Sreepathy Jayanand

CED17I038

High Performance Computing Lab - 2

Analysis of parallelization factor

**Matrix Addition:**

Size of matrix A = 5000 x 5000, B = 5000 x 5000, C = 5000 x 5000

Data type - Long double

Strategy:

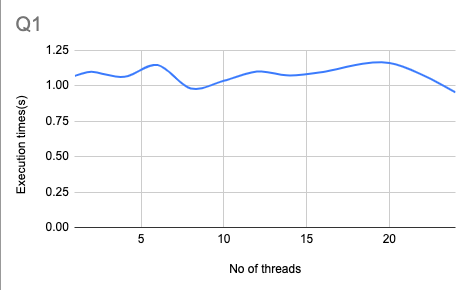
The number of threads used = n

All threads are launched and 1 thread among them assigns the tasks to the other threads.

1 task involves adding all the columns of a single row of A to all the columns of that same row of B. Then the n threads start computing the addition parallely.

Since each thread is computing a different row in the matrix there is no race condition among the threads.

| No of threads | Execution time(s) |
| --- | --- |
| 1 | 1.069 |
| 2 | 1.098 |
| 4 | 1.062 |
| 6 | 1.145 |
| 8 | 0.980 |
| 10 | 1.034 |
| 12 | 1.099 |
| 14 | 1.071 |
| 16 | 1.096 |
| 20 | 1.160 |
| 24 | 0.952 |



According to Amdahl’s law :

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

Or

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

So here N = 24;

Speed Up = 1.069/0.952 = 1.122

PF = (24 - 24/1.122)/23 = 0.11

**Matrix multiplication(Row major):**

Size of matrix A - 500 x 500, B - 500 x 500, C - 500 x 500

Data type - Long double

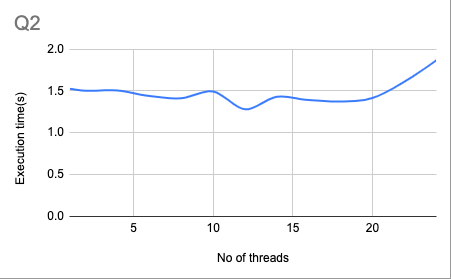
Strategy:

The number of threads used = n

N threads are launched and one among them assigns the tasks to the other threads.

Each task involves multiplying all the elements of any one row of matrix A among all the elements of any one column of matrix B and summing them up and calculating the result for a particular cell of matrix C. Again each thread is computing for a different cell of C.

| No of threads | Execution Time(s) |
| --- | --- |
| 1 | 1.525 |
| 2 | 1.501 |
| 4 | 1.504 |
| 6 | 1.438 |
| 8 | 1.412 |
| 10 | 1.494 |
| 12 | 1.279 |
| 14 | 1.431 |
| 16 | 1.390 |
| 20 | 1.414 |
| 24 | 1.864 |



According to Amdahl’s law :

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

Or

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

So here N = 12;

Speed Up = 1.525/1.279 = 1.19

PF = (12 -12/1.19)/11 = 0.17

**Matrix multiplication(Block based approach):**

Size of matrix A - 500 x 500, B - 500 x 500, C - 500 x 500

Data type - Long double

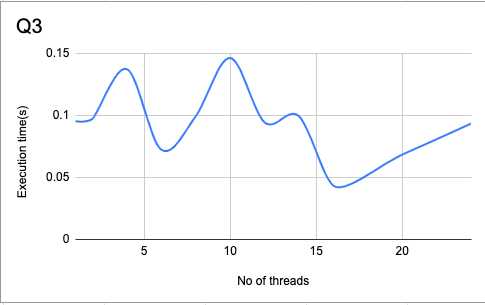
Strategy:

The number of threads used = n

N threads are launched and one among them distributes the task to the other threads.

Each task is multiplying 1 block of matrix A and 1 block of matrix B and adding that value of the block to the corresponding block of C. Here many threads have to update the same block of C because C[i][j] = ∑A[i][k] \* B[k][j] where C[i][j] is the (i, j)th block of C equals the sum of multiple block wise matrix products of A and B. Hence we have to ensure that when one thread updates a block of C no other thread updates the same block at the same time.

| No of threads | Execution time(s) |
| --- | --- |
| 1 | 0.095 |
| 2 | 0.097 |
| 4 | 0.137 |
| 6 | 0.072 |
| 8 | 0.099 |
| 10 | 0.146 |
| 12 | 0.094 |
| 14 | 0.099 |
| 16 | 0.043 |
| 20 | 0.068 |
| 24 | 0.093 |



According to Amdahl’s law :

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

Or

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

So here N = 16;

Speed Up = 0.095/0.043 = 2.375

PF = (16 - 16/2.375)/15 = 0.61