INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DESIGN AND MANUFACTURING KANCHEEPURAM

LAB ASSIGNMENT 1 - REPORT
ON
VECTOR ADDITION
AND
VECTOR MULTIPLICATION
IN OPENMP

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

VECTOR ADDITION

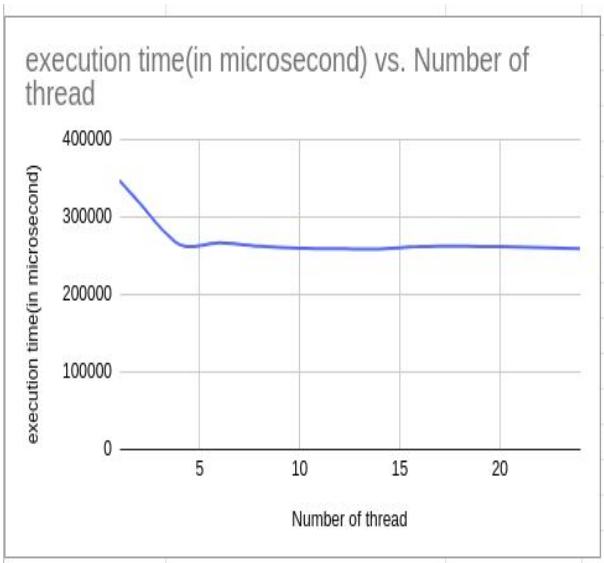
Strategy

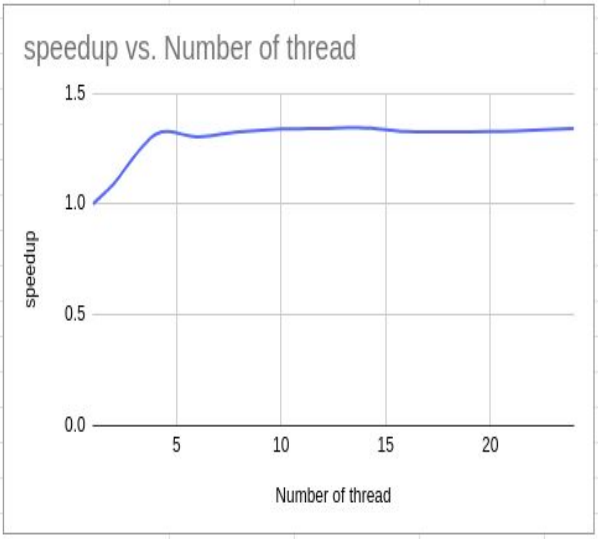
In my program for vector addition, the instruction which is running in parallel is in the "for" loop i.e **c[i] = a[i] + b[i]**
Instead of running the program serially, we can distribute the task of adding vectors among the threads so that my program could run parallely and in turn save the execution 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/1AIJJlhm_46Ftp-pGPQZYYIVDDx4ZIDHwsgXEnL_hKro/edit?usp=sharing

Number of iteration = 8 * 10^6			
Number of thread	execution time(in microsecond)	speedup	parallelization fraction(f)
1	347478	1	0
2	318566	1.090756703	0.1664105353
4	264350	1.314461888	0.3189765491
6	266640	1.303172817	0.2791704799
8	262143	1.325528433	0.2806673064
10	259679	1.338105892	0.2807499883
12	259104	1.341075398	0.2774506587
14	258599	1.343694291	0.275458723
16	261839	1.327067396	0.2628893532
20	261786	1.327336068	0.2595908382
24	259277	1.340180579	0.2648680667





Calculation of parallelization fraction

T(1) = 347478 microseconds
Here , for P = 14 the execution time is minimum
T(P) = 258599 microseconds

Speedup = $\frac{T(1)}{T(P)}$ = $\frac{347478}{258599}$ = 1.343694291.

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 = 0.275458723 which means that approx. 28% of the program is parallelizable.

VECTOR MULTIPLICATION

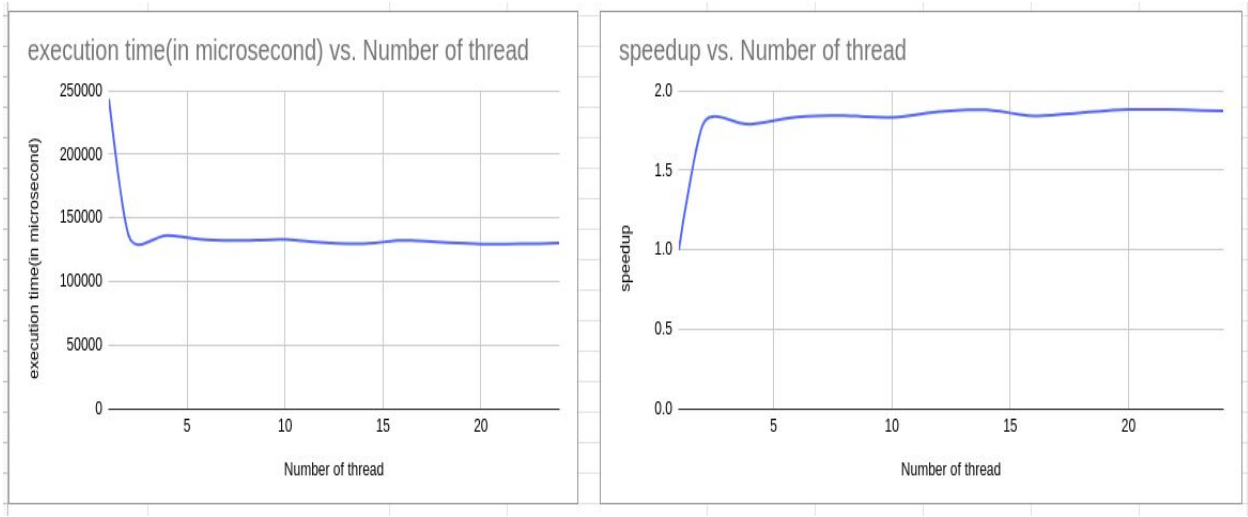
Strategy

In my program for vector multiplication, 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 multiplying vectors 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/1AIJJlhm_46Ftp-pGPQZYYIVDDx4ZIDHwsgXEnL_hKro/edit?usp=sharing

Number of iteration = 4 * 10^6			
Number of thread	execution time(in microsecond)	speedup	parallelization fraction(f)
1	243774	1	0
2	136795	1.782038817	0.8776899915
4	136265	1.788970022	0.5880255209
6	132842	1.835067223	0.546073002
8	132247	1.843323478	0.5228589947
10	133019	1.832625414	0.5048163919
12	130478	1.868314965	0.5070091001
14	129660	1.880101805	0.504122671
16	132368	1.841638462	0.4874722762
20	129511	1.882264827	0.4933948744
24	130182	1.872563027	0.4862322586



Calculation of parallelization fraction

T(1) = 243774 microseconds
Here , for P = 20 the execution time is minimum
T(P) =129511 microseconds

Speedup = $\frac{T(1)}{T(P)}$ = $\frac{243774}{129511}$ = 1.882264827.

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 = 0.4933948744 which means that approx. 50% of the program is parallelizable.