**s = 100**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P | Time | | | | Speedup | | | | Efficiency | | | |
| ijk | ikj | jki | mxm2 | ijk | ikj | jki | mxm2 | ijk | ikj | jki | mxm2 |
| 1 | 6.533 | 6.358 | 6.413 | 6.26 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2.95 | 2.948 | 2.925 | 2.849 | 2.215 | 2.157 | 2.192 | 2.197 | 1.107 | 1.078 | 1.096 | 1.099 |
| 4 | 2.438 | 2.446 | 2.49 | 2.477 | 2.68 | 2.599 | 2.576 | 2.527 | 0.67 | 0.65 | 0.644 | 0.632 |
| 8 | 1.295 | 1.255 | 1.436 | 1.283 | 5.045 | 5.066 | 4.466 | 4.879 | 0.631 | 0.633 | 0.558 | 0.61 |
| 16 | 0.708 | 0.625 | 3.052 | 0.652 | 9.227 | 10.17 | 2.101 | 9.601 | 0.577 | 0.636 | 0.131 | 0.6 |
| 24 | 0.637 | 0.457 | 2.284 | 0.489 | 10.26 | 13.91 | 2.808 | 12.8 | 0.427 | 0.58 | 0.117 | 0.533 |
| 36 | 0.43 | 0.326 | 2.803 | 0.353 | 15.19 | 19.5 | 2.288 | 17.73 | 0.422 | 0.542 | 0.064 | 0.493 |
| 48 | 1.616 | 1.029 | 3.497 | 0.324 | 4.043 | 6.179 | 1.834 | 19.32 | 0.084 | 0.129 | 0.038 | 0.403 |



**s = 1000**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P | Time | | | | Speedup | | | | Efficiency | | | |
| ijk | ikj | jki | mxm2 | ijk | ikj | jki | mxm2 | ijk | ikj | jki | mxm2 |
| 1 | 4463 | 4533 | 4751 | 4458 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2240 | 2272 | 2419 | 2242 | 1.992 | 1.995 | 1.964 | 1.989 | 0.996 | 0.998 | 0.982 | 0.994 |
| 4 | 1162 | 1150 | 1196 | 1126 | 3.841 | 3.94 | 3.972 | 3.959 | 0.96 | 0.985 | 0.993 | 0.99 |
| 8 | 573.5 | 574.8 | 603.5 | 568 | 7.782 | 7.886 | 7.872 | 7.849 | 0.973 | 0.986 | 0.984 | 0.981 |
| 16 | 319.6 | 305 | 310 | 290.4 | 13.97 | 14.86 | 15.33 | 15.35 | 0.873 | 0.929 | 0.958 | 0.959 |
| 24 | 205.3 | 196.6 | 217.6 | 205.4 | 21.74 | 23.05 | 21.84 | 21.7 | 0.906 | 0.961 | 0.91 | 0.904 |
| 36 | 148.8 | 146.5 | 160.6 | 133.2 | 29.99 | 30.94 | 29.59 | 33.47 | 0.833 | 0.86 | 0.822 | 0.93 |
| 48 | 120.4 | 115.9 | 127.1 | 113.9 | 37.06 | 39.12 | 37.37 | 39.13 | 0.772 | 0.815 | 0.779 | 0.815 |



**s = 3000**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P | Time | | | | Speedup | | | | Efficiency | | | |
| ijk | ikj | jki | mxm2 | ijk | ikj | jki | mxm2 | ijk | ikj | jki | mxm2 |
| 1 | 198279 | 123204 | 202265 | 123116 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 125207 | 64594.4 | 116616 | 63621.9 | 1.58361 | 1.90735 | 1.73445 | 1.93511 | 0.7918 | 0.95368 | 0.86723 | 0.96756 |
| 4 | 73943.7 | 32260 | 79532 | 31806.7 | 2.68148 | 3.8191 | 2.54319 | 3.87074 | 0.67037 | 0.95477 | 0.6358 | 0.96769 |
| 8 | 33749.8 | 16456.2 | 34307.4 | 16280.3 | 5.87496 | 7.4868 | 5.89567 | 7.56224 | 0.73437 | 0.93585 | 0.73696 | 0.94528 |
| 16 | 16956.8 | 7956.93 | 16676 | 7842.72 | 11.6932 | 15.4839 | 12.1291 | 15.6981 | 0.73082 | 0.96774 | 0.75807 | 0.98113 |
| 24 | 10581.4 | 5830.2 | 10827.6 | 5738.33 | 18.7385 | 21.1321 | 18.6805 | 21.455 | 0.78077 | 0.8805 | 0.77835 | 0.89396 |
| 36 | 7641.14 | 4029.34 | 8178.95 | 3937.75 | 25.9489 | 30.5768 | 24.7299 | 31.2655 | 0.7208 | 0.84936 | 0.68694 | 0.86849 |
| 48 | 5969.56 | 3353.08 | 7023.35 | 3339.76 | 33.215 | 36.7436 | 28.7989 | 36.8636 | 0.69198 | 0.76549 | 0.59998 | 0.76799 |



**s = 5000**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P | Time | | | | Speedup | | | | Efficiency | | | |
| ijk | ikj | jki | mxm2 | ijk | ikj | jki | mxm2 | ijk | ikj | jki | mxm2 |
| 1 | 988481 | 634927 | 1068827 | 624723 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 507847 | 302456 | 537548 | 297505 | 1.94642 | 2.09924 | 1.98834 | 2.09987 | 0.97321 | 1.04962 | 0.99417 | 1.04994 |
| 4 | 317035 | 148789 | 319974 | 146649 | 3.1179 | 4.26731 | 3.34036 | 4.25998 | 0.77947 | 1.06683 | 0.83509 | 1.065 |
| 8 | 164553 | 72007.1 | 173999 | 70841.4 | 6.00707 | 8.81757 | 6.14274 | 8.81861 | 0.75088 | 1.1022 | 0.76784 | 1.10233 |
| 16 | 77363.3 | 36438.1 | 88065.5 | 35869.6 | 12.7771 | 17.4248 | 12.1367 | 17.4165 | 0.79857 | 1.08905 | 0.75855 | 1.08853 |
| 24 | 56193.1 | 24677.1 | 61465 | 24317.4 | 17.5908 | 25.7294 | 17.3892 | 25.6904 | 0.73295 | 1.07206 | 0.72455 | 1.07043 |
| 36 | 38759 | 17226.1 | 44447.3 | 17040.4 | 25.5032 | 36.8585 | 24.0471 | 36.6612 | 0.70842 | 1.02385 | 0.66797 | 1.01837 |
| 48 | 29212.8 | 14416.8 | 33819.2 | 14073.3 | 33.8373 | 44.0407 | 31.6041 | 44.3906 | 0.70494 | 0.91751 | 0.65842 | 0.9248 |



**Analysis**

**1. Performance of different algorithm variations:**

Ranking based on performance: mxm2 > ikj > ijk > jki.

There are two pairs with similar performance: ikj-mxm2, and ijk-jki. ikj and mxm2 have better performance than the other two variations, especially at large problem sizes. At s = 5000, they are more than 2 times faster. ikj and mxm2 also have better speedup and efficiency when parallelization is used.

ikj is faster because in the innermost loop, matrix1[i \* size + k] is constant, leading to a much better temporal locality. This is also true for jki, but it has a terrible spatial locality, which dominates the improved temporal locality.

mxm2 has a superior spatial locality, making it the fastest algorithm. Although transposing B costs some time, it’s not significant because mmT only takes O(n^2) time, whereas matrix multiplication takes O(n^3) time.

**2. Impact of problem size:**

As expected, increasing the problem size also increases the runtime. For s >= 1000, the speedup achieved by parallelization is almost linear, and the efficiency is quite good, fluctuating between 0.6-1.1.

This is not the case when s = 100. When we go from P = 36 to P=48, the performance is actually worse (except for mxm2). This is because 100/36 and 100/48 can both be rounded up to 3. In other words, the maximum number of iterations a thread has to handle is 3 for both P=36 and P=48, so no gain in performance is expected because we have to wait for the slowest thread to finish. P=48 can be slower because of the cost of creating threads and sharing resources.

**3. Scalability of parallelization:**

-The speedup is almost linear for s >= 1000.

- ikj and mxm2 have better speedup and efficiency than the others, especially at large problem sizes.

- In some cases we achieved a greater-than-1 efficiency.

- In general, efficiency goes down slowly as we add more threads.