HIGH PERFORMANCE COMPUTING PRACTICE

EXPERIMENT - 3

OpenMP:

1) Addition of N-numbers. Identify the challenge.
    a) Implement using reduction
    b) Critical section
2) Vector dot product.
    a) Implement using reduction
    b) Critical section

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 four programs is 1,00,000.

All the elements of the matrices 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

1a. Addition of N numbers using reduction

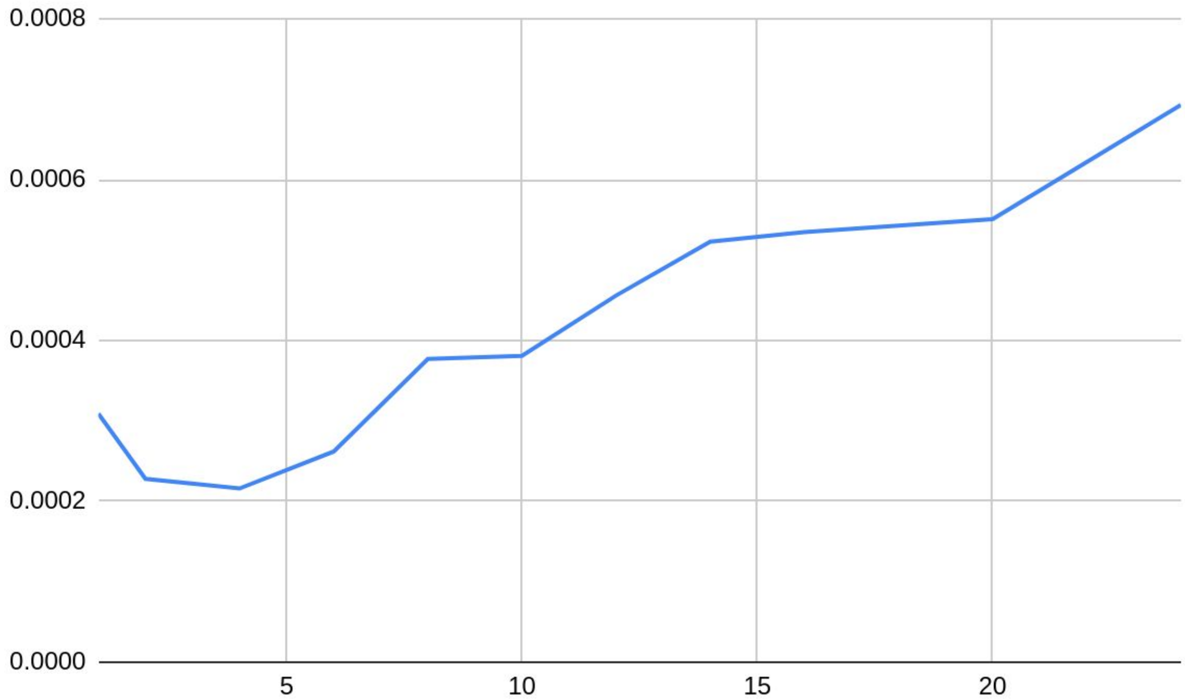
This program contains an array of size 1,00,000.

Therefore, here, N = 1,00,000.

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	0.000309	1	#DIV/0!
2	0.000228	1.355263158	0.5242718447
4	**0.000216**	**1.430555556**	**0.4012944984**
6	0.000262	1.179389313	0.1825242718
8	0.000377	0.8196286472	-0.2515025428
10	0.000381	0.811023622	-0.2588996764
12	0.000456	0.6776315789	-0.5189761695
14	0.000523	0.5908221797	-0.7458302216
16	0.000535	0.5775700935	-0.7801510248
20	0.000551	0.5607985481	-0.8243910748
24	0.000693	0.4458874459	-1.296749683

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. In this case-

Execution time = 0.000216

Speed-up = 1.430555556

Parallel fraction = 0.4012944984

In this case, the loop iterations are being parallelized.

1b. Addition of N numbers using critical section

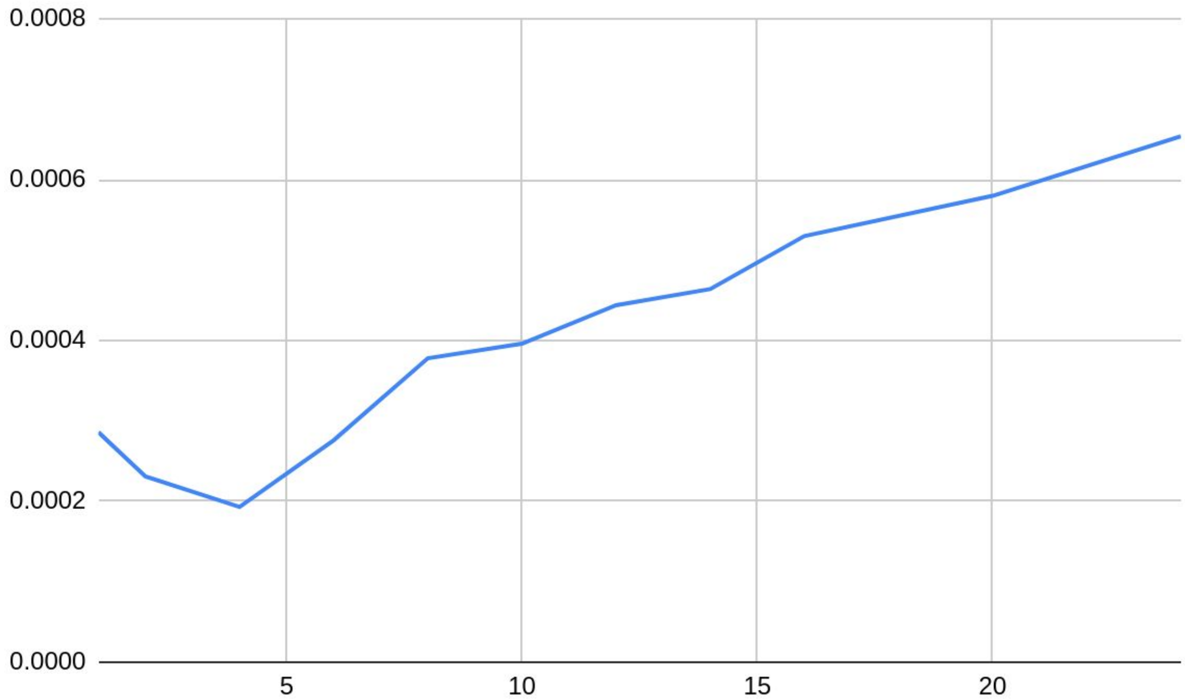
This program contains an array of size 1,00,000.

Therefore, here, N = 1,00,000.

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	0.000286	1	#DIV/0!
2	0.000231	1.238095238	0.3846153846
4	**0.000193**	**1.481865285**	**0.4335664336**
6	0.000276	1.036231884	0.04195804196
8	0.000378	0.7566137566	-0.3676323676
10	0.000396	0.7222222222	-0.4273504274
12	0.000444	0.6441441441	-0.6026700572
14	0.000464	0.6163793103	-0.6702528241
16	0.000530	0.5396226415	-0.91002331
20	0.000580	0.4931034483	-1.082075819
24	0.000654	0.4373088685	-1.342657343

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. In this case-

Execution time = 0.000193

Speed-up = 1.481865285

Parallel fraction = 0.4335664336

In this case, the loop iterations are being parallelized.

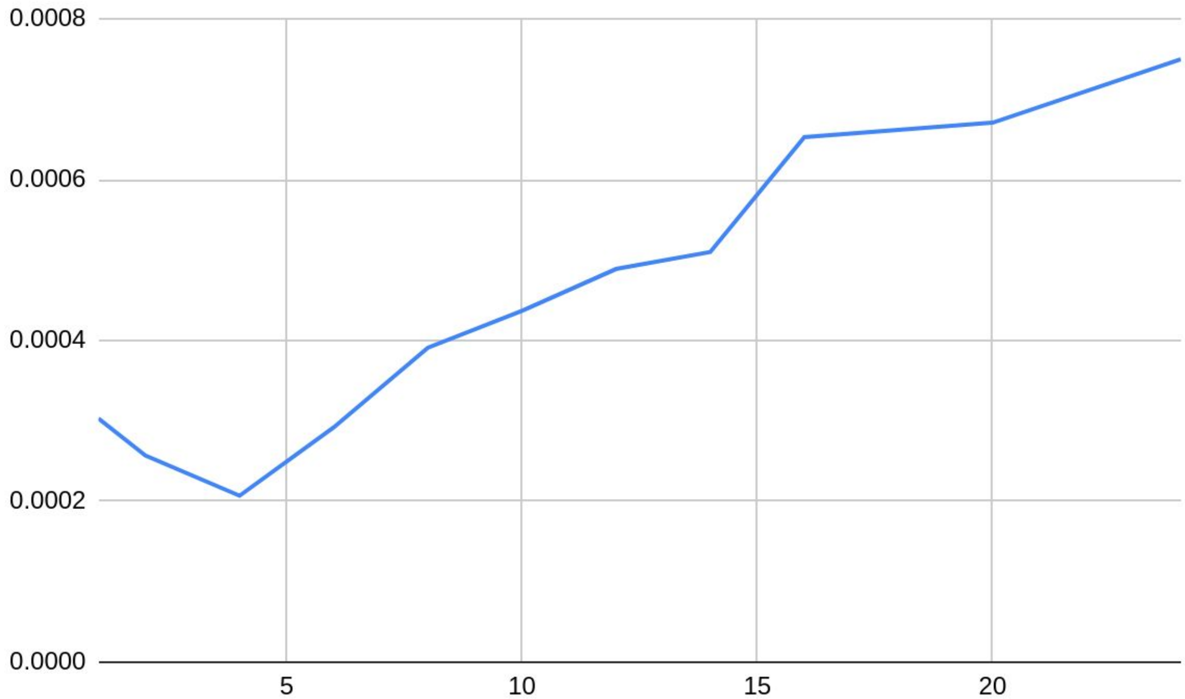
2a. Vector dot product using reduction

In this program, two vectors, each of size 1,00,000, are used for finding the dot product.

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	0.000303	1	#DIV/0!
2	0.000257	1.178988327	0.303630363
4	**0.000207**	**1.463768116**	**0.4224422442**
6	0.000292	1.037671233	0.04356435644
8	0.000391	0.7749360614	-0.3319189062
10	0.000437	0.6933638444	-0.4913824716
12	0.000489	0.6196319018	-0.6696669667
14	0.000510	0.5941176471	-0.7357197258
16	0.000653	0.4640122511	-1.232123212
20	0.000671	0.4515648286	-1.278443634
24	0.000750	0.404	-1.539388721

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. In this case-

Execution time = 0.000207

Speed-up = 1.463768116

Parallel fraction = 0.4224422442

In this case, the loop iterations are being parallelized.

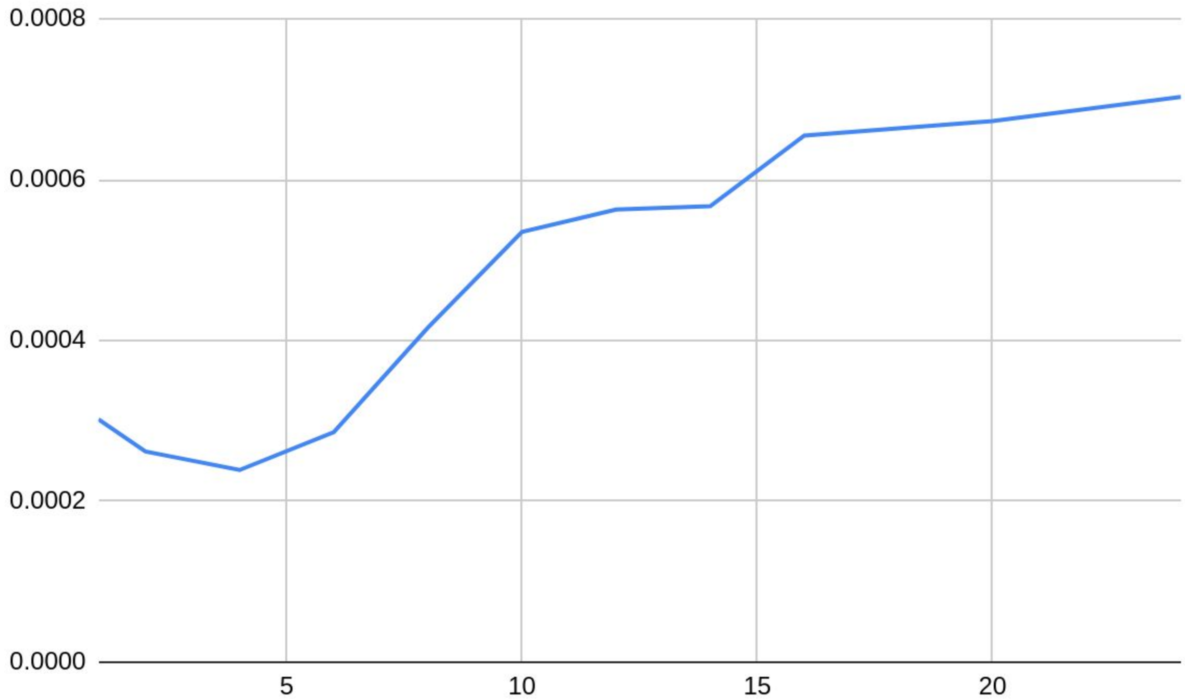
2b. Vector dot product using critical section

In this program, two vectors, each of size 100000, are used for finding the dot product.

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	0.000302	1	#DIV/0!
2	0.000262	1.152671756	0.2649006623
4	**0.000239**	**1.263598326**	**0.2781456954**
6	0.000286	1.055944056	0.06357615894
8	0.000416	0.7259615385	-0.43140965
10	0.000535	0.5644859813	-0.8572479765
12	0.000563	0.5364120782	-0.9428055388
14	0.000567	0.532627866	-0.9449821701
16	0.000655	0.4610687023	-1.246799117
20	0.000673	0.4487369985	-1.293133496
24	0.000703	0.4295874822	-1.385545638

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. In this case-

Execution time = 0.000239

Speed-up = 1.263598326

Parallel fraction = 0.2781456954

In this case, the loop iterations are being parallelized.