ICSI 520 Distributed & Parallel Computing - Fall 2019

Homework 2

Santhosh Ranganathan (NetID: SR582413)

sranganathan2@albany.edu

Results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Matrix size(n)** | **Number of threads (p)** | **Serial execution time (microseconds)** | **Parallel execution time (microseconds)** | **Speedup** |
| 10 | 4 | 7.4 | 177.8 | 0.04162 |
| 30 | 4 | 150.8 | 313.8 | 0.48056 |
| 50 | 4 | 705.6 | 658.4 | 1.07169 |
| 75 | 4 | 2279.2 | 1533.2 | 1.48656 |
| 100 | 4 | 5323.6 | 3535.8 | 1.50563 |
| 300 | 4 | 145634 | 48507.2 | 3.00232 |
| 500 | 4 | 717477.6 | 227220.4 | 3.15763 |
| 750 | 4 | 2398140.6 | 774566.6 | 3.09611 |
| 750 | 8 | 2398140.6 | 523402.2 | 4.58183 |
| 1000 | 4 | 5621185.8 | 1821879.6 | 3.08538 |
| 1000 | 8 | 5621185.8 | 1271605 | 4.42054 |
| 1000 | 16 | 5621185.8 | 740908.8 | 7.58688 |
| 1000 | 24 | 5621185.8 | 510305 | 11.01535 |

make A.csv B.csv MAT\_SIZE=N

srun -n1 -c1 Homework2\_Serial.out

srun -n1 -c4 Homework2\_Parallel.out 4

(Averaging 5 runs)

(Number of cores is kept same as number of threads)

Explanation:

Parallel execution time is slower for small size matrices due to overhead from thread creation. From the table 4 cores sees maximum speedup by cores ratio and it drops to half and below for more than 8 cores. This suggests my code is not optimized for multiple cores and there are significant more cache misses. No improvement was observed with changing the second operand to column major. Perhaps assigning adjacent rows instead of assigning blocks of rows to a single core would improve performance.