HIGH PERFORMANCE COMPUTING PRACTICE

EXPERIMENT - 1

OpenMP:

1) Hello World Program

2) Vector Addition, and

3) Vector Multiplication.

Use input as a larger double number (64-bit).

Run experiment for Threads = {1, 2, 4, 6, 8, 10, 12, 14, 16, 20, 24}.

Estimate the parallelization fraction.

Document the report and submit.

Bhaavanaa Thumu - CED17I021

Specifications

The size of all the vectors used for below two programs is 1,00,000.

All the elements of the vectors are produced randomly and are large double numbers (64-bit).

The number of threads and the approximate number of iterations in each case, for the given input size, is as shown in the table below.

No. of threads	*Approx. no. of iterations*
1	100000
2	50000
4	25000
6	16667
8	12500
10	10000
12	8334
14	7143
16	6250
20	5000
24	4167

Formulae used

Speed-up = T(1)/T(P)

Parallel fraction, f = (1 - T(P)/T(1))/(1 - (1/P))

where,

T(1) - time taken for serial execution

T(P) - time taken for parallel execution

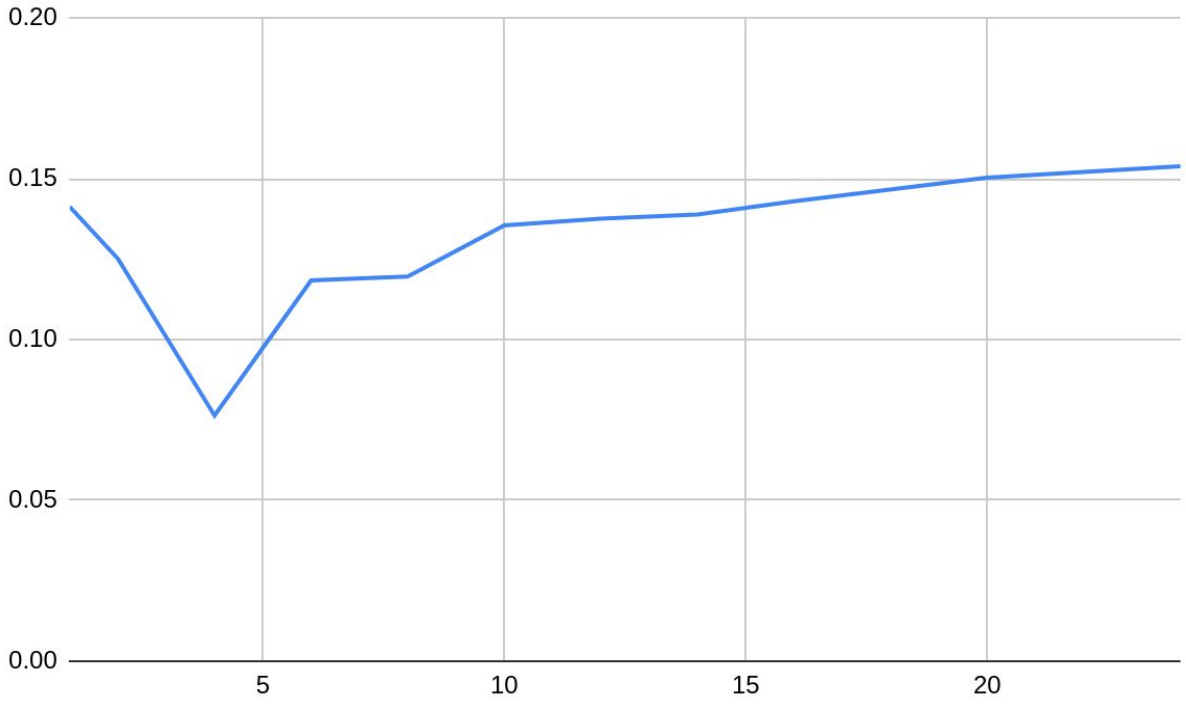
P - number of processes/threads

1. Vector Addition

In this program, two vectors, each of size 100000, are added.
The number of threads initialized and the execution time in each case is as shown in the table.

No. of threads	*Execution time*	*Speed-up*	*Parallel fraction*
1	0.141305	1	#DIV/0!
2	0.125141	1.1291663	0.2287817133
4	**0.076378**	**1.850074629**	**0.6126416852**
6	0.118397	1.193484632	0.1945408867
8	0.119568	1.181796133	0.1758061336
10	0.135476	1.043026071	0.04583466025
12	0.137625	1.026739328	0.02841049825
14	0.138889	1.017395186	0.01841298011
16	0.143029	0.9879465004	-0.01301392968
20	0.150322	0.9400154335	-0.06717086407
24	0.153896	0.9181850081	-0.09297926317

The plot below is the graph between no. of processes/threads and execution time.



x-axis : no. of processes/threads

y-axis : execution time

Hence, in this case, the optimal number of threads is 4, as we get the maximum speed-up and parallel fraction.

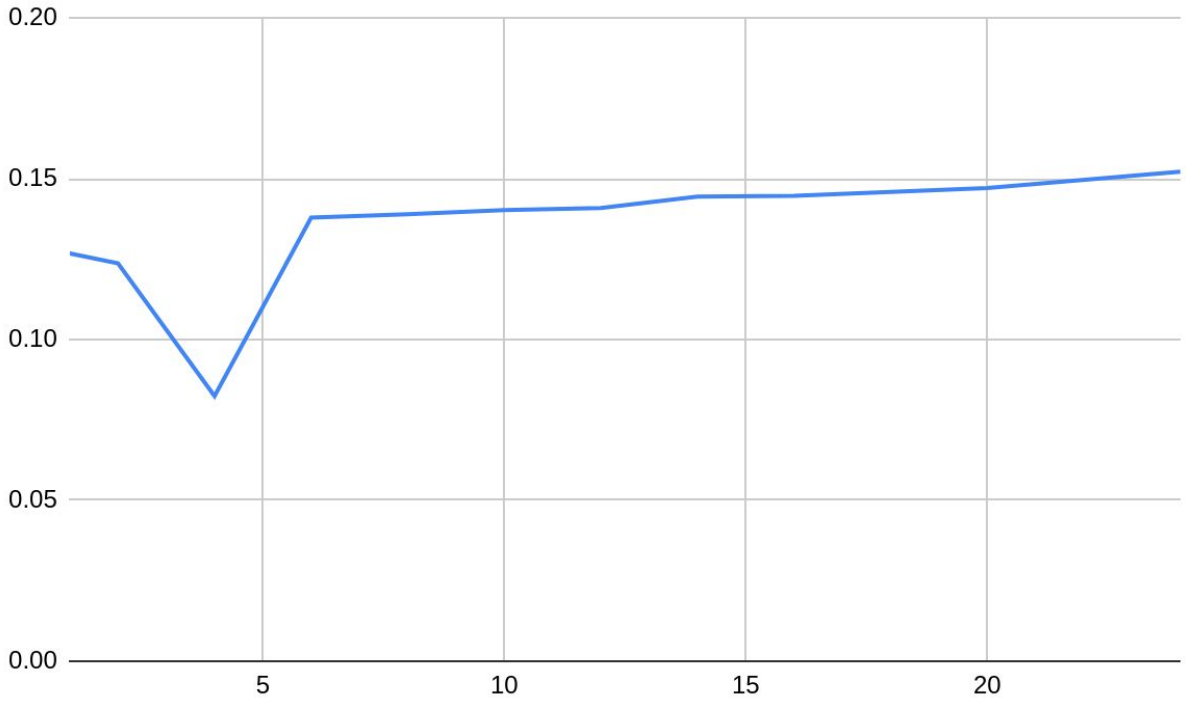
In this case, the loop iterations are being parallelized. The addition to be performed for the ith element of the vector is done as per the thread rank and the iteration of the loop.

2. Vector Multiplication

In this program, two vectors, each of size 100000, are multiplied.
The number of threads initialized and the execution time in each case is as shown in the table.

No. of threads	*Execution time*	*Speed-up*	*Parallel fraction*
1	0.126795	1	#DIV/0!
2	0.123698	1.025036783	0.2492056191
4	**0.082409**	**1.53860622**	**0.5557340505**
6	0.137907	0.9194239596	0.02885672835
8	0.138950	0.9125224901	0.01904694506
10	0.140258	0.9040126053	0.008232782515
12	0.140891	0.899951026	0.003196181053
14	0.144451	0.8777717011	-0.02397650472
16	0.144704	0.876237008	-0.02565797389
20	0.147086	0.8620466938	-0.04306474051
24	0.152217	0.832988431	-0.08058055117

The plot below is the graph between no. of processes/threads and execution time.



x-axis : no. of processes/threads

y-axis : execution time

Hence, in this case, the optimal number of threads is 4, as we get the maximum speed-up and parallel fraction.

In this case, the loop iterations are being parallelized. The multiplication to be performed for the ith element of the vector is done as per the thread rank and the iteration of the loop.