HPC Lab Week 1 Report 1

Name :	Prateek Agrawal
Roll Number :	CED18I040
Programming Environment :	OpenMP
Problem Statement :	Vector Addition
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)

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#define n 100000
#define m 100000
int main()
{
double a[n],b[n], c[n];
int i,k;
int omp_rank;
clock_t startTime = clock();
/* here, do your time-consuming job */
for(i=0;i<n;i++)
{
a[i] = i * 10.236 ;
// Use Random function and assign a[i]
b[i] = i * 152.123;
// Use Random function and assign b[i]
for(int j=0;j<m;j++)
c[i] = a[i] + b[i];
}
clock_t endTime = clock();

double time_spent = (double)(startTime - endTime) / CLOCKS_PER_SEC;
printf("\n rtime=%f\n",time_spent);
return(0);
}
```

Parallel Code :

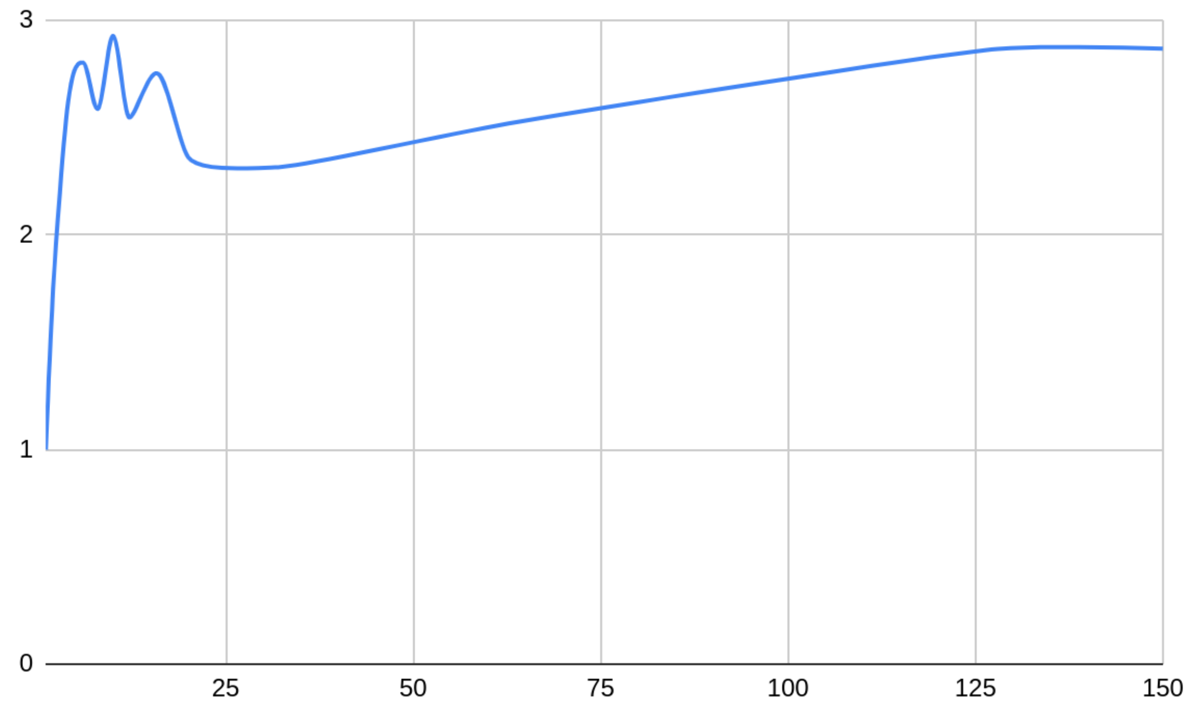
```
/*
* @Author: prateek
* @Date:   2021-08-19 15:41:50
* @Last Modified by:   prateek
* @Last Modified time: 2021-08-19 15:41:57
*/


#include <stdio.h>
#include<time.h>
#include <omp.h>
#include<stdlib.h>
#define n 100000
#define m 100000
int main()
{
double a[n],b[n], c[n];
float startTime, endTime,execTime;
int i,k;
int omp_rank;
float rtime[20];
int thread[]={1,2,4,6,8,10,12,16,20,32,64,128,150};
int thread_arr_size=13;
for(k=0;k<thread_arr_size;k++)
{
omp_set_num_threads(thread[k]);
startTime = omp_get_wtime();
#pragma omp parallel private (i) shared (a,b,c)
{
#pragma omp for
for(i=0;i<n;i++)
{
omp_rank = omp_get_thread_num();
a[i] = i * 10.236 ;
// Use Random function and assign a[i]
b[i] = i * 152.123;
// Use Random function and assign b[i]
for(int j=0;j<m;j++)
c[i] = a[i] + b[i];
}
}
endTime = omp_get_wtime();execTime = endTime - startTime;
rtime[k]=execTime;
}
for (k=0;k<thread_arr_size;k++)
printf("\nThread=%d\t rtime=%f\n",thread[k],rtime[k]);
return(0);
}
```

Compilation and Execution:

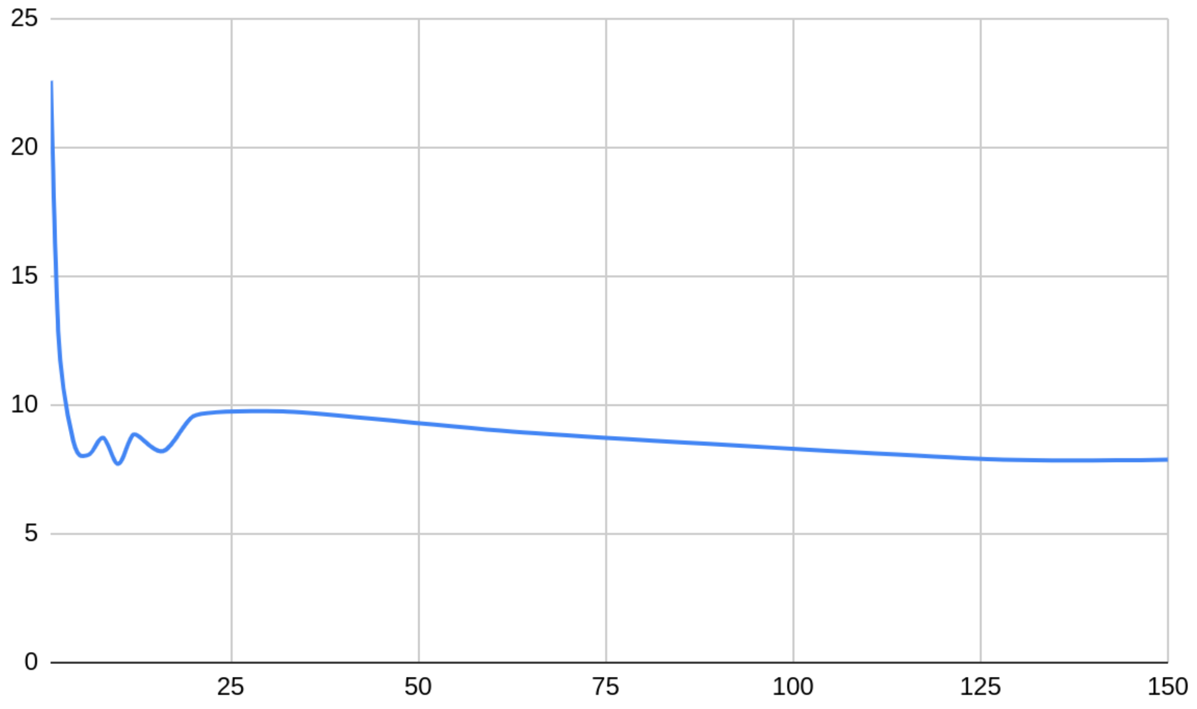
For enabling OpenMP environment use -fopenmp flag while compiling using gcc.

```
gcc -fopenmp 02_vector_addition.c
./a.out
```

NUM Threads	Execution Time	Speed-Up	Parallelization Fraction
1	22.599609	1	
2	12.898438	1.75211983	85.85255612
4	8.617188	2.622619931	82.4935865
6	8.061523	2.803391989	77.19471253
8	8.728516	2.589169682	70.14580524
10	7.717773	2.92825521	73.16663458
12	8.850586	2.553459059	66.36811363
16	8.212891	2.751723967	67.90308865
20	9.566406	2.362392836	60.7053027
32	9.757812	2.316052923	58.65609673
64	8.941406	2.527522965	61.39486693
128	7.881836	2.867302618	65.63680361
150	7.878906	2.868368908	65.57414186



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 10 maximum speedup is observed as the maximum number of parallel thread supported by the hardware is 8.
- If thread count is more than 10 then the execution time increases/decreases slightly and tapers out.