HPC Experiment 5 Report

Sum of N Numbers

Name: Reuben Skariah Mathew
Roll No: CED18I042
Programming Environment: OpenMP
Problem: Sum of N Numbers
Date: 26th August 2021

Hardware Configuration:
Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
Sockets: 1
Cores per Socket: 4
Threads per Core: 2
L1 Cache: 32 kB
L2 Cache: 256 kB
L3 Cache: 6 MB
RAM: 8 GB

Serial Code:

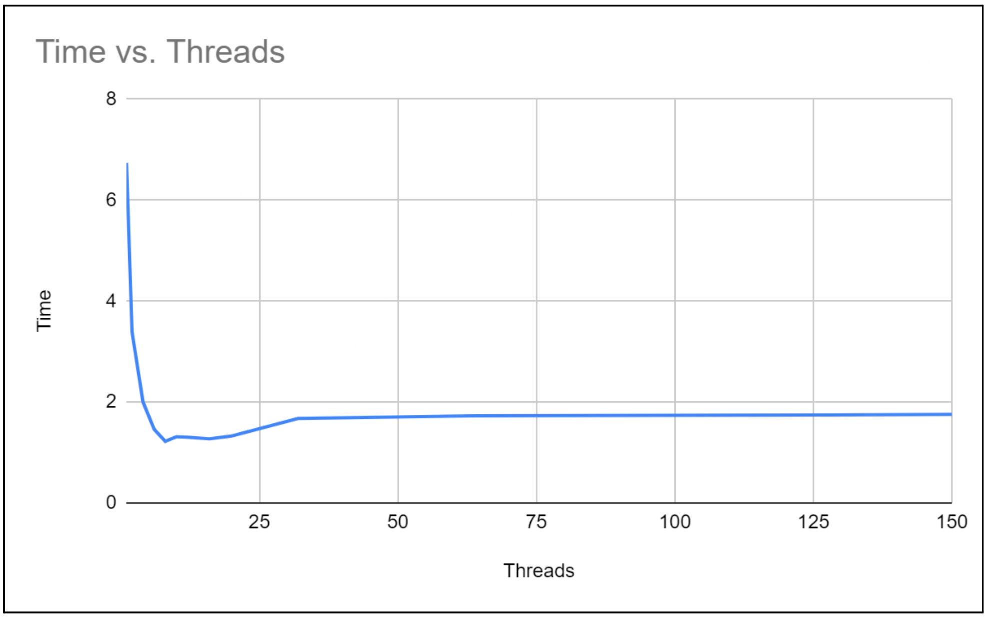
```
#include <stdio.h>
#include <omp.h>

#define n 50000
#define delay 50000

int main()
{
    double a[n],runtime;
    float startTime,endTime;
    int i;
    double sum;
    sum=0.0;
    startTime = omp_get_wtime();
    for(i=0;i<n;i++)
    {
        a[i] = (float) i * 5.52 ;
            for(int j=0;j<delay;j++)
                a[i] += 1.23;
        sum += a[i];
    }
    endTime = omp_get_wtime();
    runtime = endTime - startTime;
    printf("\n\nRun Time: %f", runtime);
    return 0;
}
```

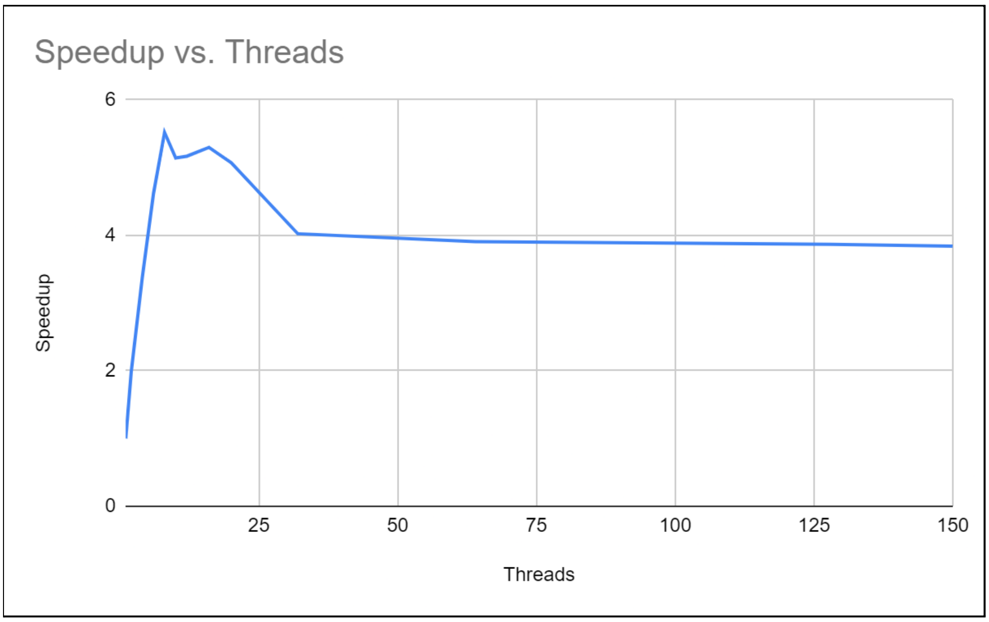
Parallel Code:

1. Reduction

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

#define n 50000
#define delay 50000

int main()
{
    double a[n],runtime[13];
    float startTime,endTime;
    int i,k,omp_rank;
    double sum;
    int threads[]={1, 2, 4, 6, 8, 10, 12, 16, 20, 32, 64, 128, 150};
    for(k=0;k<13;k++)
    {
        sum=0.0;
        omp_set_num_threads(threads[k]);
        startTime = omp_get_wtime();
        #pragma omp parallel private(i)
        {
            #pragma omp for reduction (+:sum)
            for(i=0;i<n;i++)
            {
                omp_rank = omp_get_thread_num();
                a[i] = (float) i * 5.52 ;
                    for(int j=0;j<delay;j++)
                        a[i] += 1.23;
                sum += a[i];
            }
        }
        endTime = omp_get_wtime();
        runtime[k] = endTime - startTime;
    }
    for(k=0;k<13;k++)
    printf("\n\nThread Count: %d      Run Time: %f",threads[k], runtime[k]);
    return 0;
}
```

Threads	Time	Speedup
1	6.741821	1
2	3.389893	1.988800531
4	1.996948	3.376062371
6	1.459961	4.617808969
8	**1.221313**	**5.520141847**
10	1.310669	5.143801372
12	1.304443	5.168352316
16	1.272217	5.2992697
20	1.328979	5.07293268
32	1.675537	4.023677782
64	1.72583	3.906422417
128	1.742554	3.868930891
150	1.755737	3.839880916

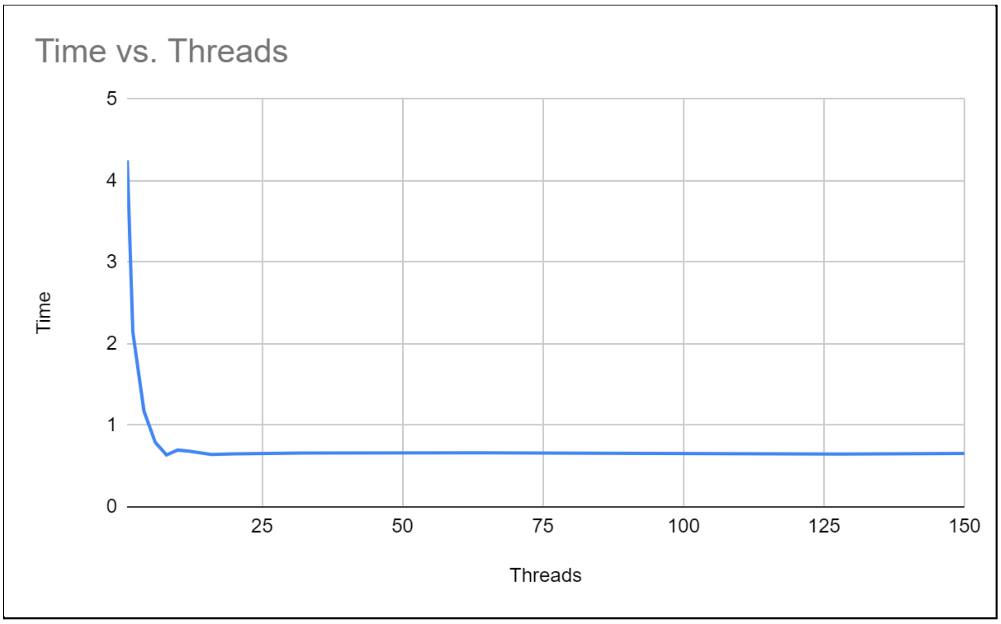


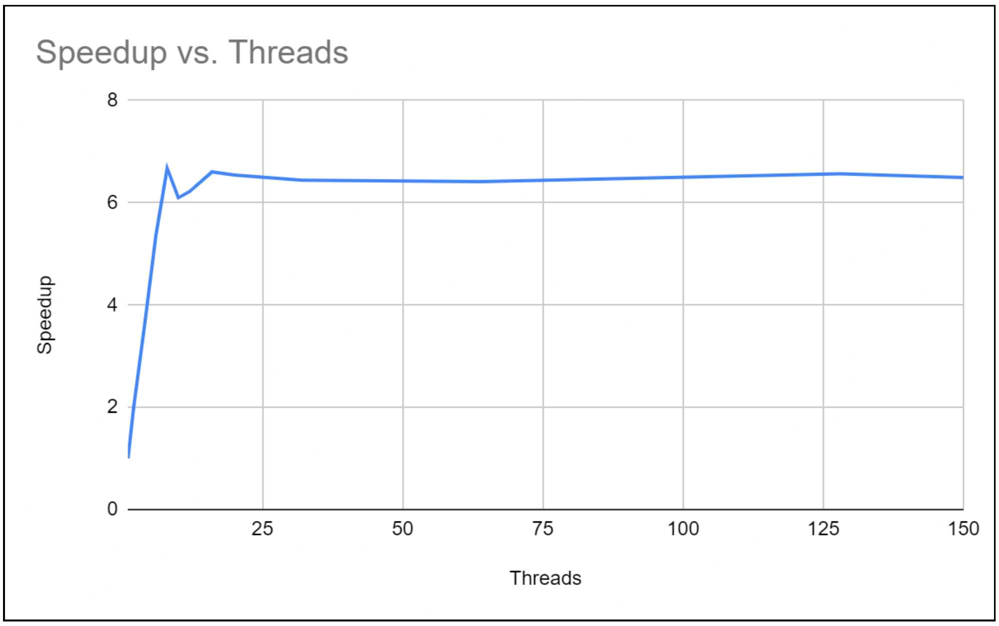


2. Critical Selection

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

#define n 50000
#define delay 50000

int main()
{
    double a[n],runtime[13];
    float startTime,endTime;
    int i,k,omp_rank;
    double sum, fsum;
    int threads[]={1, 2, 4, 6, 8, 10, 12, 16, 20, 32, 64, 128, 150};
    for(k=0;k<13;k++)
    {
        sum=0.0;
        omp_set_num_threads(threads[k]);
        startTime = omp_get_wtime();
        #pragma omp parallel private(i)
        {
            #pragma omp for
            for(i=0;i<n;i++)
            {
                omp_rank = omp_get_thread_num();
                a[i] = (float) i * 5.52 ;
                for(int j=0;j<delay;j++);
                sum += a[i];
            }
            #pragma omp critical(finalsum)
            fsum += sum;

        }
        endTime = omp_get_wtime();
        runtime[k] = endTime - startTime;
    }
    for(k=0;k<13;k++)
    printf("\n\nThread Count: %d      Run Time: %f",threads[k], runtime[k]);
    return 0;
}
```

Threads	Time	Speedup
1	4.25	1
2	2.143555	1.982687638
4	1.173828	3.620632665
6	0.790039	5.379481266
8	**0.636719**	**6.674844005**
10	0.696777	6.09951247
12	0.683105	6.221591117
16	0.643555	6.603942165
20	0.649414	6.544361532
32	0.659668	6.44263478
64	0.662598	6.41414553
128	0.646973	6.569053113
150	0.654297	6.495521147





Inference:

- Maximum speedup was observed at thread count equal to 8 in both reduction and critical selection method.

- The speedup increased from thread count 1 to 8 then tapered off as it increased further.

- Overall the critical selection method took less run time when compared to the reduction method for n and delay both equal to 50000.