**PARALLEL AND DISTRIBUTED COMPUTING**

**LAB 2**

S Shyam Sundaram

19BCE1560

18th August, 2021

Dr SK Ayesha

L43+L44

**AIM**

Compare the time taken when matrix addition is serialized and parallelized using Amdahl’s Law to calculate the speedup.

**CODE**

#include <stdio.h>

#include "omp.h"

#include<time.h>

#define ROWS 2500

#define COLS 250

int main()

{

int a[ROWS][COLS],b[ROWS][COLS],c[ROWS][COLS];

for(int i=0;i<ROWS;++i)

for(int j=0;j<COLS;++j)

{

a[i][j]=i\*10+j;

b[i][j]=j\*10+i;

}

float start=omp\_get\_wtime();

#pragma omp parallel for shared(a,b,c) //reduction(+: c)

for(int i=0;i<ROWS;++i)

for(int j=0;j<COLS;++j)

{

c[i][j]=a[i][j]+b[i][j];

}

float end=omp\_get\_wtime();

float exec=end-start;

printf("Time taken %f\n",exec);

return 0;

}

**EXECUTION**

**Bash commands:**

export OMP\_NUM\_THREADS=2

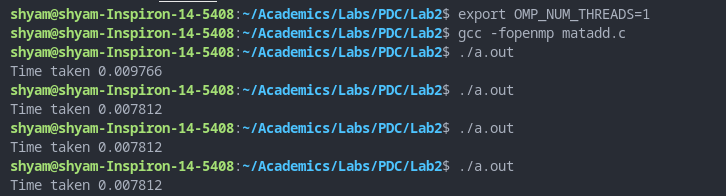
gcc -fopenmp prog.c

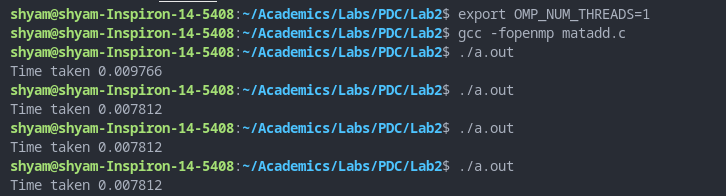
./a.out

*Note*: For Sequential, OMP\_NUM\_THREADS=1

**RESULTS**

We see that serial execution takes a lot longer than most of the parallelized implementations. The average execution time reduces as the number of threads increase from 2 to 6. But, from thread count 8, we see a general increase in the time taken to execute matrix addition. The observations are recorded in the table below.





**TABLE**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thread Count | Execution Time 1 | Execution Time 2 | | Execution Time 3 | Execution Time 4 | | Average Execution Time |
| 1 | 0.009766 | 0.007812 | | 0.011719 | 0.001953 | | 0.007812 |
| 2 | 0.003906 | 0.003906 | | 0.003906 | 0.004883 | | 0.004150 |
| 4 | 0.001953 | 0.003906 | | 0.001953 | 0.003906 | | 0.002930 |
| 6 | 0.001953 | 0.001953 | | 0.003906 | 0.001953 | | 0.002443 |
| 8 | 0.001953 | 0.001953 | | 0.001953 | 0.015625 | | 0.005371 |
| 16 | 0.003906 | 0.003906 | | 0.003906 | 0.003906 | | 0.003906 |
| 32 | 0.003906 | 0.005859 | | 0.005859 | 0.005859 | | 0.005371 |
| 64 | 0.005859 | 0.019531 | | 0.007812 | 0.007812 | | 0.010226 |
| Thread Count | | | | PF | | | Speed Up | |
| 2 | | | | 0.937532 | | | 1.88241 | |
| 4 | | | | 0.833248 | | | 2.66621 | |
| 6 | | | | 0.824731 | | | 3.19771 | |
| 8 | | | | 0.357106 | | | 1.45448 | |
| 16 | | | | 0.533333 | | | 1.99999 | |
| 32 | | | | 0.322548 | | | 1.45447 | |
| 64 | | | | -0.313917 | | | 0.76394 | |

We see that the optimal number of threads is 6, so p=6.

Tp=0.002443 and T1=0.007812. Thus, PF = = 0.824731

From Amdahl’s law, speedup = , where p = PF, s=1-p and n=optimal num of threads which here is 6. Which gives us a speedup = 3.197706.

**GRAPH**

**CONCLUSION**

We have thus calculated the speed up to be nearly 3.2 when parallelizing the problem of matrix addition for large matrices.