High Performance Computing

COM403P

Experiment-4

Matrix Multiplication

Kadali Paul Babu

CED19I002

CED19I002
Kadali Paul Babu

Objective

Matrix Multiplication for given n x n double precision floating point numbers.

Serial Code

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>

#define SIZE 100

int main()
{
    double a[SIZE][SIZE], b[SIZE][SIZE], c[SIZE][SIZE], rand_a, rand_b;
    double start, end, exec;

    start = omp_get_wtime();

    for (int i = 0; i < SIZE; i++)
    {
        for(int j = 0; j < SIZE; j++)
        {
            rand_a = rand();
            rand_b = rand();

            a[i][j] = i*rand_a;
            b[i][j] = i*rand_b;

            for(int k = 0; k < 100000; k++)
            c[i][j] = a[i][j] * b[i][j];
        }
    }

    end = omp_get_wtime();
    exec = end - start;

    printf("Serial Exec time - %f\n", exec);

    return 0;
}
```

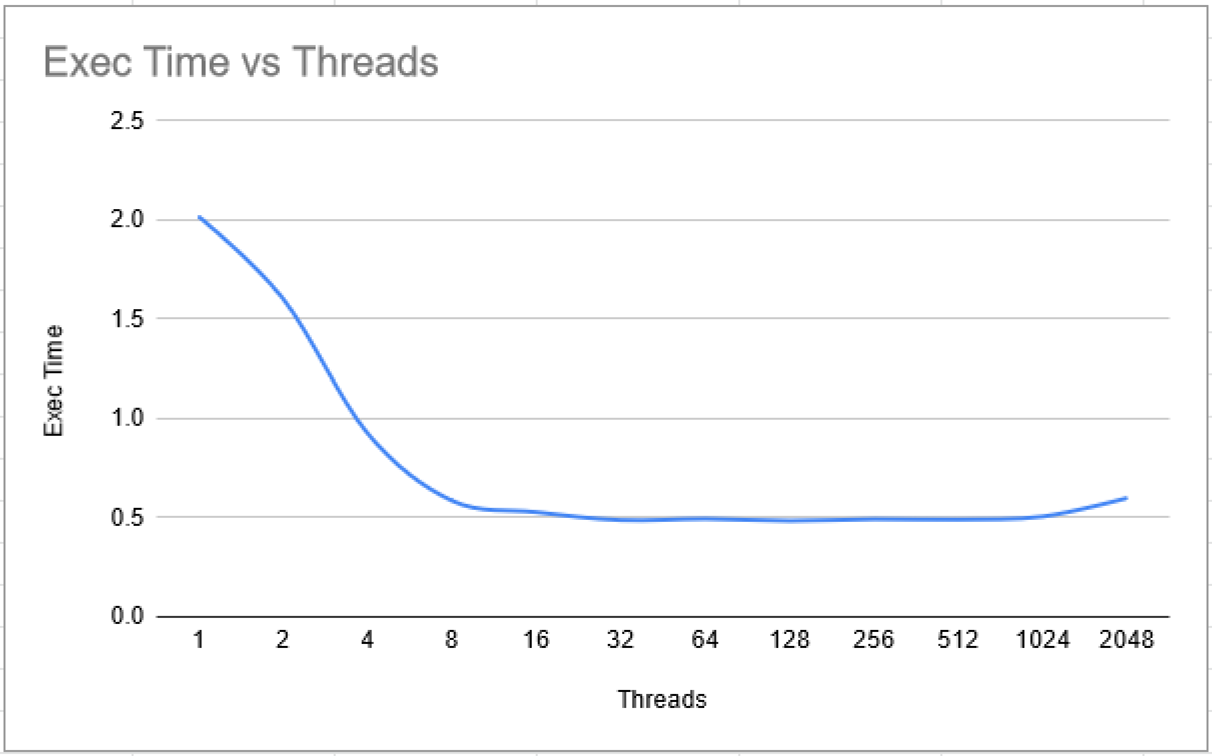
CED19I002
Kadali Paul Babu

Parallel Code

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>

#define SIZE 100

int main()
{
    double a[SIZE][SIZE], b[SIZE][SIZE], c[SIZE][SIZE], rand_a, rand_b;
    double start, end, exec;
    int i, j;

    start = omp_get_wtime();
    int thread;

    #pragma omp parallel private(j) shared(c)
    {
        thread = omp_get_num_threads();
        #pragma omp for
        for (i = 0; i < SIZE; i++)
        {
            for(j = 0; j < SIZE; j++)
            {
                rand_a = rand();
                rand_b = rand();

                a[i][j] = i*rand_a;
                b[i][j] = i*rand_b;

                for(int k = 0; k < 100000; k++)
                c[i][j] = a[i][j] * b[i][j];
            }
        }
    }

    end = omp_get_wtime();

    exec = end - start;

    printf("%d, %f\n", thread, exec);
```

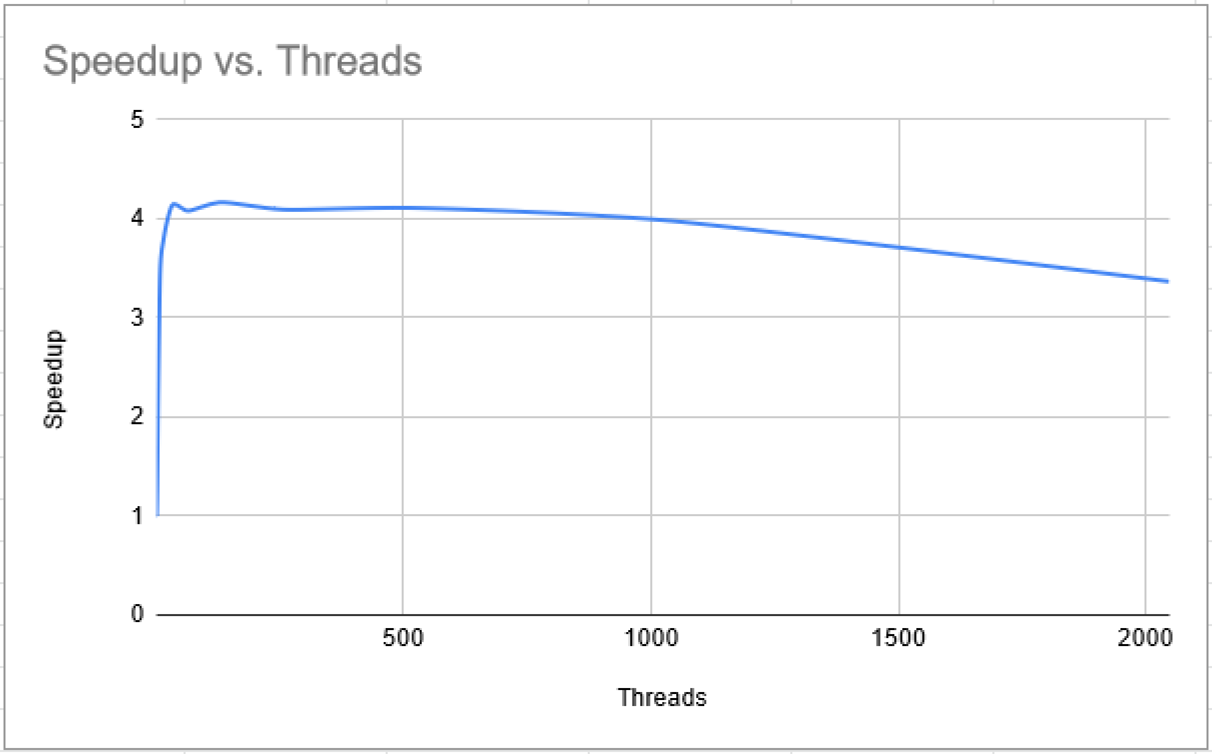
```
    return 0;
}
```

Threads vs Time



CED19I002
Kadali Paul Babu

Speedups vs Threads



Parallelization factor

Formula for calculating the Parallelization Factor = (1-(1/speedup))*(1-(1/p))
where, p = number of threads/processor

Threads	Exec Time	Speedup	Parallelization Factor
1	2.016622	1	0
2	1.602335	1.258552051	0.171196767
4	0.926133	2.177464792	0.360500216
8	0.586046	3.441064353	0.2364640804
16	0.527915	3.819974807	-0.2460727229
32	0.487774	4.134336804	-1.26353873
64	0.494327	4.079530351	-3.271119559
128	0.484223	4.164655541	-7.345546331
256	0.492791	4.092246003	-15.36458675
512	0.490749	4.109273784	-31.52699994
1024	0.506007	3.985363839	-63.17257192
2048	0.599315	3.364878236	-119.2438416

Inferences

From the above graphs and data we can clearly observe that the execution time decreases until a particular point and then increases as we add more threads to run the process.Using many threads leads to more context switches which lowers the speed up. We can conclude that the maximum parallelization happens at 64 threads