Sreepathy Jayanand

CED17I038

High Performance Computing Lab - 8

Analysis of parallelization factor

No of Grids used = 1

No of blocks used per grid = 1

No of threads per block = 1, 2, 4, 6, 8, 10, 12, 14, 16, 20, 24

Problem statement 1 : Given 2 matrices A & B find C such that find C = A + B

Input size of matrices: 1e4 x 1e4

Seg fault : 3 x 1e4 x 3 x 1e4

Let the number of threads = t

Strategy: The matrices A and B are first stored in row major fashion and converted to single dimensional arrays. So A and B have size of n \* n. Each thread computes the addition of

(n \* n) / t indices.

Thread 0 : Indices -> 0, t, 2t, ….

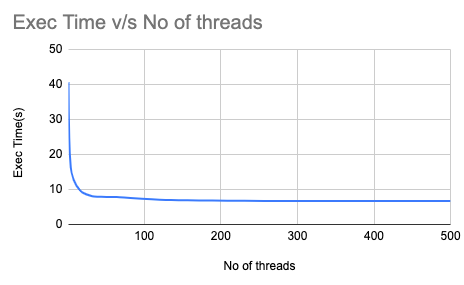
Thread 1 : Indices -> 1, t+1, 2t + 1, …..

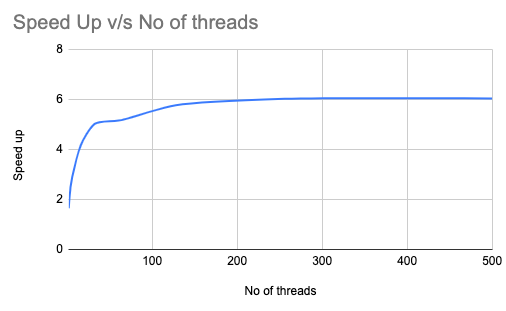
Thread t : Indices -> t-1, 2t - 1, 3t - 1, …

Each thread performs the addition of the indices it is assigned to and sends back the result.

| No of threads | Exec time (s) |
| --- | --- |
| 1 | 40.5 |
| 2 | 24.7 |
| 4 | 16.7 |
| 8 | 12.7 |
| 16 | 9.74 |
| 32 | 8.09 |
| 64 | 7.84 |
| 128 | 7.03 |
| 256 | 6.73 |
| 500 | 6.71 |

Plots:





The most efficient thread size for this program is 500

According to Amdahl’s law :

Speed up = 1/((1-p) + p/N)

Or

Parallelization factor = (N - N/S.U)/(N - 1)

So here N = 500;

Speed Up = 40.5/6.71 = 6.04

PF = (500 - 500/6.04)/499 = 0.83

**Problem Statement 2**: Given 2 matrices find their product

No of Grids used = 1

No of blocks used per grid = 1

No of threads per block = 1, 2, 4, 6, 8, 10, 12, 14, 16, 20, 24

The matrix size = 500 x 500 (n x n)

Segmentation fault = 3 \* 1e4 x 1 \* 1e4 (n x n)

Let the number of threads doing the job = t,

Strategy: The matrices are converted to 1 dimension. For every n \* n elements in C, we have to calculate the value. Again (n \* n)/t work is given to each thread.

Thread 0 : Indices -> 0, t, 2t, ….

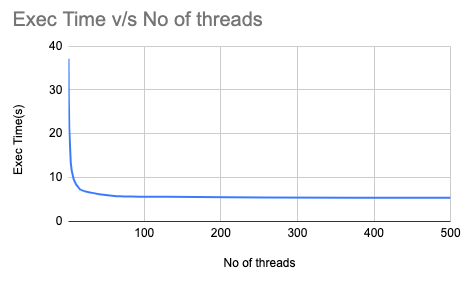
Thread 1 : Indices -> 1, t+1, 2t + 1, …..

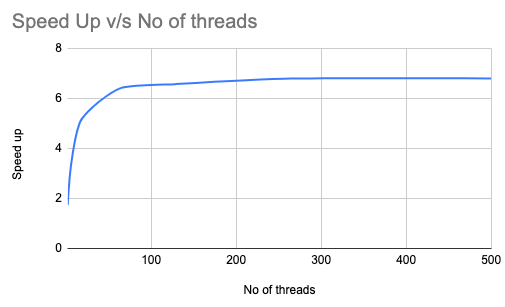
Thread t : Indices -> t-1, 2t - 1, 3t - 1, …

Each thread performs the multiplication of n numbers of A and n numbers of B, sums them to get the final value in a cell of C.

| No of threads | Exec Time(s) |
| --- | --- |
| 1 | 37.1 |
| 2 | 21.3 |
| 4 | 13.4 |
| 8 | 9.59 |
| 16 | 7.39 |
| 32 | 6.54 |
| 64 | 5.79 |
| 128 | 5.65 |
| 256 | 5.47 |
| 500 | 5.46 |

Plots:





The most efficient thread size for this program is 500

According to Amdahl’s law :

Speed up = 1/((1-p) + p/N)

Or

Parallelization factor = (N - N/S.U)/(N - 1)

So here N = 500;

Speed Up = 37.1/5.46 =6.79

PF = (500 - 500/6.79)/499 = 0.85

**Problem Statement 3**: Given 2 matrices find their product using **block multiplication** method

No of Grids used = 1

No of blocks used per grid = 1

No of threads per block = 1, 2, 4, 6, 8, 10, 12, 14, 16, 20, 24

The matrix size = 500 x 500 (n x n)

Segmentation fault = 3 \* 1e4 x 3 \* 1e4 (n x n)

Let the number of threads doing the job = t,

Let block size = bs

Strategy: The matrices are converted to 1 dimension. For each block in C assign a number.

To get the value of a block in C, n/bs row blocks of A are multiplied by n/bs column blocks of B.

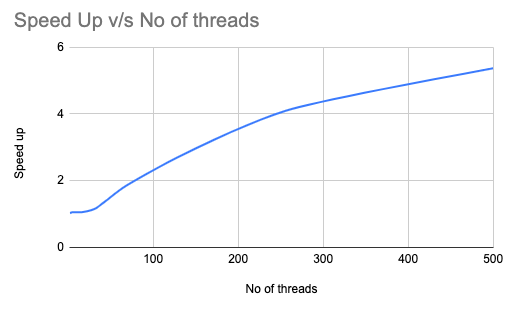
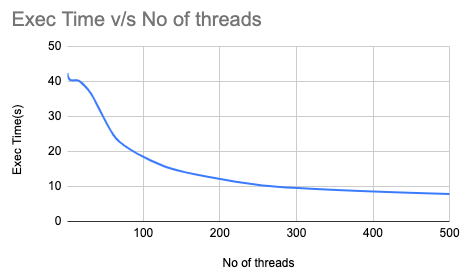
Thread 0 : Blocks -> 0, t, 2t, ….

Thread 1 : Blocks -> 1, t+1, 2t + 1, …..

Thread t : Blocks -> t-1, 2t - 1, 3t - 1, …

| No of threads | Exec time (s) |
| --- | --- |
| 1 | 42.2 |
| 2 | 41.1 |
| 4 | 40.4 |
| 8 | 40.2 |
| 16 | 40.1 |
| 32 | 36.4 |
| 64 | 23.8 |
| 128 | 15.7 |
| 256 | 10.3 |
| 500 | 7.86 |

Plots:



The most efficient thread size for this program is 500

According to Amdahl’s law :

Speed up = 1/((1-p) + p/N)

Or

Parallelization factor = (N - N/S.U)/(N - 1)

So here N = 500;

Speed Up = 42.2/7.68 = 5.55

PF = (500 - 500/5.55)/499 = 0.81