In my openMP program for the matrix multiplication part I use atlas call. And use openMP to parallelise the dividing matrix part and multiply them in parallel in order to gain more efficiency. In the experiment we request 8 threads from openMP and pass difference size matrix A and B to the algorithm (where the matrix is square matrix size of n\*n), we generate random number to fill in the n\*n matrices A and B, for then serial program we use atlas call in cblas as well to maximise the fairness for the comparison.

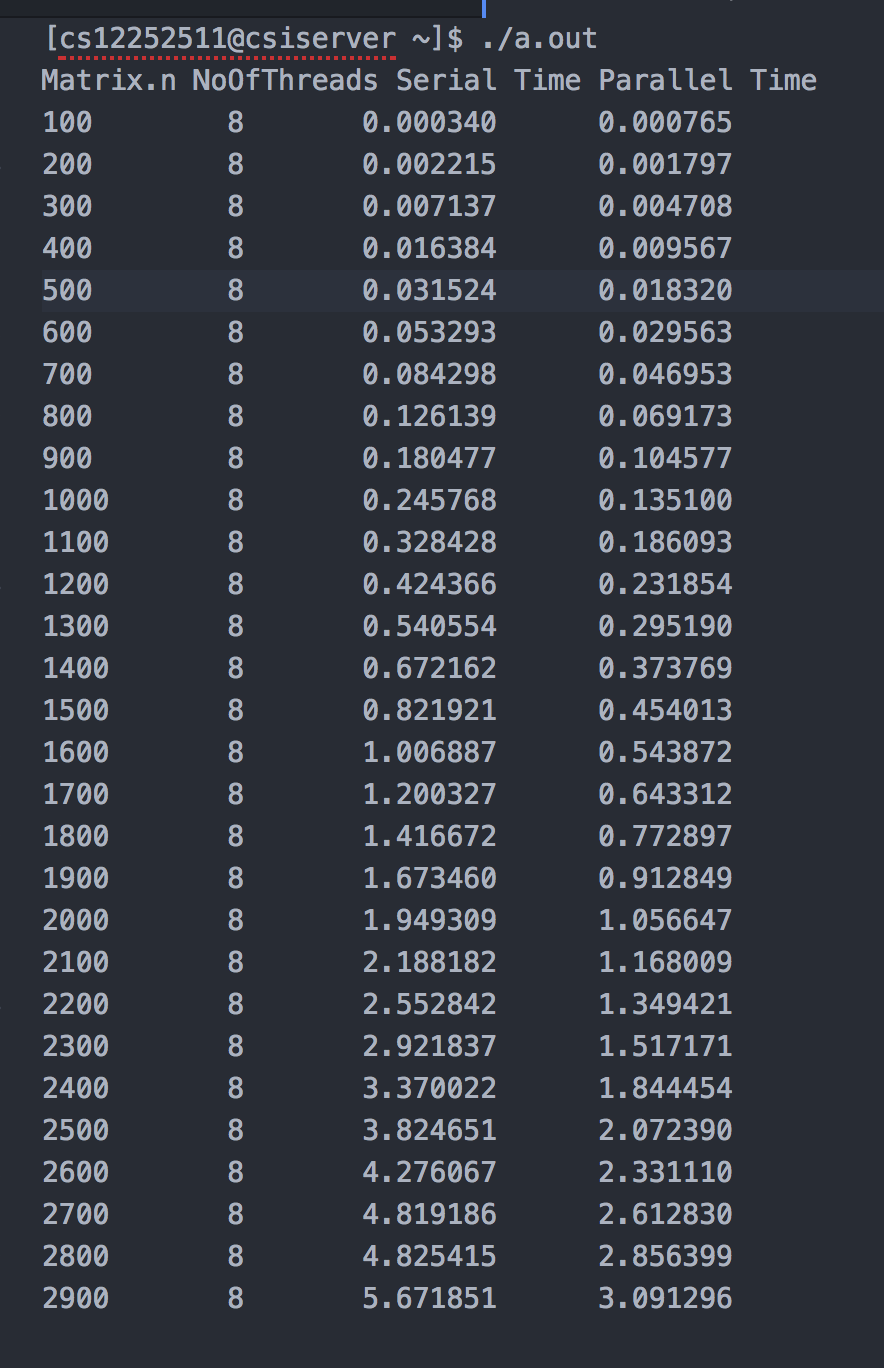
**1) The dependence of the execution time of the program on the matrix size n.**

Below graph shows that the dependence between the execution time of the program(blue for openMP parallel altas call program and orange for serial altas call program) on the matrix size n. We notices that regardless which program they both have a linear relation with the matrix size n where n is larger the execution time gets longer. But for small matrix size there are no significant difference between the parallel openMP version and the serial one but while we increase the size of the matrix the advantage of parallelism become really obvious. As shown in the graph.

**2) The speedup over a serial counterpart of the program.**

As we can observe from the graph below which shows the speedup over a serial counterpart of the program vs the matrix size N, if the matrix size is small then our serial program is actually doing faster than the openMP parallel version. And this is due to the openMP parallelisation overhead. However, after the matrix size n increase to N =200 we can see that our openMP version is 23.2% faster then our serial version. If we go even larger N = 1000 we get almost as twice as faster(openMP) than the serial program.

|  |  |
| --- | --- |
| Matrix size N | Speed up time serial/parallel |
| 100 | 0.444444444 |
| 200 | 1.232609905 |
| 300 | 1.515930331 |
| 400 | 1.71255357 |
| 500 | 1.720742358 |
| 600 | 1.802692555 |
| 700 | 1.795369838 |
| 800 | 1.823529412 |
| 900 | 1.725781003 |
| 1000 | 1.819156181 |
| 1100 | 1.764859506 |
| 1200 | 1.83031563 |
| 1300 | 1.831207019 |
| 1400 | 1.798335335 |
| 1500 | 1.81034684 |

Below is the output of the program.