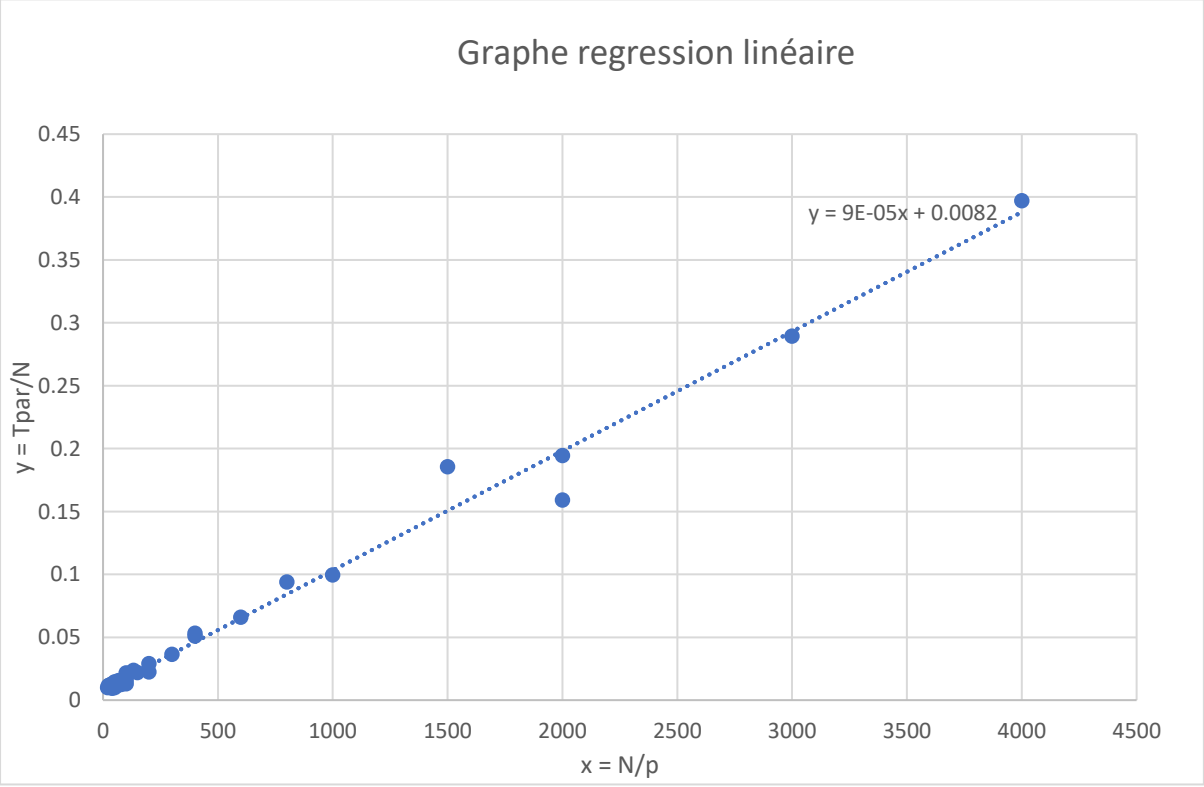
Results: (nb iterations = 1800)

SIZE : NB TASKS :	2000x2000 (Tseq + Tp)	3000x3000 (Tseq + Tp)	4000x4000 (Tseq + Tp)	2000x2000 (Tp)	3000x3000 (Tp)	4000x4000 (Tp)
1	06 :40	14 :32	26 :35	06 :38	14 :28	26 :28
2	03:21	09 :21	10 :43	03 :19	09 :17	10 :36
5	01:44	03 :22	06 :23	01 :42	03 :18	06 :16
10	00 :47	01 :53	03 :40	00 :45	01 :49	03 :33
20	00 :28	01 :10	02 :03	00 :26	01 :06	01 :56
30	00 :33	00 :52	01 :42	00 :31	00 :48	01 :35
40	00 :31	00 :45	01 :34	00 :29	00 :41	01 :27
50	00 :28	00 :40	00 :57	00 :26	00 :36	00 :50
60	00 :26	00 :34	00 :55	00 :24	00 :30	00 :48
70	00 :26	00 :37	00 :57	00 :24	00 :33	00 :50
80	00 :23	00 :40	00 :55	00 :21	00 :36	00 :48
90	00 :25	00 :38	00 :52	00 :23	00 :34	00 :45
100	00 :22	00 :35	00 :45	00 :20	00 :31	00 :38

These are the results for the total execution time (Tseq + Tp), the sequential part of the code is not affected by the total number of cores awarded for this computation, it is only dependant on one core, and the total size of the matrix.

Size:	2000x2000	3000x3000	4000x4000
Sequential Time (s):	2.34074	4.47615	7.14863

Linear regression:



And the Equation of this linear regression is:

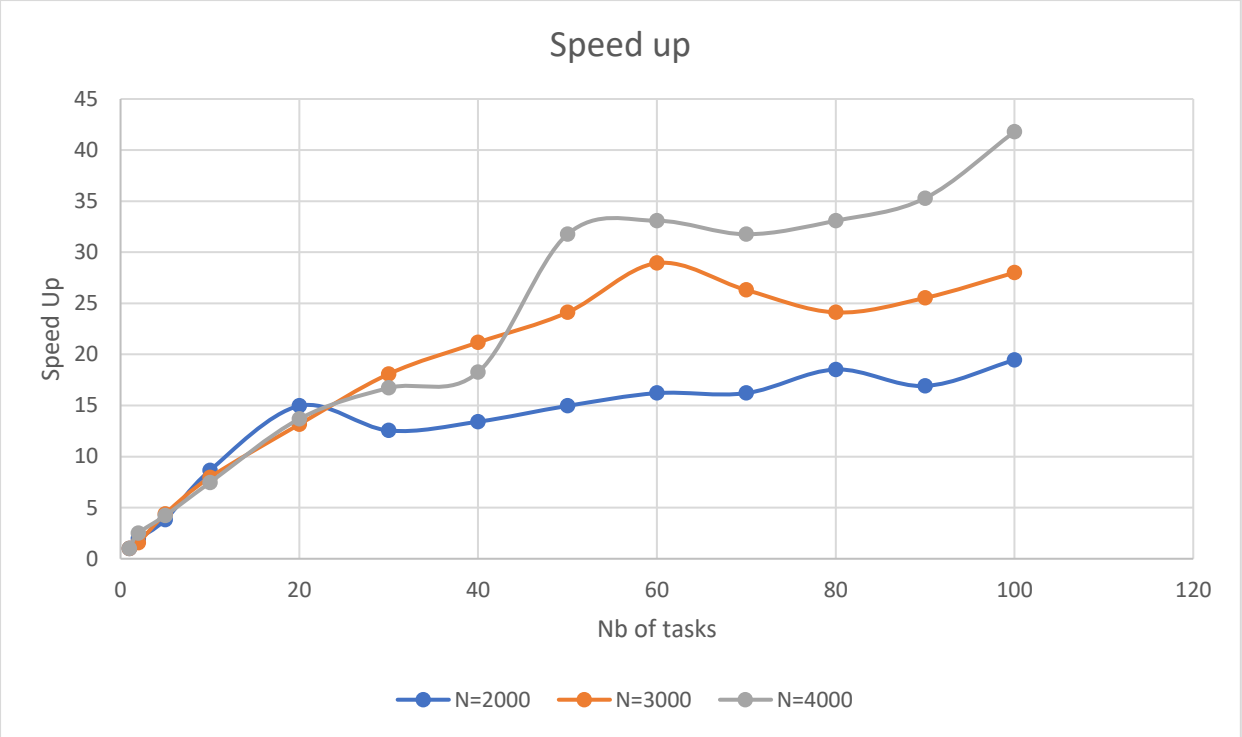
Y = (9*10^(-5)) * x + 0.0082

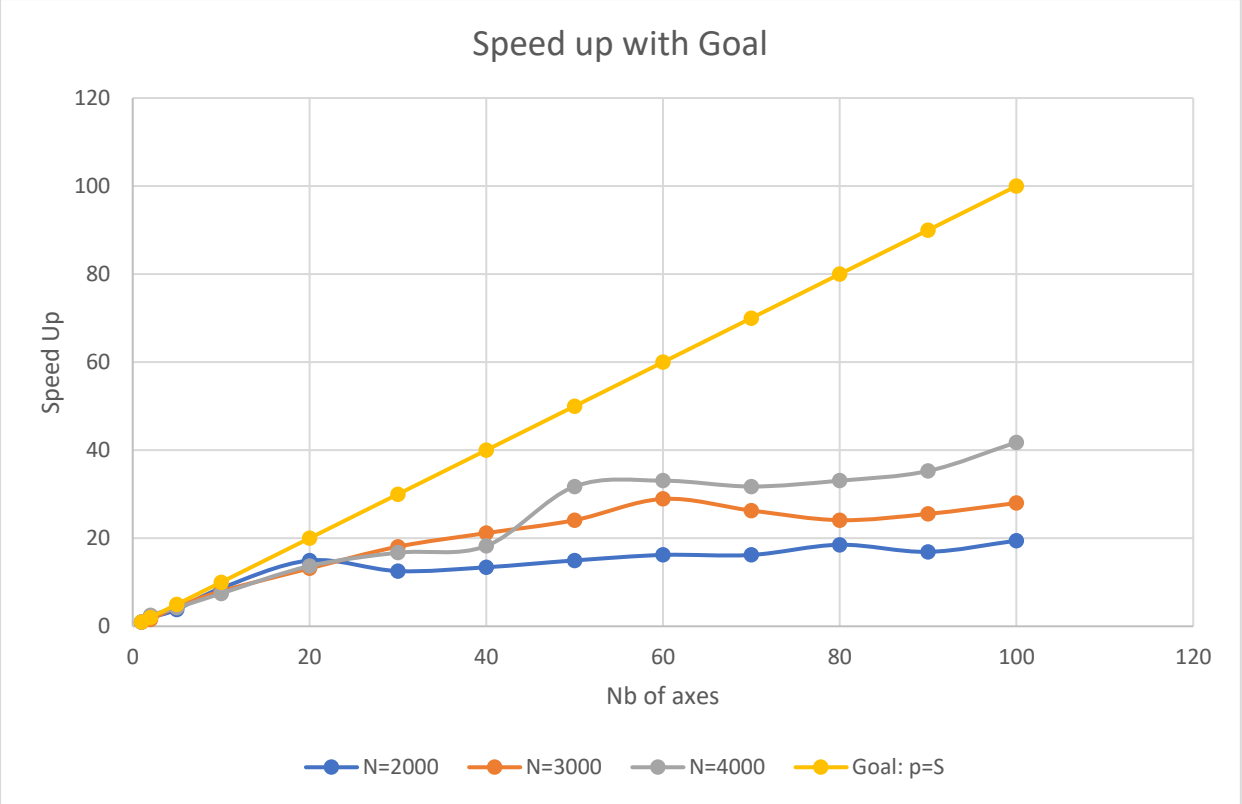
Which means:

Alpha = 9*10^(-5)

Beta = 0.0082

Speed Up:





It is easier to see how far off we are from our goal if we map the goal line.

Discussion of results:

The most interesting thing for me, is that the speed up gets much better results for a bigger matrix, so if we were to do computations with larger matrices, we would technically get better speed up ratios, and eventually get closer to the Goal line. Not predicting that we will get to 100 speed up, but the closer the better.