HPC Lab Week 1 Report 2

Name :	Prateek Agrawal
Roll Number :	CED18I040
Programming Environment :	OpenMP
Problem Statement :	Vector Multiplication
Date :	19th August 2021

Systems Specifications :

CPU Name :	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
Number of Sockets: :	1
Cores per Socket :	6
Threads per core :	2
L1 Cache size	192 Kb
L2 Cache size	1.5 MB
L3 Cache size(Shared):	9 MB
RAM	32 GB

Serial Code: (Run Time : 20.152976)

```
/*
* @Author: prateek
* @Date:   2021-08-19 15:41:50
* @Last Modified by:   prateek
* @Last Modified time: 2021-08-19 17:14:06
*/

#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
#define n 100000
#define m 100000

int main()
{
        double a[n],b[n], c[n], rand_a, rand_b;
        float startTime, endTime,execTime;
        int i;
        int omp_rank;
        srand(time(0));
   // Get the start time
        startTime = omp_get_wtime();
        {
                for(i=0;i<n;i++)
                {
                        rand_a = rand() , rand_b = rand();
                        omp_rank = omp_get_thread_num();
                        a[i] = i * rand_a;
                        b[i] = i * rand_b;

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

        }
        // Get the end time
        endTime = omp_get_wtime();

        // Get the total execution time :
        execTime = endTime - startTime;
```

```
		printf("%f \n",execTime);
		return(0);
	}
```

Parallel Code :

```
/*
* @Author: prateek
* @Date:   2021-08-19 15:41:50
* @Last Modified by:   prateek
* @Last Modified time: 2021-08-19 17:08:29
*/

#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
#define n 100000
#define m 100000

int main()
{
	double a[n],b[n], c[n], rand_a, rand_b;
	float startTime, endTime,execTime;
	int i;
	int omp_rank;
	srand(time(0));
   // Get the start time
	startTime = omp_get_wtime();
	#pragma omp parallel private (i) shared (a,b,c)
	{
		#pragma omp for
		for(i=0;i<n;i++)
		{
			rand_a = rand() , rand_b = rand();
			omp_rank = omp_get_thread_num();
			a[i] = i * rand_a;
			b[i] = i * rand_b;

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

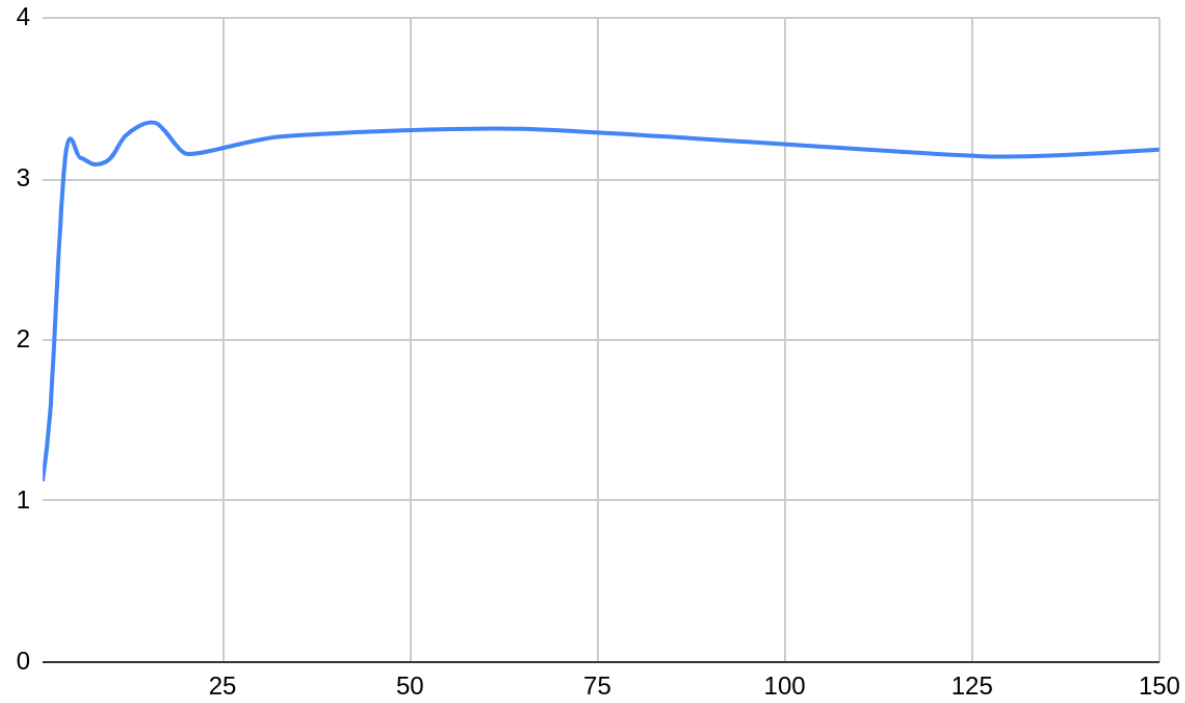
```
        }
        // Get the end time
        endTime = omp_get_wtime();

        // Get the total execution time :
        execTime = endTime - startTime;
        printf("%f \n",execTime);
        return(0);
}
```

Compilation and Execution:

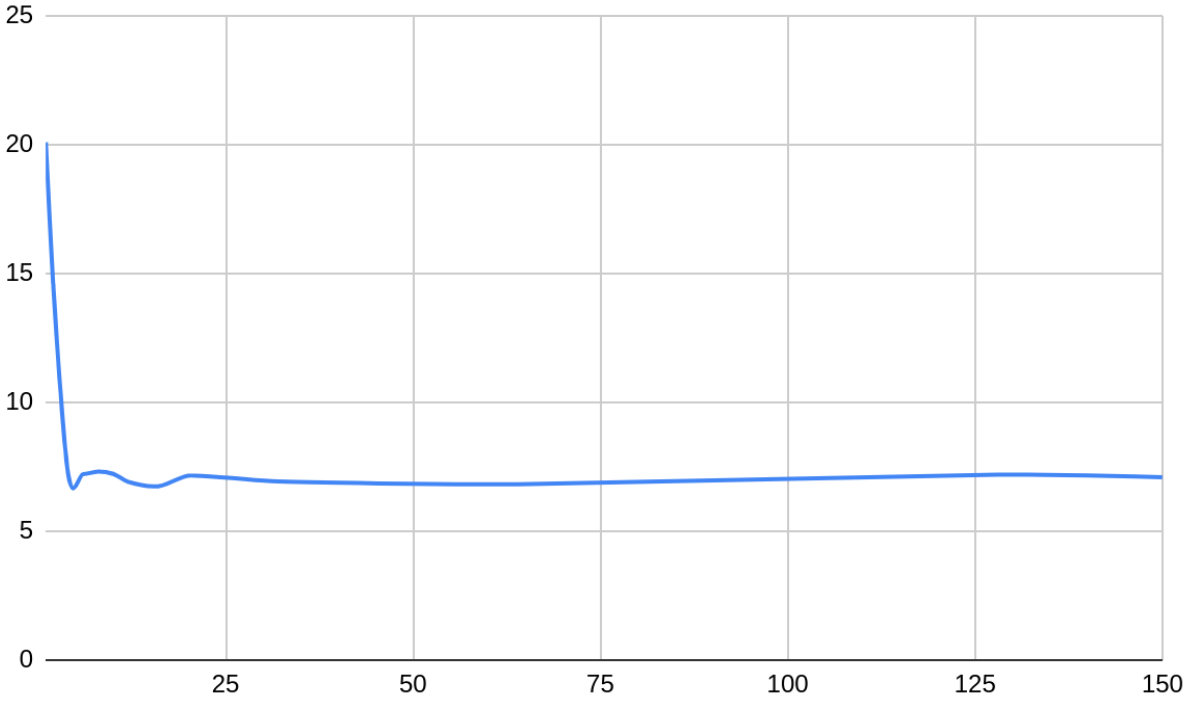
For enabling OpenMP environment use -fopenmp flag while compiling using gcc.

```
gcc -fopenmp vectoradd.c -o vectoradd
```

NUM Threads	Execution Time	Speed-Up	Parallelization Fraction
1	20.097656	1	0
2	14.6206	1.374612259	54.50442579
4	7.217773	2.78446773	85.44865796
6	7.214844	2.785598136	76.92128077
8	7.313477	2.748030246	72.69748415
10	7.226562	2.781081239	71.15862445
12	6.918945	2.904728394	71.53458909
16	6.746094	2.979154456	70.86232411
20	7.154297	2.809172725	67.79192768
32	6.933594	2.898591409	67.61340308
64	6.823242	2.945470203	67.09796896
128	7.199219	2.791643927	64.68415768
150	7.1	2.830655775	65.10654074



NUMBER OF THREADS vs SPEED-UP



NUMBER OF THREADS vs Execution Time

Inference: (Note: Execution time, graph and inference will be based on hardware configuration)

• At thread count 16 maximum speedup is observed..
• If thread count is more than 16 then the execution time increases/decreases slightly and tapers out.