HPC Lab Week 2 Report 1

Name :	Prateek Agrawal
Roll Number :	CED18I040
Programming Environment :	OpenMP
Problem Statement :	Matrix 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 17:37:58
* @Last Modified by:   prateek
* @Last Modified time: 2021-08-19 17:43:00
*/

#include <iostream>
#include <omp.h>
#include <stdlib.h>
#include <time.h>

using namespace std;

int main()
{
	int a[200][200], b[200][200], c[200][200], m = 200, n = 200, p = 200, q = 200, i, j,
k;
	for (i = 0; i < m; ++i)
		for (j = 0; j < n; ++j)
			a[i][j] = 5;

	for (i = 0; i < p; ++i)
		for (j = 0; j < q; ++j)
			b[i][j] = 7;

	double start = omp_get_wtime ();
	{
		for (i = 0; i < m; ++i)
		{
			for (j = 0; j < q; ++j)
			{
				c[i][j] = 0;

				for(int count=0;count<10;count++)
				{
					for (k = 0; k < n; ++k)
c[i][j] = c[i][j] + (a[i][k] *
b[k][j]);}
			}
		}
	}
```

```
		double end = omp_get_wtime ();
		double time_elapsed = end - start;
		cout << time_elapsed << endl;

		return 0;
	}
```

Parallel Code :

```
/*
* @Author: prateek
* @Date:   2021-08-19 17:37:58
* @Last Modified by:   prateek
* @Last Modified time: 2021-08-19 17:43:00
*/

#include <iostream>
#include <omp.h>
#include <stdlib.h>
#include <time.h>

using namespace std;

int main()
{
	int a[200][200], b[200][200], c[200][200], m = 200, n = 200, p = 200, q = 200, i, j, k;
	for (i = 0; i < m; ++i)
		for (j = 0; j < n; ++j)
			a[i][j] = 5;

	for (i = 0; i < p; ++i)
		for (j = 0; j < q; ++j)
			b[i][j] = 7;

	double start = omp_get_wtime ();
	#pragma omp parallel
	{
		#pragma omp for
		for (i = 0; i < m; ++i)
		{
			for (j = 0; j < q; ++j)
			{
				c[i][j] = 0;
```

```
                                        for(int count=0;count<10;count++)
                                        {
                                                for (k = 0; k < n; ++k)
                                                                                        c[i][j] = c[i][j] + (a[i][k] *
b[k][j]);}
                                }
                        }
        }
        double end = omp_get_wtime ();
        double time_elapsed = end - start;
        cout << time_elapsed << endl;

        return 0;
}
```

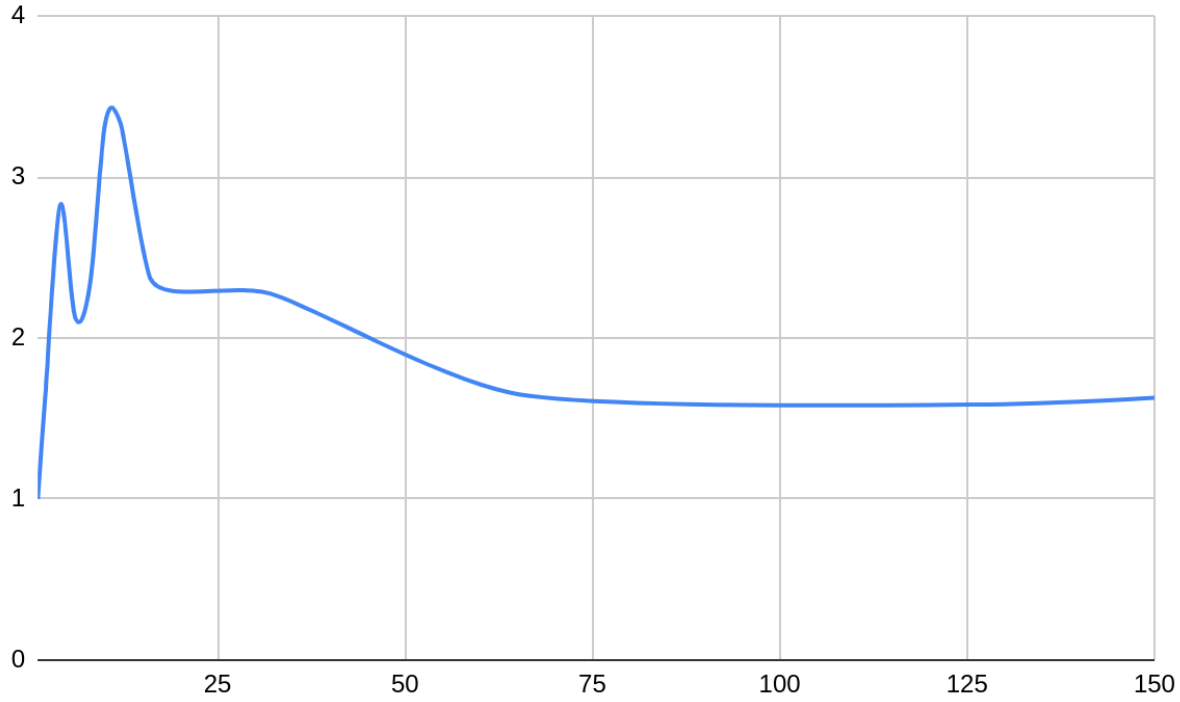
Compilation and Execution:

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

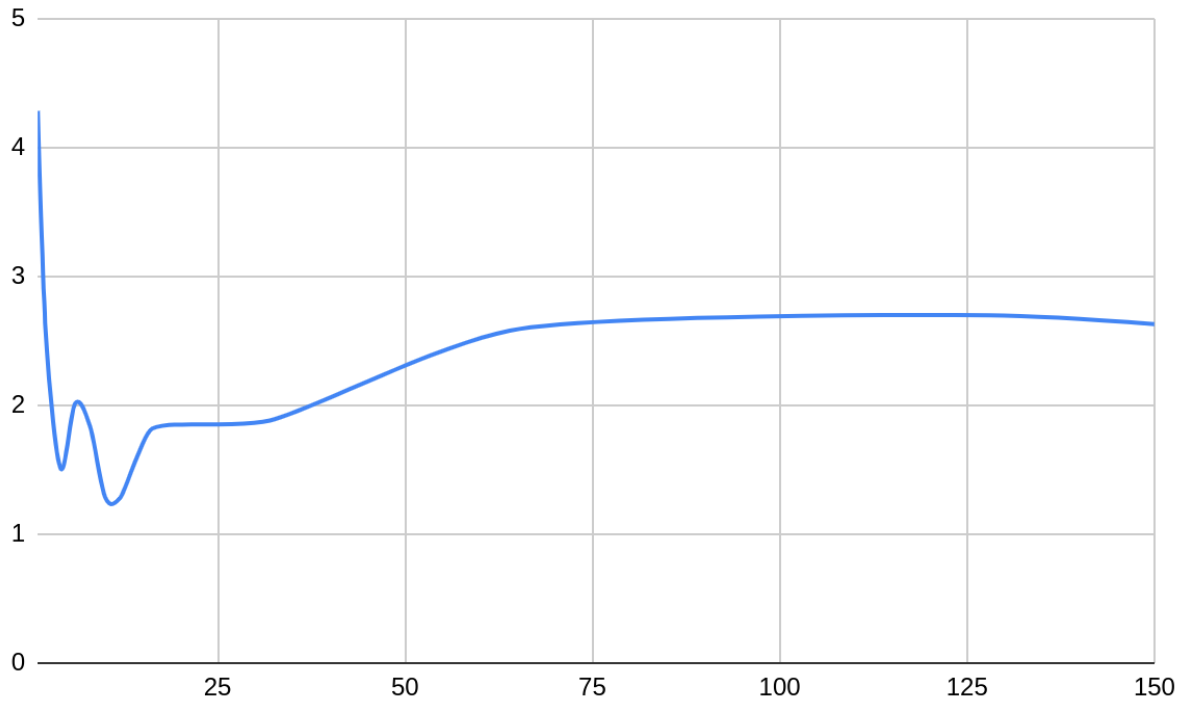
```
gcc -fopenmp matrix_multiplication.c -o matrix_multiplication
```

NUM Threads	Execution Time	Speed-Up	Parallelization Fraction
1	4.28682	1	0
2	2.6397	1.623979998	76.84577379
4	1.51687	2.826095842	86.15399449
6	2.0147	2.127770884	63.60295044
8	1.83298	2.338716189	65.41885527
10	1.28682	3.331328391	77.75771629
12	1.2829	3.341507522	76.44369571
16	1.80662	2.372839889	61.71350014
32	1.88319	2.276360856	57.8789511
64	2.58076	1.661068832	40.42950993

128	2.70002	1.58769935	37.30724612
150	2.63197	1.628749568	38.86228897



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 12 maximum speedup is observed.
• If thread count is more than 12 then the execution time increases/decreases slightly and tapers out.