UCS645

PARALLEL & DISTRIBUTED COMPUTING

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

ASSIGNMENT 1:

Submitted by: Vansh Bhasin (102303270)

B.E Computer Engineering (3rd Year)

Submitted to: Dr. Saif Nalband

LAB 1

1.DAXPY LOOP

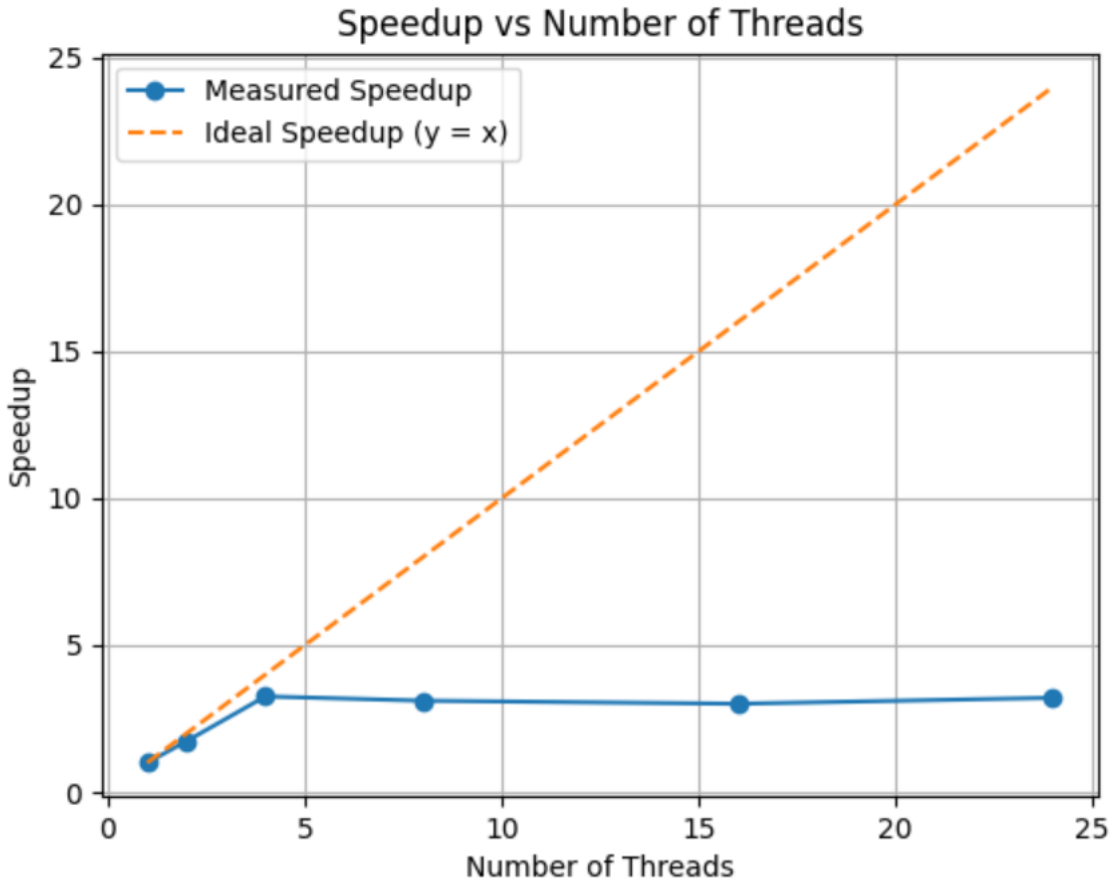
daxpy.c

```cpp
#include <iostream>
#include <omp.h>
#include <vector>

using namespace std;

#define N 65536 //fixed the size of the vectors

int main() {
omp_set_num_threads(16);
vector<double> X(N), Y(N); //double precision vectors
double a = 2.5;

for(int i = 0; i < N; i++) {
X[i] = i * 1.0;
Y[i] = i * 2.0;
}
double start = omp_get_wtime();
#pragma omp parallel for
for(int i = 0; i < N; i++) {
X[i] = a * X[i] + Y[i];
}

double end = omp_get_wtime();

cout << "Time taken: "
<< end - start << " seconds" << endl;

return 0;
}
```

```
root@LAPTOP-23841083:~# perf stat ./daxpy
Time taken: 0.0145406 seconds

 Performance counter stats for './daxpy':

            225.67 msec task-clock                #   11.534 CPUs utilized
                 3      context-switches          #   13.294 /sec
                 0      cpu-migrations            #    0.000 /sec
               433      page-faults               #    1.919 K/sec
   <not supported>      cycles
   <not supported>      instructions
   <not supported>      branches
   <not supported>      branch-misses

       0.019565667 seconds time elapsed

       0.220187000 seconds user
       0.003931000 seconds sys
```

INFERENCE:

The program finished very fast in 0.01956 seconds which depicts it is very lightweight and simple. The task-clock is 225.67 ms and it shows 11.534 CPUs utilized. This means the program used multiple CPU cores at the same time, which indicates good parallel execution. There were only 3 context switches. This means the operating system did not frequently switch between tasks, so the program ran smoothly without much interruption. The value is 0, which means the program stayed on the same CPU cores and was not moved between cores. This helps in better performance and cache usage. There were 433 page faults. This means the program accessed memory pages that were not already in main memory, but the number is small, so memory performance is still acceptable. The user time (0.220 s) is much higher than the system time (0.0039 s). This shows that most of the time was spent doing actual computation rather than operating system work. Values for cycles, instructions, branches, and branch-misses are shown as not supported because I was using WSL and these required full hardware access which was possible by dual boot or ubuntu-native.



Threads N	Execution Time (s)	Speedup (S)	Efficiency (E=S/N)
1 Sequential	0.004998	1.00 x	100.0 %
2	0.002881	1.73 x	86.7 %
4	0.001534	3.26 x	81.4 %
8	0.001606	3.11 x	38.9 %
16	0.001662	3.01 x	18.8 %
24	0.001557	3.21 x	13.4 %

2.MATRIX MULTIPLY:

matrix.cpp

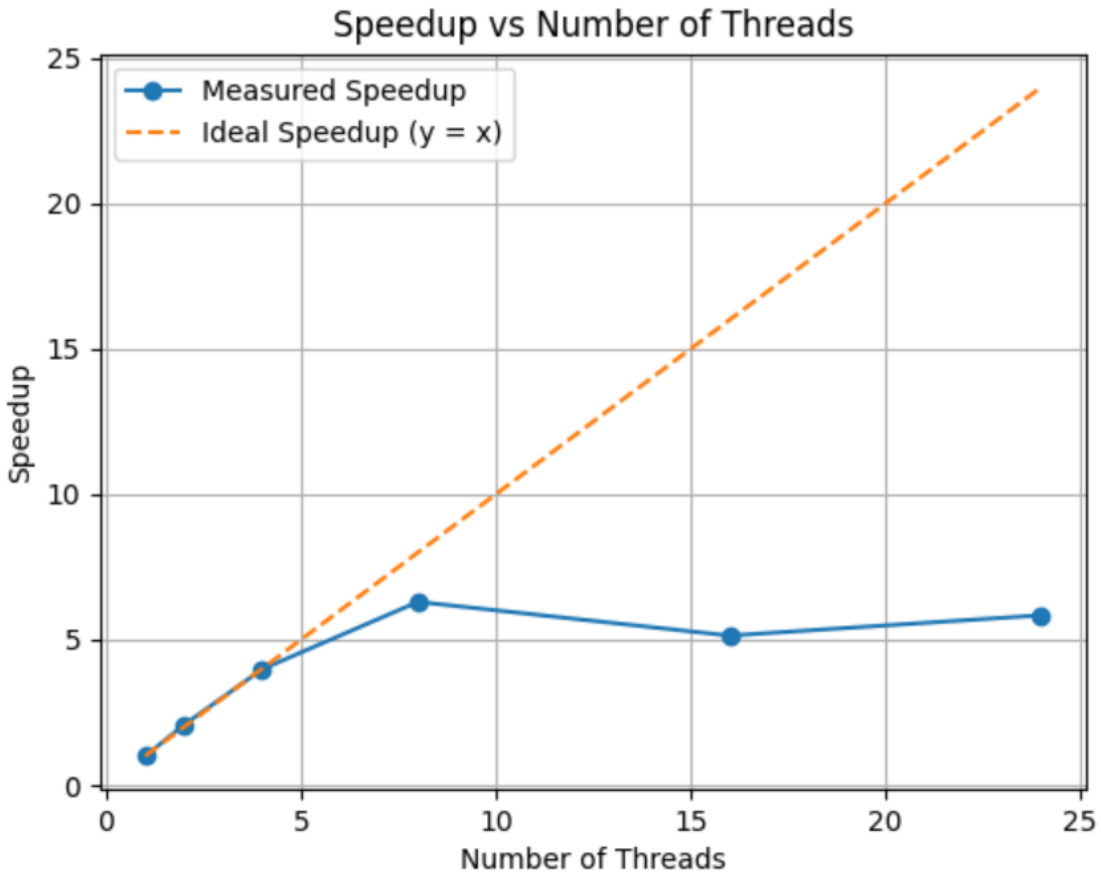
```cpp
#include <iostream>

#include <omp.h>

using namespace std;

#define N 1000

int main() {
    static int A[N][N], B[N][N], C[N][N];
    double start, end;


    for(int i = 0; i < N; i++) {
        for(int j = 0; j < N; j++) {
            A[i][j] = 1;
            B[i][j] = 1;
            }
            }

start = omp_get_wtime();


#pragma omp parallel for collapse(2)
    for(int i = 0; i < N; i++) {
        for(int j = 0; j < N; j++) {
            C[i][j] = 0;
                for(int k = 0; k < N; k++) {
                    C[i][j] += A[i][k] * B[k][j];
                }
            }
        }

end = omp_get_wtime();
cout << "2D Threading Time: "
<< end - start << " seconds" << endl;
return 0;
}
```

```
root@LAPTOP-23841083:~# ./matrix
2D Threading Time: 0.0761368 seconds
root@LAPTOP-23841083:~# perf stat ./matrix
2D Threading Time: 0.0854187 seconds

 Performance counter stats for './matrix':

          1566.76 msec task-clock                #   15.733 CPUs utilized
               34      context-switches          #   21.701 /sec
                5      cpu-migrations            #    3.191 /sec
             3120      page-faults               #    1.991 K/sec
   <not supported>      cycles
   <not supported>      instructions
   <not supported>      branches
   <not supported>      branch-misses

       0.099587309 seconds time elapsed

       1.679493000 seconds user
       0.054470000 seconds sys


root@LAPTOP-23841083:~#
```

INFERENCE:

The program took about 0.099 seconds to complete. This is still fast, but slower than smaller programs because matrix operations involve more calculations. The task-clock is 1566.76 ms and it shows 15.733 CPUs utilized. This means the program used many CPU cores at the same time, which indicates good parallel performance. There were 34 context switches. This means the operating system switched between tasks a few times, but not too often, so the program still ran smoothly. The value is 5, which means the program was moved between CPU cores a few times. This can slightly affect performance, but the number is small, so the impact is low. There were 3120 page faults. This means the program accessed memory pages that were not already in main memory. Since matrix programs use more memory, this number is expected and acceptable. The user time (1.679 s) is much higher than the system time (0.054 s). This shows that most of the time was spent on actual computation rather than

operating system tasks. The values for cycles, instructions, branches, and branch-misses are shown as not supported.



Threads N	Execution Time (s)	Speedup (S)	Efficiency (E=S/N)
1 Sequential	0.449530	1.00 x	100.0 %
2	0.216443	2.08 x	103.8 %
4	0.112889	3.98 x	99.6 %
8	0.071335	6.30 x	78.8 %
16	0.087433	5.14 x	32.1 %
24	0.076970	5.84 x	24.3 %

3.CALCULATION OF PI:

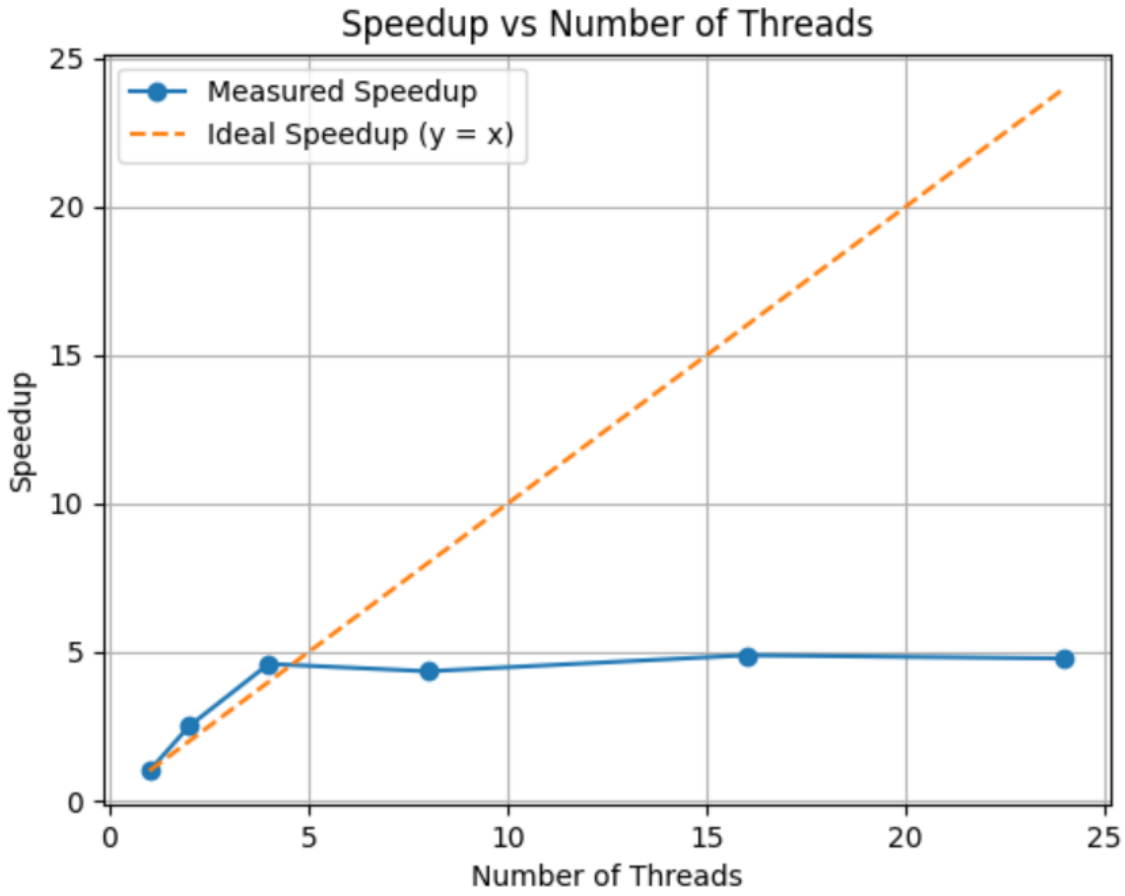
pi.cpp

```cpp
#include<iostream>
#include<omp.h>

using namespace std;

static long num_steps=100000000;

int main(){
    double step=1.0/num_steps;
    double sum=0.0;
    double pi;

    #pragma omp parallel for reduction(+:sum)
    for (long i=0;i<num_steps;i++){
        double x=(i+0.5)*step;
        sum+=4.0/(1.0+x*x);
    }
    pi=step*sum;

    cout<<"Approximate Value of Pi="<<pi<<endl;
    return 0;
}
```

```
root@LAPTOP-23841083:~# g++ -O3 -fopenmp -pg pi.cpp -o pi
root@LAPTOP-23841083:~# ./pi
Approximate Value of Pi=3.14159
root@LAPTOP-23841083:~# perf stat ./pi
Approximate Value of Pi=3.14159

 Performance counter stats for './pi':

            553.41 msec task-clock                #   15.948 CPUs utilized
                19      context-switches          #   34.332 /sec
                 9      cpu-migrations            #   16.263 /sec
               189      page-faults               #  341.516 /sec
   <not supported>      cycles
   <not supported>      instructions
   <not supported>      branches
   <not supported>      branch-misses

       0.034701571 seconds time elapsed

       0.536483000 seconds user
       0.008317000 seconds sys
```

INFERENCE:

The program completed in about 0.034 seconds, which shows that the calculation of Pi was done very quickly. The task-clock is 553.41 ms and it shows 15.948 CPUs utilized. This means the program used many CPU cores at the same time, which indicates good parallel performance using OpenMP There were 19 context switches. This means the operating system switched between tasks a few times, but not very frequently, so the program ran smoothly. The value is 9, which means the program moved between CPU cores a few times. This can slightly affect performance, but the number is small, so the impact is low. There were 189 page faults. This is a small number, which means the program did not heavily use memory and had efficient memory access. The user time (0.536 s) is much higher than the system time (0.008 s). This shows that most of the time was spent on actual computation rather than operating system work. The values for cycles, instructions, branches, and branch-misses are shown as not supported as I am using WSL and these require full hardware access.



Threads N	Execution Time (s)	Speedup (S)	Efficiency (E=S/N)
1 Sequential	0.109998	1.00 x	100.0 %
2	0.043895	2.51 x	125.3 %
4	0.023901	4.60 x	115.1 %
8	0.025261	4.35 x	54.4 %
16	0.022474	4.89 x	30.6 %
24	0.022975	4.79 x	19.9 %