Assignment 3 - Parallel Computation of Matrix Norm
COMP30250 - Parallel and Cluster Computing

November 14, 2021

Aleryc SERRANIA - N°21204068

1 Introduction

The variants for this assignments are the following:

1. Compute the norm in two successive steps: parallelisation of matrix multiplication, then parallelisation of matrix norm computation
2. Left matrix is horizontally partitioned
3. Compute 1-norm (maximum absolute column sum norm)

The command `cat /proc/cpuinfo` displays 16 processors, therefore `p` will be 16 for the parallel programs.

The timing unit is the second.

2 Benchmarks of parallel and serial programs for difference matrix size

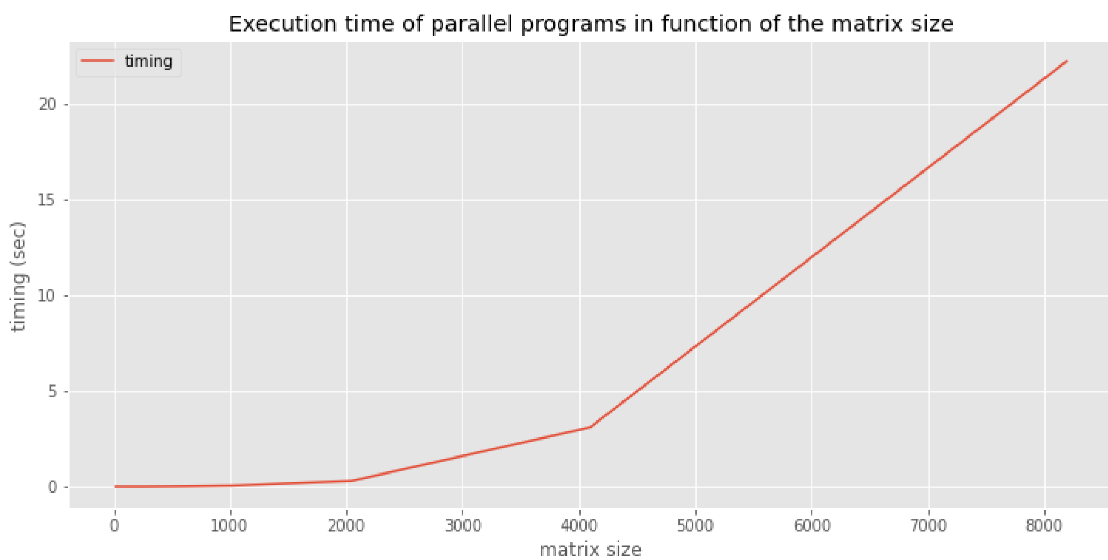
2.1 Benchmark of parallel programs

	filename	matrix_size	nb_threads	timing
0	matrixnorm.out	16	16	0.002613
1	matrixnorm.out	32	16	0.003028
2	matrixnorm.out	64	16	0.003179
3	matrixnorm.out	128	16	0.003454
4	matrixnorm.out	256	16	0.003391
5	matrixnorm.out	512	16	0.014273
6	matrixnorm.out	1024	16	0.054600
7	matrixnorm.out	2048	16	0.304863
8	matrixnorm.out	4096	16	3.092737
9	matrixnorm.out	8192	16	22.181388

2.2 Benchmark of serial programs

	filename	matrix_size	nb_threads	timing
0	matrixnorm_serial.out	16	1	0.000028
1	matrixnorm_serial.out	32	1	0.000052
2	matrixnorm_serial.out	64	1	0.000200
3	matrixnorm_serial.out	128	1	0.000806
4	matrixnorm_serial.out	256	1	0.004914
5	matrixnorm_serial.out	512	1	0.032421
6	matrixnorm_serial.out	1024	1	0.246230
7	matrixnorm_serial.out	2048	1	1.863570
8	matrixnorm_serial.out	4096	1	16.219560
9	matrixnorm_serial.out	8192	1	120.093703

3 Dependence of the execution time of the parallel program on the matrix size n



4 Speedup over a serial counterpart of the program

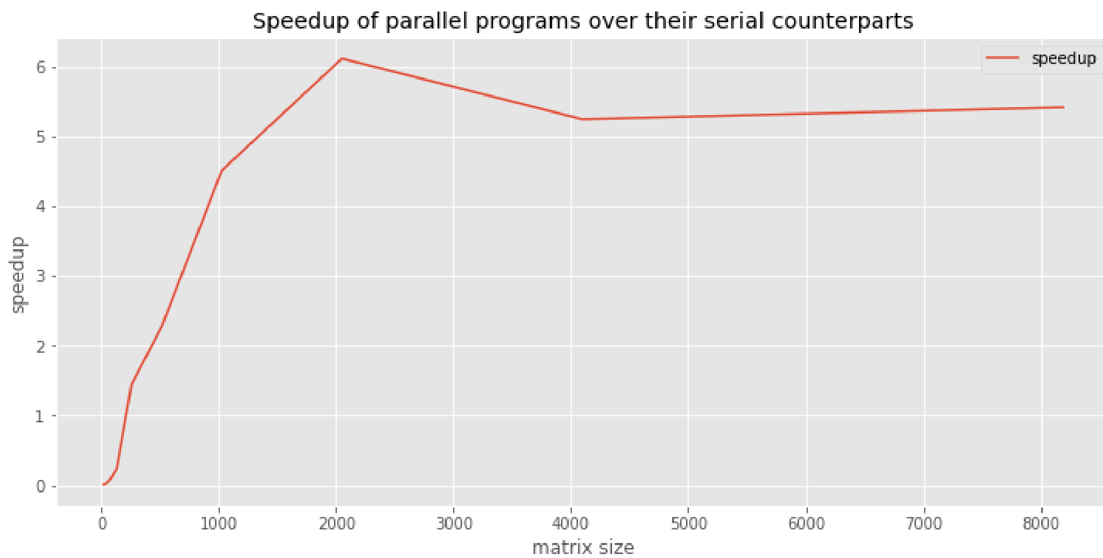
Speedup is calculated as follows:

$$S(m) = \frac{T_{serial}(m)}{T_{parallel}(m)}$$

where m is matrix size

	filename	matrix_size	nb_threads	timing	speedup
0	matrixnorm.out	16	16	0.002613	0.010669

1	matrixnorm.out	32	16	0.003028	0.017028
2	matrixnorm.out	64	16	0.003179	0.062768
3	matrixnorm.out	128	16	0.003454	0.233375
4	matrixnorm.out	256	16	0.003391	1.449025
5	matrixnorm.out	512	16	0.014273	2.271476
6	matrixnorm.out	1024	16	0.054600	4.509736
7	matrixnorm.out	2048	16	0.304863	6.112822
8	matrixnorm.out	4096	16	3.092737	5.244403
9	matrixnorm.out	8192	16	22.181388	5.414165



The results show that we have significant speedup for matrix of size n > 200 and the speedup stabilises around 5.5 for bigger matrices.

The parallelisation is inefficient for small matrices as it reduces the processing speed. This problem stems from the overhead of creating and initializing new threads.

Speedup stabilisation for matrix of size n > 2000 is due to the fact that this overhead cost is negligible compared to the computation time of matrix slices.