Exercise 01

Strong and weak scalability of an HPC application

Scalability is related to an increase in performance when more computing elements are used in parallel, such as more processors, vector units, memory, or network connections. In this context, the concept can be differentiated in strong and weak scalability.

The term "strong scaling" is associated to how performance varies with the number of computing elements for a fixed total problem size. In contrast, the term "weak scaling" is associated to how performance varies with the number of computing elements for a fixed problem size per processor, and additional computing elements are used to solve a larger total problem[1].

For the completion of this exercise, it was given two computer codes, based on the Monte Carlo method that integrate a quarter of a unit circle to compute the number PI (given as area_computed*4), the first one being a basic implementation of the algorithm ("pi.c") and the second one being a parallel MPI implementation of the same algorithm ("MPIpi.c"). The exercise, here, is to understand how well the application scales up to the total number of cores when considering one node.

The time required to compute the code may be accessed using the command line "time", as the following example:

```
$ time mpirun -np 1 ./a.out 1000000000
```

Strong Scaling

In order to compute the strong scaling, the number of trials of the code was kept constant while the number of processors was considered in increasing number. Table 1 and table 2 show the data obtained after three different runs considering 10^9 and 10^{10} the number of trials, respectively. One example of output is presented in appendix A.

Table 1: Strong Scaling Experiment for 10^9 number of trials (cn07-31)

#processors	Run 1 s	Run 2 s	Run 3 s	Mean s	Std s	Speedup
1	21,131	21,139	21,089	21,120	0,027	1,000
2	11,681	11,619	11,602	11,634	0,042	1,815
4	6,599	6,618	6,627	6,615	0,014	3,193
8	4,378	4,359	4,281	4,339	0,051	4,867
16	3,216	3,203	3,137	3,185	0,042	6,630
20	2,996	2,939	2,959	2,965	0,029	7,124

[1] Optimizing HPC Applications with Intel Cluster Tools, Paperback – October 15, 2014 by Alexander Supalov, Andrey Semin, Michael Klemm, ISBN-13: 978-1430264965 ISBN-10: 1430264969 Edition: 1st.

Exercise 01

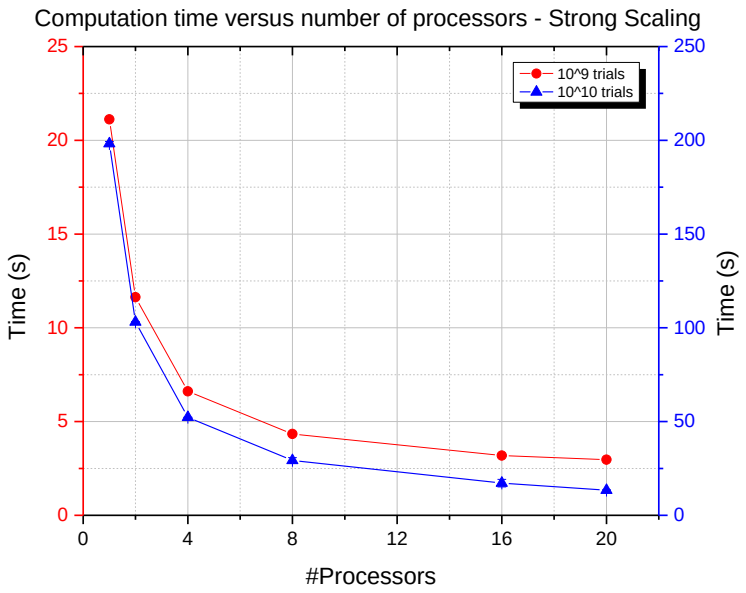
Table 2: Strong Scaling Experiment for 10^10 number of trials (cn07-31)

#processors	Run 1 s	Run 2 s	Run 3 s	Mean s	Std s	Speedup
1	199,569	197,446	197,689	198,235	1,162	1,000
2	103,238	103,151	102,854	103,081	0,201	1,923
4	52,351	52,242	52,132	52,242	0,110	3,795
8	31,016	28,386	28,385	29,262	1,519	6,774
16	15,994	15,993	19,499	17,162	2,024	11,551
20	13,196	13,273	13,655	13,375	0,246	14,822

The Speedup for “p” processors was computed as follows:

Speedup (p) = Execution time for a single processor/Execution time using “p” parallel processors

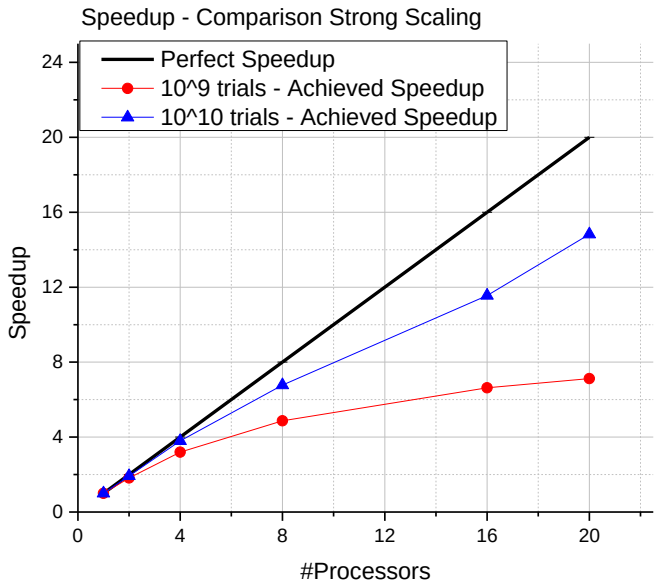
In order to understand how the computation time varies with the number of processors, graph 1 was plotted.



Graph 1: Computation time versus number of processors (10^9 and 10^{10} trials).

As expected, the computation time decreases exponentially and the same behaviour is observed regardless the number of trials. In order to compare the achieved Speedup with the perfect (ideal) one, graph 2 was plotted.

Exercise 01



Graph 2: Comparison between the perfect Speedup and the achieved Speedup (10^9 and 10^{10} trials).

It is possible to notice from graph 2 that both curves of the achieved Speedup move away from the linear perfect Speedup with the increase in number of processors. This behaviour is explained by the Amdahl's law, that is: "The performance improvement to be gained by parallelisation is limited by the proportion of the code which is serial". However, the 10^{10} trials case presents a higher achieved Speedup.

Weak Scaling

In order to compute the weak scaling, the number of iterations as well as the number of processors were considered in increasing number, in such a way that the problem size per processor was kept constant (trials per processor). Tables 3 and 4 show the data obtained after three different runs considering 10^9 and 10^{10} the number of trials per processor, respectively.

Table 3: Weak Scaling Experiment for 10^9 number of trials per processor (cn07-31)

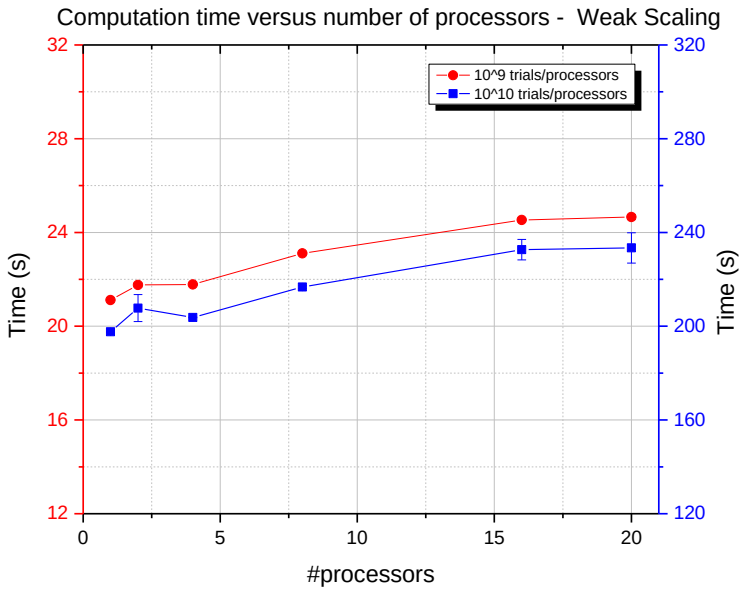
#processors	Run 1 s	Run 2 s	Run 3 s	Mean s	Std s	Speedup
1	21,131	21,098	21,113	21,114	0,017	1,000
2	21,858	21,726	21,7	21,761	0,085	0,970
4	21,792	21,785	21,764	21,780	0,015	0,969
8	23,08	23,107	23,142	23,110	0,031	0,914
16	24,495	24,552	24,546	24,531	0,031	0,861
20	24,623	24,779	24,571	24,658	0,108	0,856

Exercise 01

Table 4: Weak Scaling Experiment for 10^10 number of trials per processor (cn07-31)

#processors	Run 1 s	Run 2 s	Run 3 s	Mean s	Std s	Speedup
1	197,441	197,86	197,55	197,617	0,217	1,000
2	214,372	204,327	204,529	207,743	5,742	0,951
4	203,756	203,871	203,625	203,751	0,123	0,970
8	217,37	216,423	216,374	216,722	0,561	0,912
16	237,656	230,814	229,554	232,675	4,360	0,849
20	229,544	240,854	229,825	233,408	6,450	0,847

In order to understand how the computation time varies with the number of processors, graph 3 was plotted.

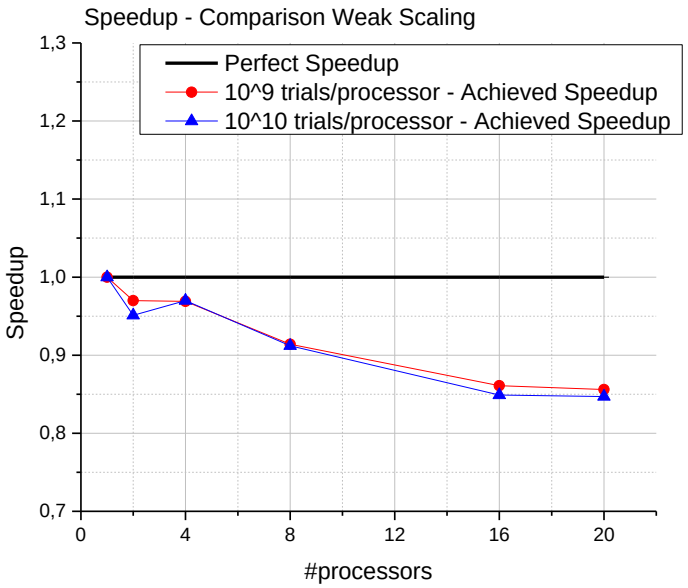


Graph 3: Computation time versus number of processors (10^9 and 10^{10} trials/processor).

As expected, the computation time is kept approximately constant (although with a slightly tendency to increase) and the same behaviour is observed regardless the number of trials. This behaviour is explained by the Gustafson's law, that states that if the amount of work done by each parallel task is increased, then the serial component will not dominate.

Exercise 01

In order to compare the achieved Speedup with the perfect (ideal) one, graph 4 was plotted.



Graph 4: Comparison between the perfect Speedup and the achieved Speedup (10^9 and 10^{10} trials/processor).

It is possible to notice, from graph 4, that both curves of speedup decrease with the increase of the number of processors in the same behaviour, regardless of the number of trials per processor.

Considering the presented results, the problem studied is believed to be more appropriate for strong scaling, since the related speedup approximates to the ideal with the increase of trials and processors and the time required decreases under the same conditions.

Exercise 01

APPENDIX A - Output:

```
$ time mpirun -np 1 ./a.out 10000000000

 # of trials = 10000000000 , estimate of pi is 3.141596178

 # walltime on master processor : 198.05153513

real    3m19.569s
user    3m19.279s
sys    0m0.075s
```