High Performance Computing

COM403P

Experiment-2

Vector Multiplication

Kadali Paul Babu

CED19I002

CED19I002
Kadali Paul Babu

Objective

Parallelize the Vector Multiplication for given N double precision floating point numbers.

Serial Code

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

#define SIZE 100000

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

    start = omp_get_wtime();

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

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

        c[i] = a[i] * b[i];
    }

    end = omp_get_wtime();

    exec = end - start;

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

    return 0;
}
```

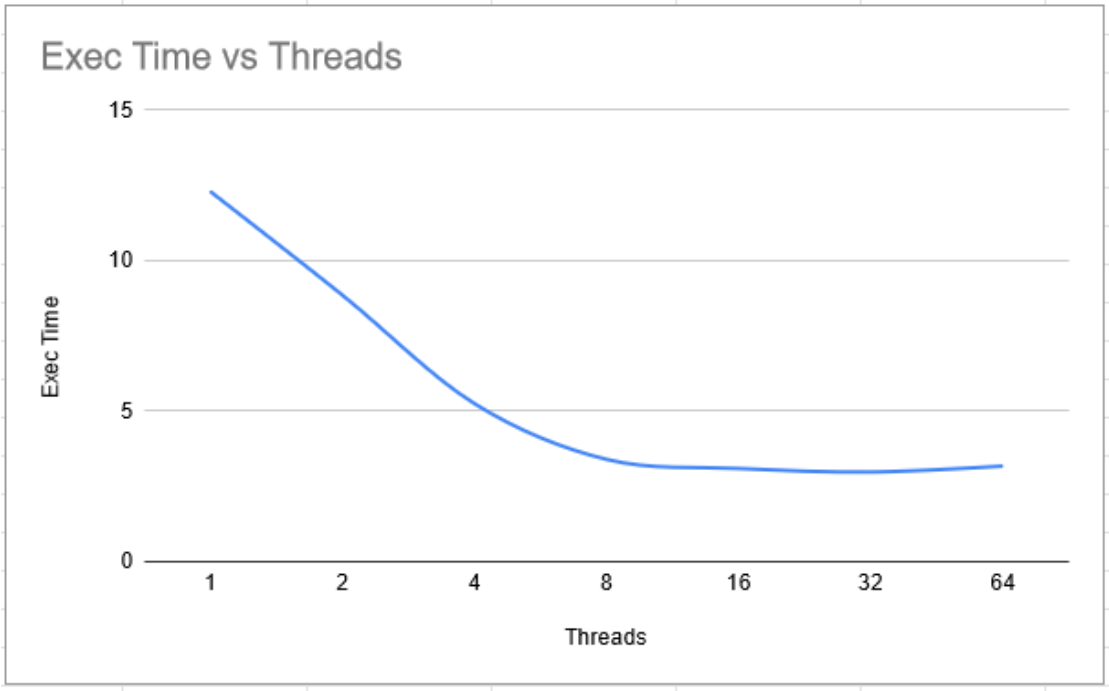
CED19I002
Kadali Paul Babu

Parallel Code

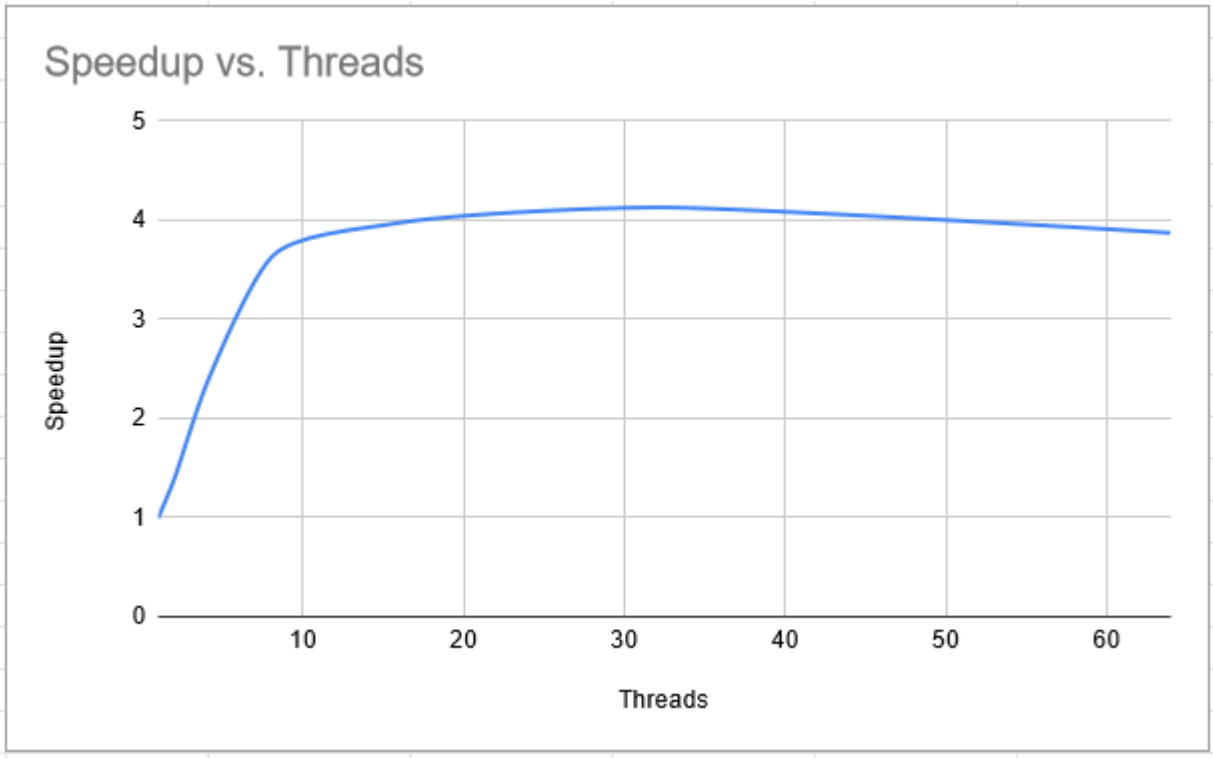
```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>

#define SIZE 100000

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

    start = omp_get_wtime();
    int thread;

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

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

            c[i] = a[i] * b[i];
        }
    }
    end = omp_get_wtime();

    exec = end - start;

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

    return 0;
}
```

CED19I002
Kadali Paul Babu

Threads vs Time



Speedups vs Threads



CED19I002
Kadali Paul Babu

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	12.29055	1	0
2	8.847745	1.389116662	0.2334317152
4	5.244967	2.34330359	0.3821680342
8	3.395862	3.619272515	0.2412337934
16	3.09364	3.972844287	-0.2494303889
32	2.97745	4.127877882	-1.262907952
64	3.174993	3.871047905	-3.213912071
128	3.201198	3.839359515	-7.148885607
256	3.264025	3.765458292	-14.9333709
512	3.335309	3.684980912	-30.35950724
1024	3.316519	3.705858462	-61.57657287
2048	3.286097	3.740166526	-124.3032676

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 32 threads