PARALLEL AND DISTRIBUTIVE COMPUTING (UCS645)

LAB – 1

By

Aditya Chauhan

102496001

3P14

Instructor: Dr. Saif Nalband

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, (A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB INDIA

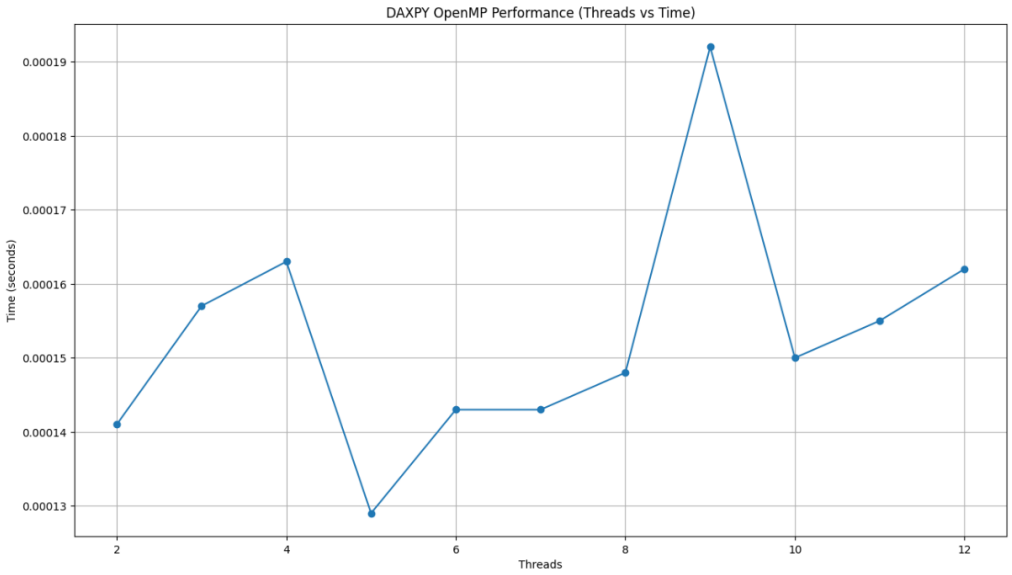
Session-Year (Jan-May, 2026)

Q1. DAXPY Loop Analysis:

```
aditya_chauhan@Dodo: $ ./daxpy
Time taken: 0.000165 seconds
```

```
aditya_chauhan@Dodo:~$ nano daxpy.c
aditya_chauhan@Dodo:~$ gcc daxpy.c -fopenmp -o daxpy
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=2 ./daxpy
Time taken: 0.000141 seconds
aditya_chauhan@Dodo:~$
```

```
aditya_chauhan@Dodo:~$ daxpy.c
daxpy.c: command not found
aditya_chauhan@Dodo:~$ nano daxpy.c
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=3 ./daxpy
Time taken: 0.000157 seconds
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=4 ./daxpy
Time taken: 0.000163 seconds
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=5 ./daxpy
Time taken: 0.000129 seconds
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=6 ./daxpy
Time taken: 0.000143 seconds
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=7 ./daxpy
Time taken: 0.000143 seconds
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=8 ./daxpy
Time taken: 0.000148 seconds
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=9 ./daxpy
Time taken: 0.000192 seconds
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=10 ./daxpy
Time taken: 0.000150 seconds
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=11 ./daxpy
Time taken: 0.000155 seconds
aditya_chauhan@Dodo:~$ OMP_NUM_THREADS=12 ./daxpy
Time taken: 0.000162 seconds
aditya_chauhan@Dodo:~$
```



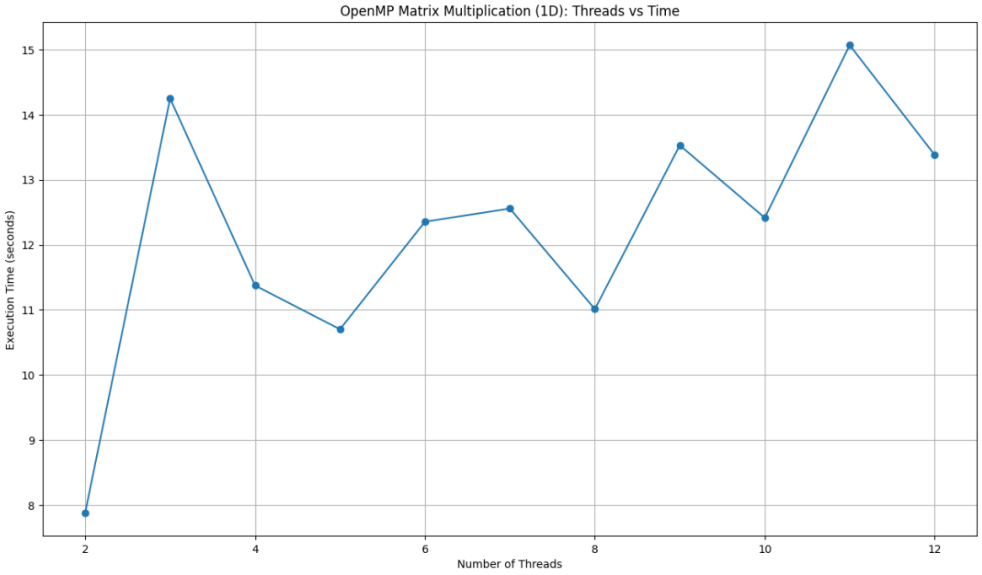
Observation :

From the execution time graph, it is observed that increasing the number of threads does not significantly improve performance. In fact, after a small number of threads, the execution time increases. This happens because DAXPY is a memory-bound operation where threads compete for memory bandwidth and cache resources. Therefore, DAXPY shows poor scalability with OpenMP and performs best with fewer threads.

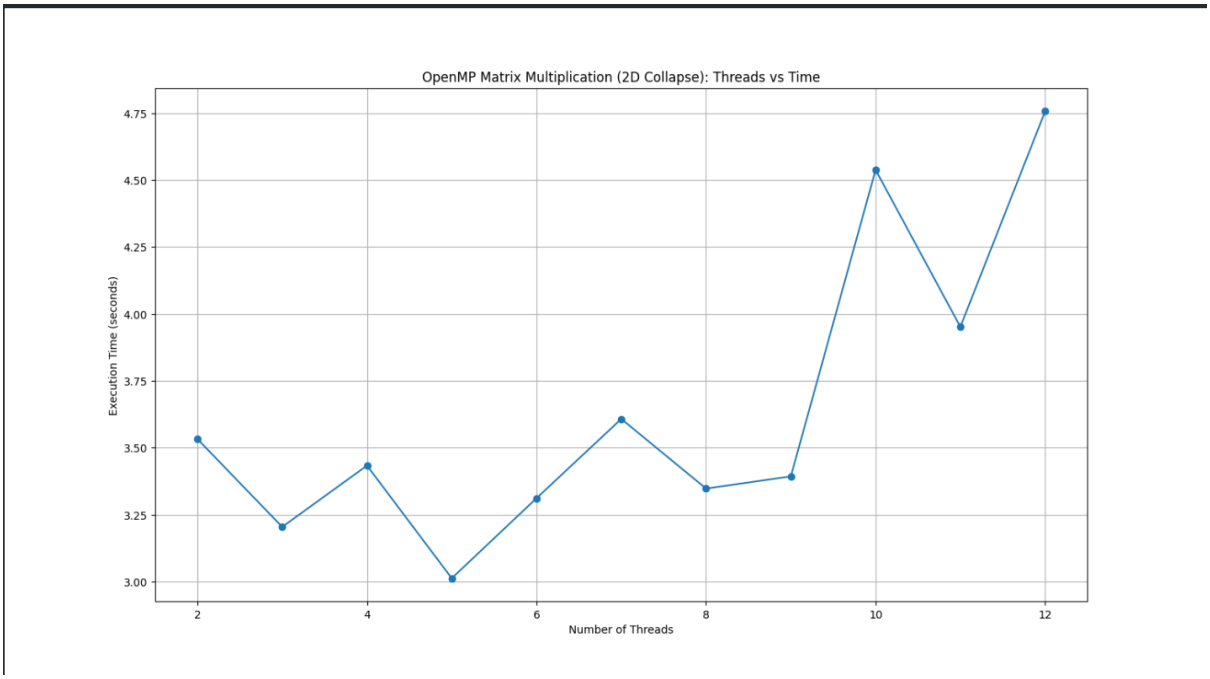
Q2. Matrix Multiplication Analysis:

```
aditya_chauhan@Dodo: $ nano matrixm_serial.c
aditya_chauhan@Dodo: $ ls
 a.out   daxpy   daxpy.c   daxpy_serial.c   eg  'eg.3 '  'eg.3 .save'   eg.c   eg3   eg3.c   matrixm   matrixm.c   matrixm_serial.c   nm   nm.c
aditya_chauhan@Dodo: $ ./matrixm
Time taken = 19.210871aditya_chauhan@Dodo: $
```

```
aditya_chauhan@Dodo: $ nano matrixm.c
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=2 ./matrixm
-bash: ./matrixm: No such file or directory
aditya_chauhan@Dodo: $ ls
 a.out   daxpy.c  'eg.3 '        eg.c   eg3.c       nm
 daxpy   eg       'eg.3 .save'   eg3    matrixm.c   nm.c
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=2 ./matrixm
-bash: ./matrixm: No such file or directory
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=2 ./matrixm.c
-bash: ./matrixm.c: Permission denied
aditya_chauhan@Dodo: $ gcc matrixm.c -fopenmp -o matrixm
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=2 ./matrixm
Time taken = 7.886953aditya_chauhan@Dodo: $ OMP_NUM_THREADS=3 ./matrixm
Time taken = 14.245350aditya_chauhan@Dodo: $ OMP_NUM_THREADS=4 ./matrixm
Time taken = 11.373993aditya_chauhan@Dodo: $ OMP_NUM_THREADS=5 ./matrixm
Time taken = 10.702409a
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=6 ./matrixm
Time taken = 12.353829aditya_chauhan@Dodo: $ OMP_NUM_THREADS=7 ./matrixm
Time taken = 12.555386aditya_chauhan@Dodo: $ OMP_NUM_THREADS=8 ./matrixm
Time taken = 11.012545aditya_chauhan@Dodo: $ OMP_NUM_THREADS=9 ./matrixm
Time taken = 13.526352aditya_chauhan@Dodo: $ OMP_NUM_THREADS=10 ./matrixm
Time taken = 12.413990aditya_chauhan@Dodo: $ OMP_NUM_THREADS=11 ./matrixm
Time taken = 15.063118aditya_chauhan@Dodo: $ OMP_NUM_THREADS=12 ./matrixm
Time taken = 13.379920aditya_chauhan@Dodo: $
```



```
aditya_chauhan@Dodo: $ nano matrixm2D.c
aditya_chauhan@Dodo: $ gcc matrixm2D.c -fopenmp -o matrixm2D
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=2 ./matrixm2D
2D Threading Time = 3.532760aditya_chauhan@Dodo: $ OMP_NUM_THREADS=3 ./matrixm2D
2D Threading Time = 3.204738aditya_chauhan@Dodo: $ OMP_NUM_THREADS=4 ./matrixm2D
2D Threading Time = 3.433706aditya_chauhan@Dodo: $ OMP_NUM_THREADS=5 ./matrixm2D
2D Threading Time = 3.012966aditya_chauhan@Dodo: $ OMP_NUM_THREADS=6 ./matrixm2D
2D Threading Time = 3.311818aditya_chauhan@Dodo: $ OMP_NUM_THREADS=7 ./matrixm2D
2D Threading Time = 3.607897aditya_chauhan@Dodo: $ OMP_NUM_THREADS=8 ./matrixm2D
2D Threading Time = 3.347843aditya_chauhan@Dodo: $ OMP_NUM_THREADS=9 ./matrixm2D
2D Threading Time = 3.393361aditya_chauhan@Dodo: $ OMP_NUM_THREADS=10 ./matrixm2D
2D Threading Time = 4.537635aditya_chauhan@Dodo: $ OMP_NUM_THREADS=11 ./matrixm2D
2D Threading Time = 3.951667aditya_chauhan@Dodo: $ OMP_NUM_THREADS=12 ./matrixm2D
2D Threading Time = 4.756428aditya_chauhan@Dodo: $
```



Observation: The results show a clear reduction in execution time as the number of threads increases, demonstrating good parallel scalability. Since matrix multiplication is compute-intensive, the workload is effectively distributed across threads, leading to significant speedup. The 2D collapse strategy performs better than 1D threading because it improves load balancing and reduces idle time among threads. However, after a certain thread, performance begins to degrade due to overheads.

Q3. π Calculation Analysis:

Problem Statement:

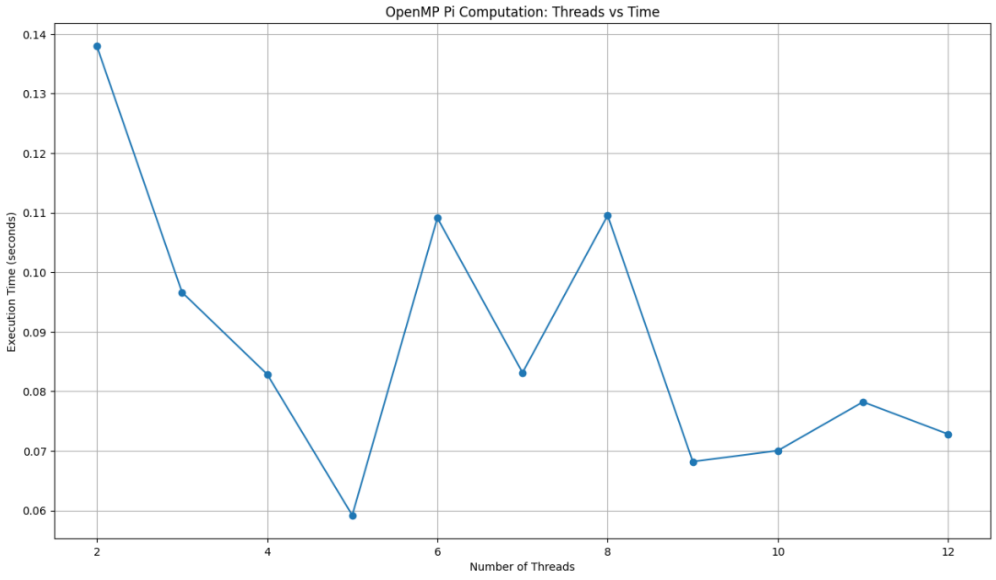
The value of π is computed using numerical integration:

$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx$$

The integral is approximated using the rectangle method and parallelized using OpenMP reduction.

```
aditya_chauhan@Dodo: $ ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.079336 seconds
aditya_chauhan@Dodo: $
```

```
aditya_chauhan@Dodo: $ ./pi
-bash: ./pi: No such file or directory
aditya_chauhan@Dodo: $ gcc pi.c -fopenmp -o pi
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=2 ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.137936 seconds
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=3 ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.096630 seconds
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=4 ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.082862 seconds
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=5 ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.059185 seconds
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=6 ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.109100 seconds
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=7 ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.083130 seconds
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=8 ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.109525 seconds
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=9 ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.068199 seconds
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=10 ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.070041 seconds
aditya_chauhan@Dodo: $ OMP_NUM_THREADS=11 ./pi
Calculated value of Pi = 3.1415926536
Time taken = 0.078215 seconds
```



Observation:

The parallel reduction approach significantly improves performance compared to the serial implementation. Execution time decreases steadily as the number of threads increases up to the number of available CPU cores. Beyond this point, the performance gain saturates and may slightly degrade due to synchronization overhead and thread management costs.