HPC Lab Week 3 Report 1

Name :	Prateek Agrawal
Roll Number :	CED18I040
Programming Environment :	OpenMP
Problem Statement :	Sum of N Natural Numbers
Date :	26th August 2021

Systems Specifications :

CPU Name :	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
CPU Type :	Intel Coffeelake processor
CPU Stepping :	10
Number of Sockets: :	1
Cores per Socket :	6
Threads per core :	2
L1 Cache size	32 kB
L2 Cache size	256 kB
L3 Cache size:	9 MB
RAM	32 GB

Serial Code:

```
/*
* @Author: prateek
* @Date:   2021-08-26 15:34:26
* @Last Modified by:   prateek
* @Last Modified time: 2021-08-26 17:18:35
*/

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

int main(int argc, char const *argv[])
{
    double sum = 0.0;
    double start, end;

    start = omp_get_wtime (); //addition starts here
    for (int k = 0; k < 100000; k++)
    {
        sum = 0.0;
        for (int i = 0; i < 100000; i++)
        {
            sum += i;
        }
    }
    end = omp_get_wtime (); //addition starts here
    printf("%lf", end - start);

    return 0;
}
```

Parallel Code : (Reduction)

```
/*
* @Author: prateek
* @Date:   2021-08-26 17:19:25
* @Last Modified by:   prateek
* @Last Modified time: 2021-08-26 17:21:08
*/
#include <stdio.h>
#include <time.h>
#include <omp.h>

int main(int argc, char const *argv[])
{
    double sum = 0.0;
    double start, end;

    start = omp_get_wtime (); //addition starts here
    #pragma omp parallel for reduction (+:sum)

        for (int k = 0; k < 100000; k++)
        {
            sum = 0.0;
            for (int i = 0; i < 100000; i++)
            {
                sum += i;
            }
        }

    end = omp_get_wtime (); //addition starts here
    printf("%lf", end - start);

    return 0;
}
```

Parallel Code : (Reduction + Critical Section)

```
/*
* @Author: prateek
* @Date:   2021-08-26 17:23:12
* @Last Modified by:   prateek
* @Last Modified time: 2021-08-26 17:28:51
*/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
#define N 100000
int main (int argc, char *argv[])
{

    double start, end;

    int i;
    double sum = 0;
    double psum = 0;

    start = omp_get_wtime();

    #pragma omp parallel shared(sum) private(i,psum)
    {
        #pragma omp for
        for (int k = 0; k < 100000; k++)
        {
            sum=0.0;
            psum = 0.0;
            for ( i = 0; i < N; i++)
            {
                psum += i;
            }
            #pragma omp critical
            {
                sum += psum;
            }
        }
    }
    printf("%f",sum);
    end = omp_get_wtime();
    printf("%lf", end - start);

    return 0;

}
```

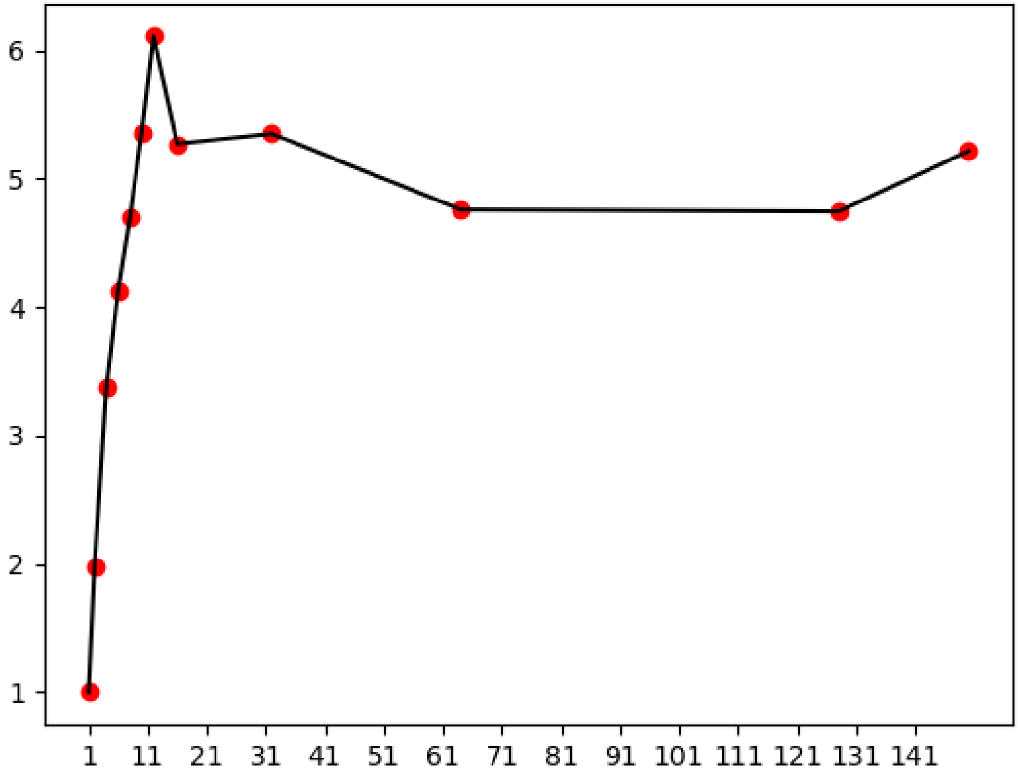
Compilation and Execution:

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

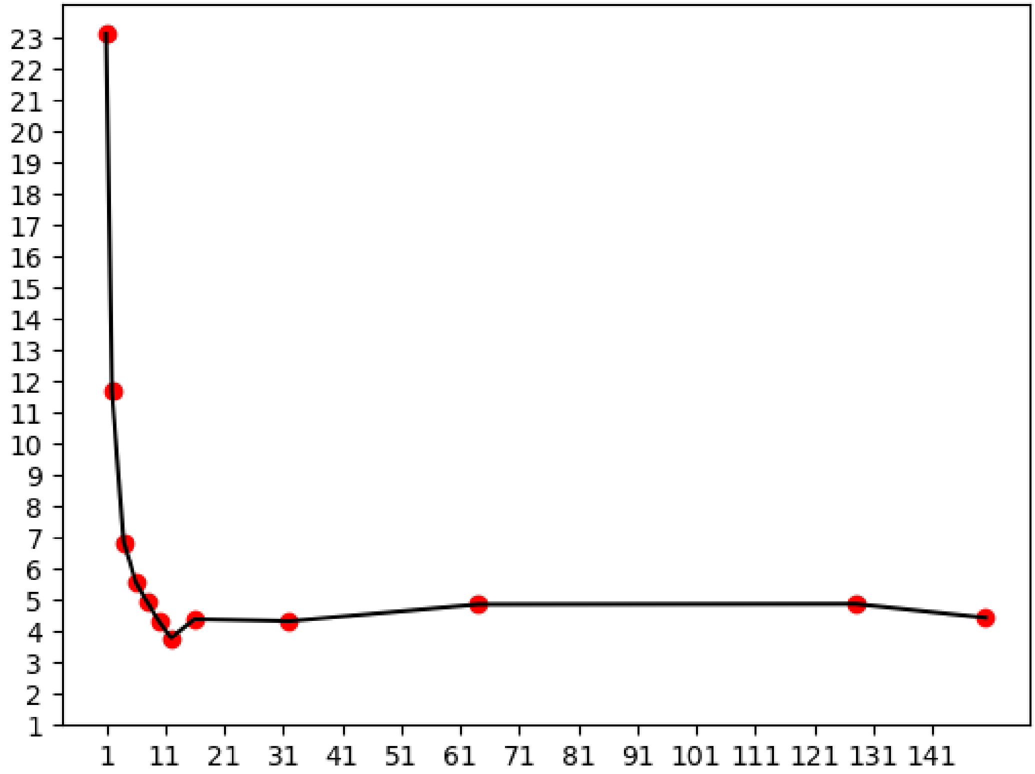
```
g++ -fopenmp sum_of_n_natural_numbers.cpp
```

Table 1 - Parallel Code with Reduction

NUM Threads	Execution Time	Speed-Up	Parallelization Fraction
1	23.121	1	
2	11.681	1.97936820477699	98.9576575407638
4	6.847	3.37680736088798	93.848305292447
6	5.602	4.12727597286683	90.9251329959777
8	4.921	4.69843527738265	89.9615068552398
10	4.31	5.36450116009281	90.3988197357861
12	3.783	6.11181601903251	91.241728298949
16	4.386	5.27154582763338	86.4322477401496
32	4.321	5.35084471187225	83.9343091254843
64	4.856	4.76132619439868	80.251375956579
128	4.87	4.7476386036961	79.5584475646266
150	4.433	5.21565531242951	81.3694166290037



NUMBER OF THREADS vs SPEED-UP (Parallel with Reduction)



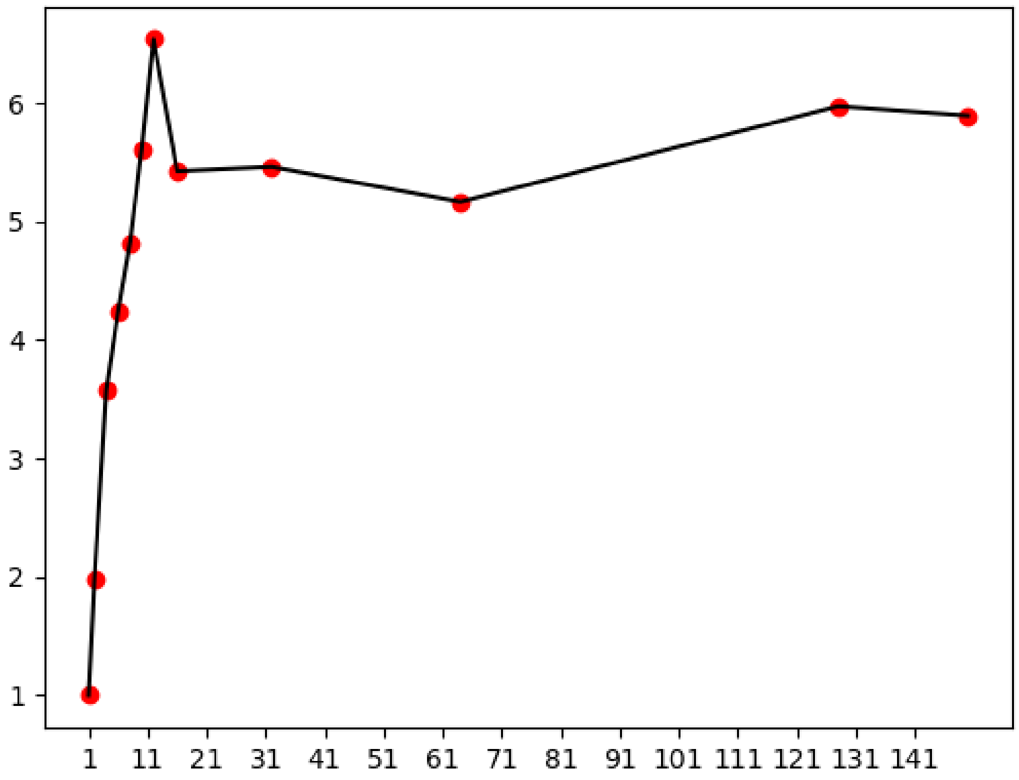
NUMBER OF THREADS vs Execution Time (Parallel with Reduction)

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

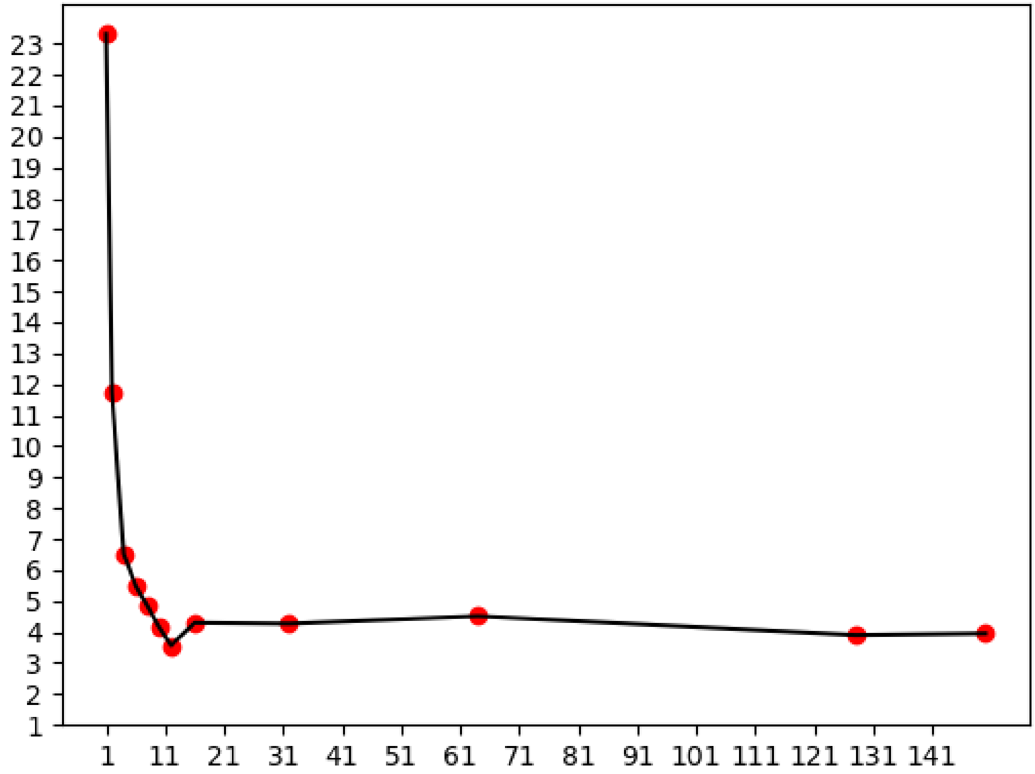
- At thread count 12 maximum speedup is observed.
- If thread count is more than 12 then the execution time increases/decreases slightly and tapers out.

Table 2 - Parallel Code with Reduction and Critical Section

NUM Threads	Execution Time	Speed-Up	Parallelization Fraction
1	23.31	1	
2	11.745	1.98467432950192	99.2277992277992
4	6.523	3.57350912156983	96.021736021736
6	5.507	4.2327946250227	91.6499356499356
8	4.84	4.81611570247934	90.5558619844334
10	4.159	5.60471267131522	91.286524619858
12	3.565	6.53856942496494	92.4066924066924
16	4.298	5.42345276872964	86.998998998999
32	4.269	5.46029515108925	84.3210030306804
64	4.512	5.16622340425532	81.9235562092705
128	3.904	5.97079918032787	83.9073494191605
150	3.956	5.89231547017189	83.585982914842



NUMBER OF THREADS vs SPEED-UP (Parallel with Reduction and Critical Section)



NUMBER OF THREADS vs SPEED-UP (Parallel with Reduction and Critical Section)

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.