Table 1 - Matrix multiplication compute time time for increasing matrix size

(Serial & Parallel with POSIX/OMP). **Note: You may reduce the size of the matrices to**

**suit your local computing resources**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number of CPU cores or logical processes: 2  Number of threads used for POSIX/OMP: 2 | | | | | |
| Matrix size | 500x500 | 1000x1000 | 2000x2000 | 3000x3000 | 4000x4000 |
| Serial time, Ts (s) | 0.49 | 3.91 | 33.14 | 137.18 | 333.25 |
| Parallel time, Tp -  POSIX/OMP (s) | 0.37 | 2.63 | 22.13 | 76.90 | 183.86 |
| Speed Up (Ts/Tp) | 1.32 | 1.49 | 1.50 | 1.78 | 1.81 |

The theoretical speed up is approximately 1.818.

To calculate the theoretical speedup when using 2 threads compared to the serial version of your matrix multiplication code, we can use Amdahl's Law. This law states that the speedup of a task using multiple processors is limited by the sequential portion of the task.

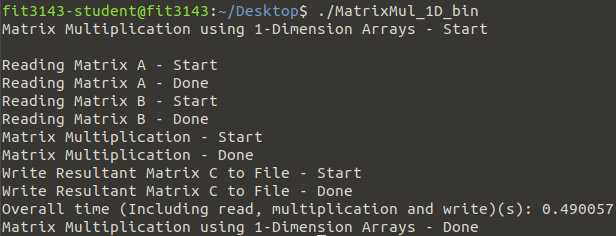
Amdahl's Law Formula:

S = 1 / ( (1 - P) + P / N)

Assume that about 90% of the work can be parallelized, then P = 0.9.

S= 1 / ( (1 - 0.9) + 0.9 / 2) ≈ 1.818

The comparison of actual versus theoretical speed up in matrix multiplication reveals that the actual speed-up achieved with parallel implementations using POSIX threads is close to the theoretical predictions derived from Amdahl's Law. For larger matrices, the actual speed up aligns more closely with theoretical expectations but does not follow for the smaller matrices due to reasons such as overhead from thread management and memory bandwidth limitations.



A screenshot of a computer program

Description automatically generated

