spreadsheets/d/1FfjGPtIBFa86WQXOMp2RSFA2-rXr9PHSzDc44sj_CTQ/edit#gid=0

Size of matrix A = 500 x 500 ; Size of matrix B = 500 x 500			
Number of thread	execution time(in microsecond)	speedup	parallelization fraction(f)
1	824487	1	0
2	406800	2.026762537	1.013204574
4	356567	2.31229194	0.7567048763
6	356686	2.311520497	0.68086119
8	361401	2.281363361	0.6419035629
10	352354	2.339939379	0.6362650014
12	356299	2.314031193	0.6194767722
14	353090	2.335061882	0.6157262731
16	355301	2.320531043	0.607001768
20	356438	2.313128791	0.5975632823
24	349695	2.357731738	0.600901082





Calculation of parallelization fraction

T(1) = 824487 microseconds

Here , for P = 24 the execution time is minimum

T(P) =349695 microseconds

Speedup = $\frac{T(1)}{T(P)}$ = $\frac{824487}{349695}$ = 2.357731738.

From Amdahl's Law,

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

f = $\frac{(1-T(P)/T(1))}{(1-(1/P))}$

Therefore, f = 0.600901082 which means that approx. 60% of the program is parallelizable.