Student Name: Rakyan Adhikara

Student ID: 219548135

Task M2.T1P (Parallel Matrix Multiplication)

Codes are available on attached files.

Comparing methods according to data size with time taken (in seconds):

1. Between 100 and 1000 (with increment of 100)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Data Size | | | | | | | | | |
| 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
| Sequencial | 0.003732 | 0.034201 | 0.113765 | 0.273223 | 0.560687 | 1.047661 | 1.773478 | 2.738366 | 4.738423 | 7.364651 |
| pthread | 0.004283 | 0.031616 | 0.107309 | 0.265765 | 0.549814 | 1.010229 | 1.726905 | 2.678215 | 4.112499 | 6.911036 |
| OpenMP\* | 0.001855 | 0.011482 | 0.035044 | 0.090319 | 0.129169 | 0.289267 | 0.478883 | 0.780043 | 1.117878 | 1.817399 |

Graph:

1. Between 1000 and 10000 (with increment 1000)
2. Comparing between 10, 100, 1000 and 10000

Based on my findings:

1. Using parallel programming does improve performance on matrix multiplication, which is how it must be compared with sequential programming. However, OpenMP improves the performance significantly compared with pthread implementation.
2. Size of the matrices also affects the performance. In size 10 x 10, sequential has the lowest execution time. In both size 100 x 100 and 1000 x 1000, OpenMP has the lowest execution time. In addition, surprisingly, in size 100 x 100, pthread has higher execution time compared with sequential, which makes it the least efficient. Pthread also has the highest execution time in size 10 x 10.