FHPC, Assignment 1: Report

Matteo Poggi

November 13, 2020

1 Theoretical Model

Let us denote by N the number of iteration and by P the number of cores. Then the contribution to the execution time $T(P, N)$ are the following:

- T_{read}: time to read numbers;
- $(P-1)T_{\text{comm.}}$: time to communicate to $P-1$ slaves;
- $\left(\left\lceil \frac{N}{P} \right\rceil - 1\right) T_{\text{comp.}}$: time for each core to compute the sum;
- $(P-1)T_{\text{comm.}}$: time to collect the result;
- $(P-1)T_{\text{comp.}}$: time to sum up all the partial results.

All in all we have

$$T(P, N) = T_{\text{read}} + 2(P-1)T_{\text{comm.}} + \left(\left\lceil \frac{N}{P} \right\rceil + P - 2\right) T_{\text{comp.}} \,. \tag{1.1}$$

(There is a slight discrepancy with the text of the assignment, but it disappears for large N.) We will use the values shown in Table 1.

time	**value** $[s]$
T_{read}	10^{-4}
$T_{\text{comm.}}$	10^{-6}
$T_{\text{comp.}}$	$2 \cdot 10^{-9}$

Table 1: Value of various times in our cluster.

In Figure 1 we plot the function $T(P, N)$ for various value of N. For a fixed N it is possible to seek for the minimum analytically (discarding the upper integer part): we have

$$P_{\text{min.}} \simeq \sqrt{N} \cdot \sqrt{\frac{T_{\text{comp.}}}{2T_{\text{comm.}} + T_{\text{comp.}}}} \,. \tag{1.2}$$

In Figure 2 we show the analytical solution for $P_{\text{min,}}$ against its effective value. The small discrepancies are due to the suppression of upper integer part.

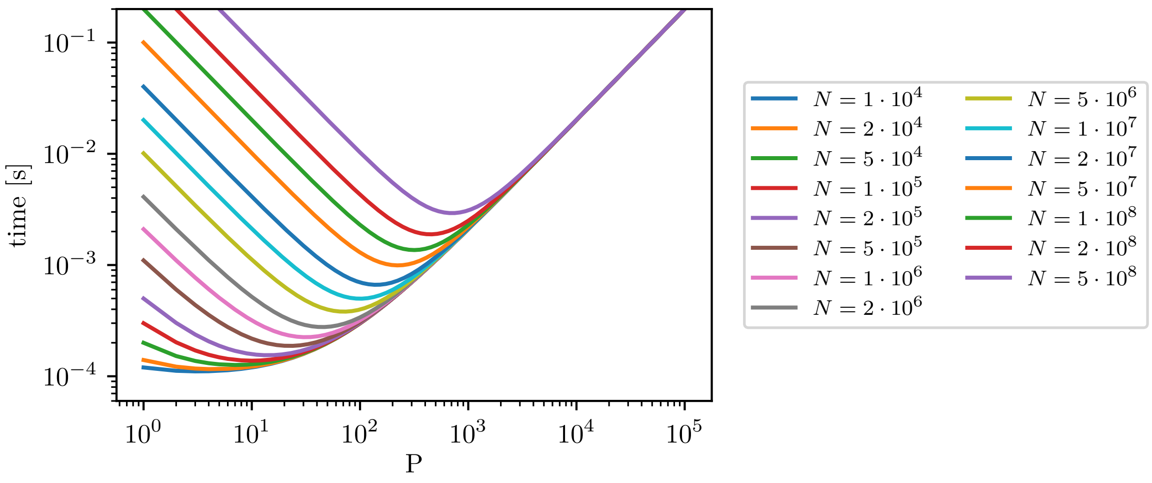

Figure 1: Theoretical model: parallel time. We can see the non-monotonous behavior which implies the existence of a $P_{\text{min.}}$

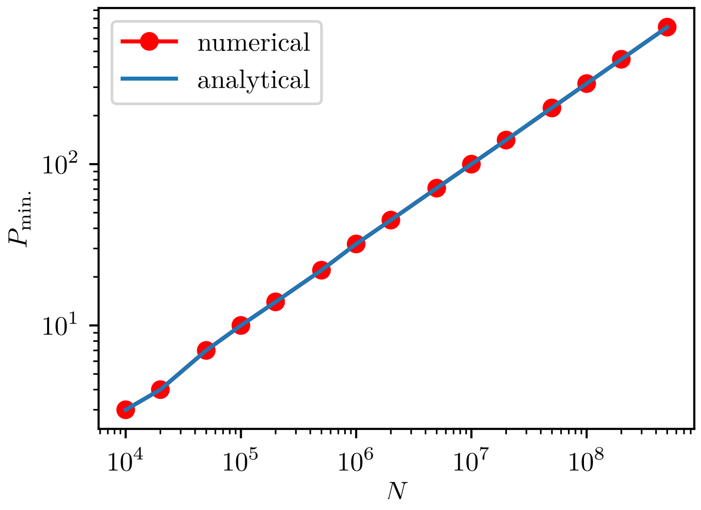

Figure 2: Numerical and analytical value of $P_{\text{min.}}$

1.1 Scaling Analysis

We define the speedup function as

$$S(P, N) = \frac{T(1, N)}{T(P, N)} \ . \tag{1.3}$$

In the perfect scaling limit one would have the linear relation $S(P, N) \sim P$. However this linear regime is spoiled by non linear effect of the overheads. This effect will become more important for larger and larger P. In order to develop an analytic intuition we can expand for "small P" (we will clarify later the meaning of "small").

$$S(P, T) = s_1 P - s_2 P^2 + \mathcal{O}(P^3) \ , \tag{1.4}$$

with

$$s_1 = \frac{T_{\text{read}} + (N-1)T_{\text{comp.}}}{N T_{\text{comp.}}}\,, \qquad s_2 = \frac{(T_{\text{read}} - 2T_{\text{comm.}} - 2T_{\text{comp.}})(T_{\text{read}} + (N-1)T_{\text{comp.}})}{N^2 T_{\text{comp.}}^2}\,. \tag{1.5}$$

With our value of T_{read}, $T_{\text{comm.}}$ and $T_{\text{comp.}}$ we have $s_1 > 0$ and $s_2 > 0$. The latter, in particular, tells us that the non-linearity has a worsening effect on our performances, as we expect. However this non-linearity will be negligible if $s_1 P \gg s_2 P^2$ that is if

$$N \gg \frac{T_{\text{read}} - 2T_{\text{comm.}} - 2T_{\text{comp.}}}{T_{\text{comp.}}} P\,. \tag{1.6}$$

This gives an scale of smallness for P. With our values we have that the non-linearity effect are negligible if

$$N \gg 4.9 \cdot 10^5 P. \tag{1.7}$$

The graph in Figure 3 reports in log-log scale the speedup factor for various values of N. From it one can understand when the linear regime is valid.

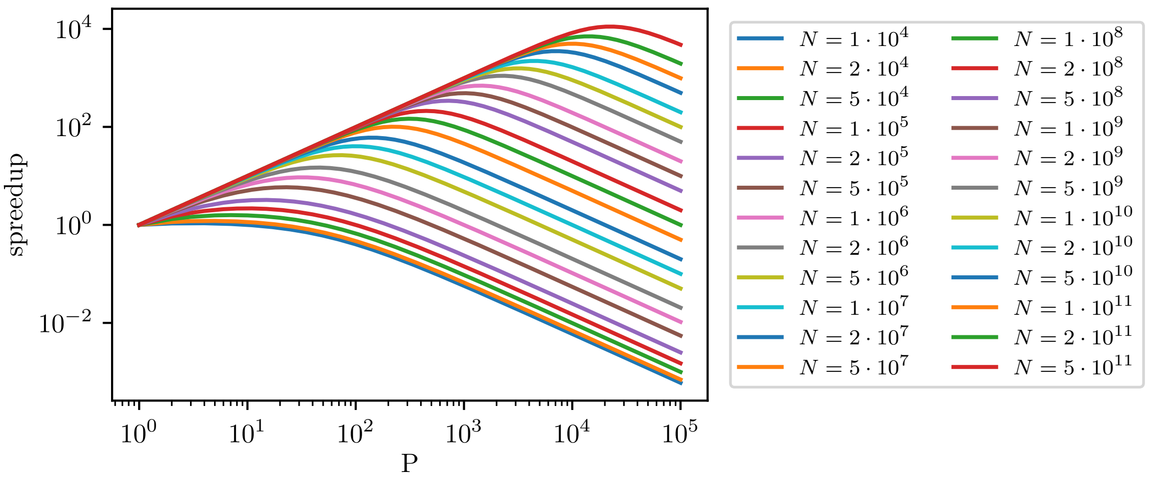

Figure 3: Speedup function computed for several values of N. Notice that this graph provide a nice way to verify the linearity relation (1.7). For instance, with $P \simeq 10^3$ cores, the graph shows us that we need at least $N \sim 5 \cdot 10^9$ to scale linearly.

1.2 Attempt of improvement

Perhaps it is possible to save some communication time. The ideas is basically to exploit binary tree collective operation. This will affect

- the communication part at the beginning using *scatter*-like operation, will reduce the initial communication overhead to $\lceil \log_2 P \rceil\, T_{\text{comm.}}$;
- the sum itself using a *reduce*-like operation. This will give a contribution of $\lceil \log_2 P \rceil\, T_{\text{comm.}}$ again for the communication part and $\lceil \log_2 P \rceil\, T_{\text{comp.}}$ for the computation part.

All in all the new time is

$$\tilde{T}(P,N) = T_{\text{read}} + 2\left\lceil \log_2(P) \right\rceil T_{\text{comm.}} + \left(\left\lceil \frac{N}{P} \right\rceil + \left\lceil \log_2(P) \right\rceil - 1 \right) T_{\text{comp.}} \, . \tag{1.8}$$

In this case it is easy to see that the minimum is attained at $P_{\text{min.}} \simeq N \frac{T_{\text{comp.}}}{2T_{\text{comm.}}+T_{\text{comp.}}} \log 2$. In Figure 4 a graph is plotted to compare the best value of P in function of N. Notice that since the overhead time does not scale linearly with P in this case, we have that $P_{\text{min.}}^{(\text{improved})} > P_{\text{min.}}^{(\text{naive})}$. In Figure 5 we see that the improved model perform better than the naive one expecially for higher values of N. However, as we noticed just before, the number of processors required to minimize the time, in the improved model in quite higher: with our numbers probably is much more convenient to wait $\sim 10^{-2}$ s more than to buy $\sim 10^5$ cores.

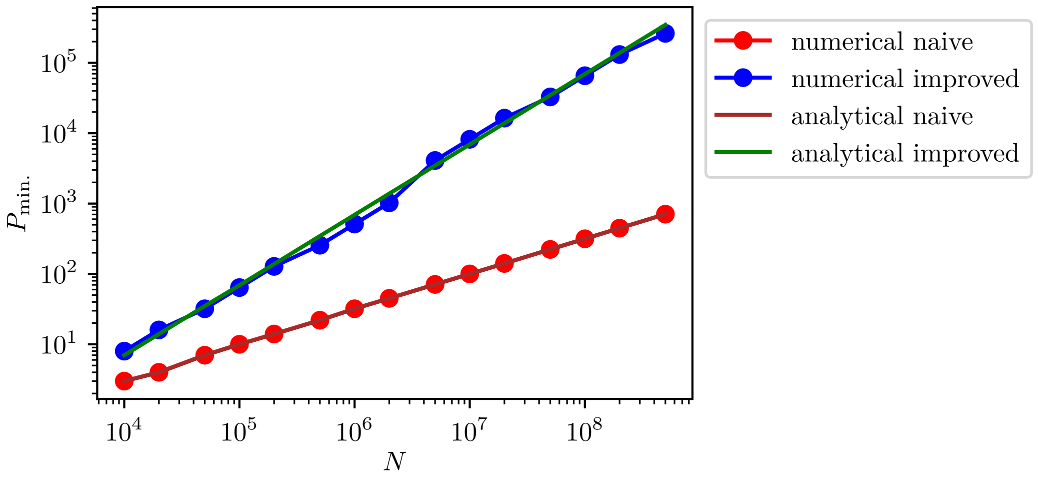

Figure 4: Comparison between naive and refined algorithm. Notice that, for the improved model the analytical and numerical point do not overlap perfectly; this is because of the upper integral part.

2 `MPI` analysis

All through this section we will measure time in two ways:

walltime : this is the time measured internally by the programs, both serial and parallel one. Of course every processor will have its own time. We will consider the maximum among all the processors.

elapsed : this is the time as measured by the command `/usr/bin/time`. The command provide use three times: `user`, `system` and `elapsed`. We will consider the last one.

On top of that all the result reported hereafter are the average of the various run and the error is the maximum deviation.

We we will deal with speedup we will use the definition

$$S(P,N) = \frac{T_{\text{serial}(N)}}{T_{\text{parallel}}(P,N)} \, , \tag{2.1}$$

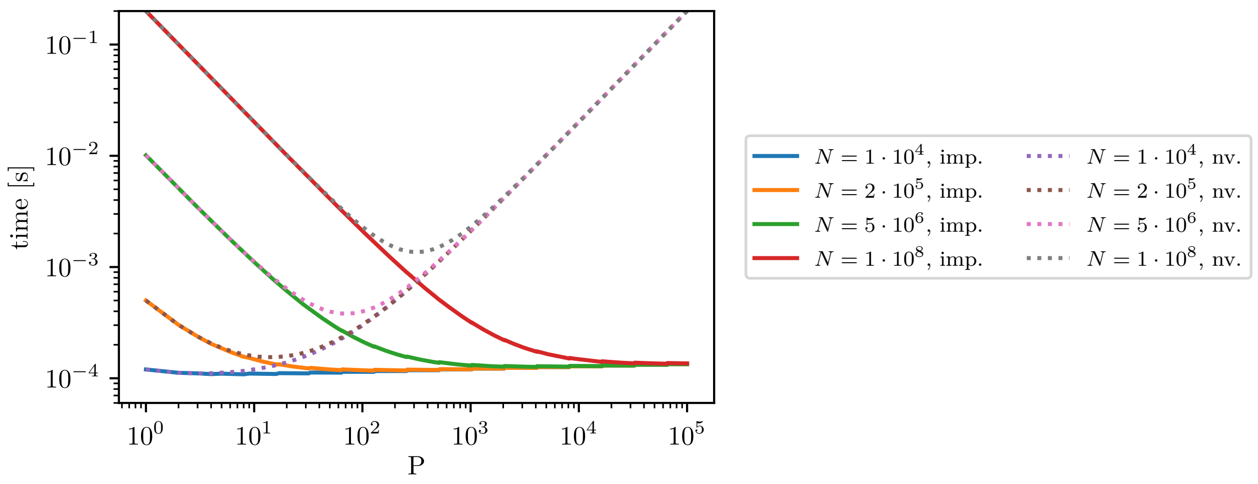

Figure 5: Performance of naive and improved model, compared

where for the serial time we used the serial launch for $N = 10^8, 10^0, 10^{10}, 10^{11}$. It is also interesting to compare the $T_{\text{serial}}(N)$ and $T_{\text{parallel}}(1, N)$: this is done in Table 2. Notice that either serial time is a little lower than the parallel time or they are compatible within their error. More details of this kind of analysis can be found in Subsection 2.2.

N	$T^{(\text{wall})}_{\text{serial}}(N)$ [s]		$T^{(\text{wall})}_{\text{parallel}}(1, N)$ [s]	
10^8	2.56		2.86	± 0.09
10^9	25.74	± 0.13	25.71	± 0.04
10^{10}	256.73	± 0.60	257.50	± 0.70
10^{11}	2601.56	± 86.55	2585.26	± 36.25

Table 2: Comparison between serial and parallel walltimes.

2.1 Strong

In the strong scaling experiments we run the parallel program for $N = 10^8, 10^9, 10^{10}, 10^{11}$ on $P = 1, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48$ cores, three times each.

- In Figure 6 we plot the absolute timing on semilog scale with error bars;
- in Figure 7 we plot the speedup in linear scale;
- in Figure 8 we plot the speedup comparing the various N on the same graph

We notice that while for lower N walltime and elapsed time have does not overlap (expecially for larger value of P), the situation is very different for higher values of N, where the two curves tend to coincide. This can be due to the fact that for low N times are very short and there can be some overhead in the execution of other function of the program (not the parallel overhead) which tends to disappear when we increase the number of iterations.

As far as the speedup is concerned we notice that again for lower values of N the walltime and elapsed time scalabilities tend to differ, while for higher values of N they tend to coincides. They also tend to coincide with the perfect scalability limit. This phenomenon is expected as of the theoretical model above; it is the benchmark of the the parallel overhead.

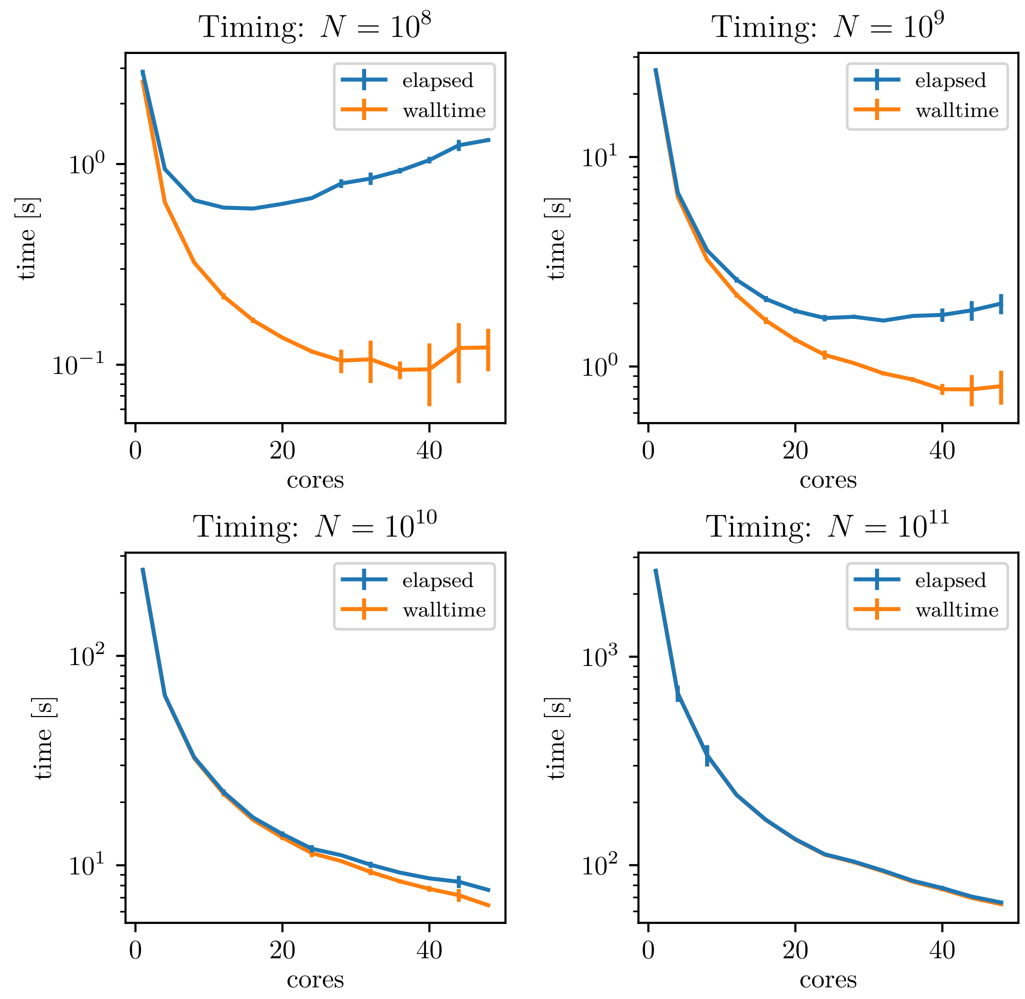

Figure 6: Absolute timing.

2.2 Parallel Overhead

From the teoretical model above we can identify the parallel overhead as

$$T_{\text{overhead}}(P, N) = P \cdot T_{\text{parallel}}(P, N) - P_{\text{serial}}(N) \; . \tag{2.2}$$

The results are plotted in Figure 9. We notice that the overhead grows as P grows, which is expected from the model.

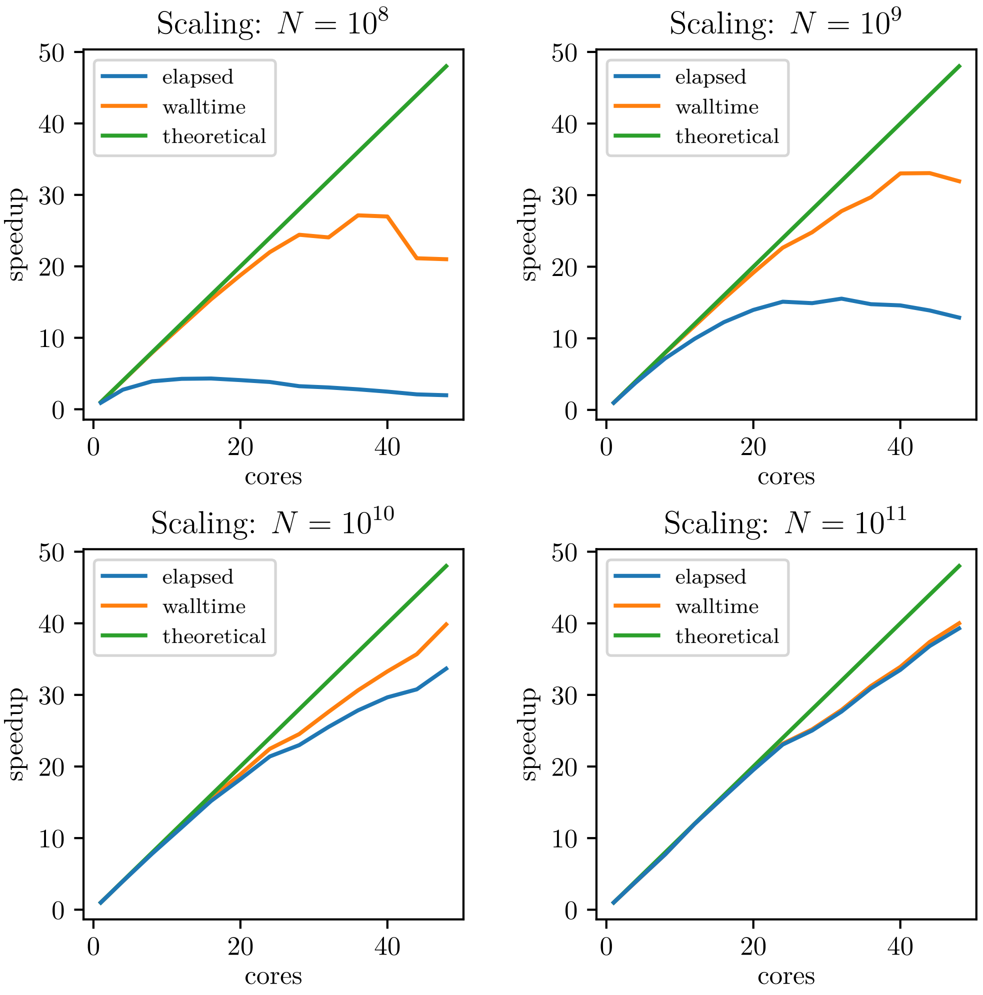

Figure 7: Speedup.

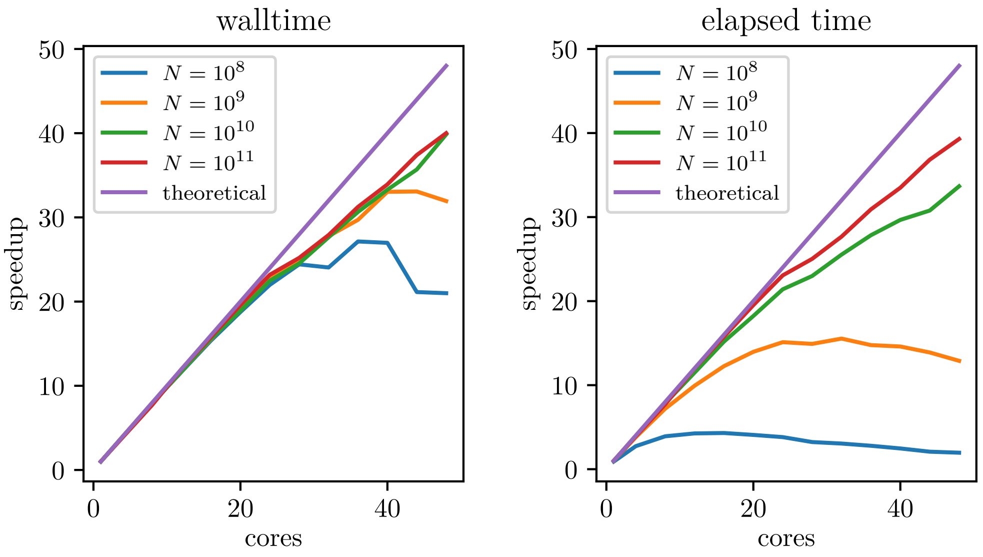

Figure 8: Speedup: summary.

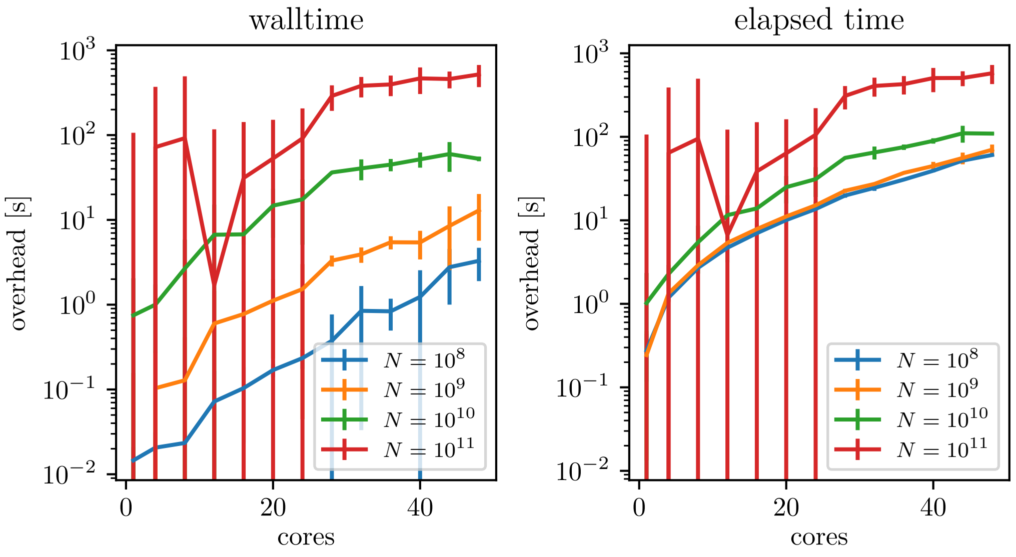

Figure 9: Overhead time. Notice that the curve for $N = 10^{11}$ is particularly affected by errors, especially for lower values of N.

We can also notice, especially from the walltime graph that the overhead depend also on N. We see, in the right part of the graph (the cleaner one) that $T_{\text{overhead}}(P, N) \sim N$ for a certain P. This points to the direction that, apart from communication time, there is a slowdown of performance when the parallel program is run.

2.3 Weak

In the weak scaling experiments we run the parallel program for $N = 10^8, 10^9, 10^{10}$ on $P = 1, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48$ cores and for $N = 10^{11}$ on $P = 1, 12, 24, 48$ cores, three times each.

- In Figure 11 we plot the absolute timing on semilog scale with error bars;
- in Figure 12 we plot the speedup in linear scale;
- in Figure 10 we plot the speedup comparing the various N on the same graph

As for the strong case we notice that for lower values of N the walltime and elapsed time are distinct; their difference tend to disappear for larger values of N. If scalability were perfect we would observe a flat graph. Instead we notice that, apart for a point at $N = 10^{10}$ and $P = 24$ the curves are monotonously increasing. This is the benchmark of the overhead in communication time. As a direct consequence the speedup curves are not constant and equal to 1.

Let us now try to explain that "spike", located at $N = 10^{10}$ and $P = 24$. During the experiment several `MPI` errors occurred. These error where spoiling the result and we were forced to run the script again. Usually I decided to make a new set of measurement. In that case that was not possible (the queue on the cluster was very long) and so I did only some measurement of $N = 10^{10}$ and $P = 24$. I noticed that this last measure was done on a certain node, while the rest on another node (both gpunodes). I do not know how much changing noted (they are supposed to be identical twin) can make the difference. This is however what I noticed.

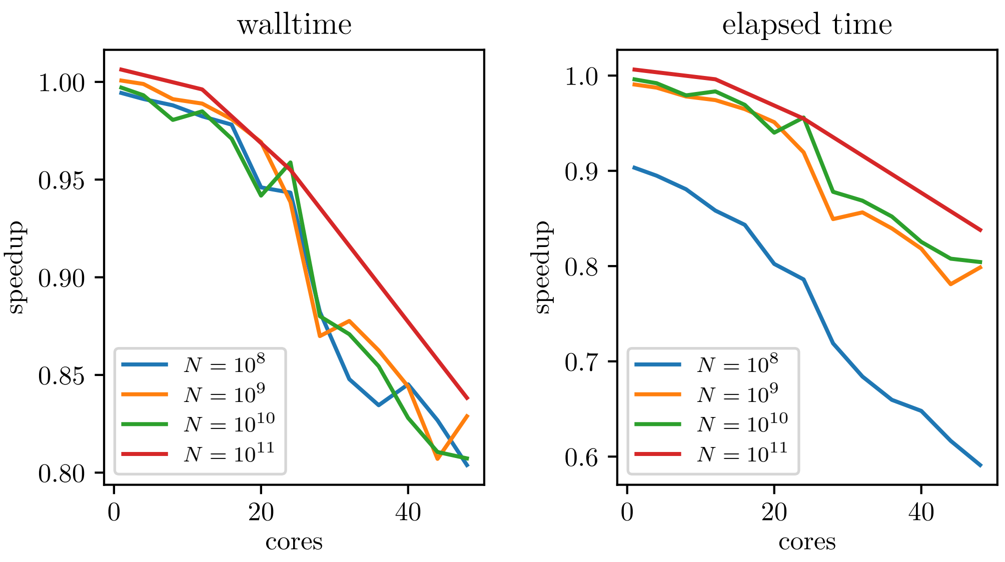

Figure 10: Speedup: summary.

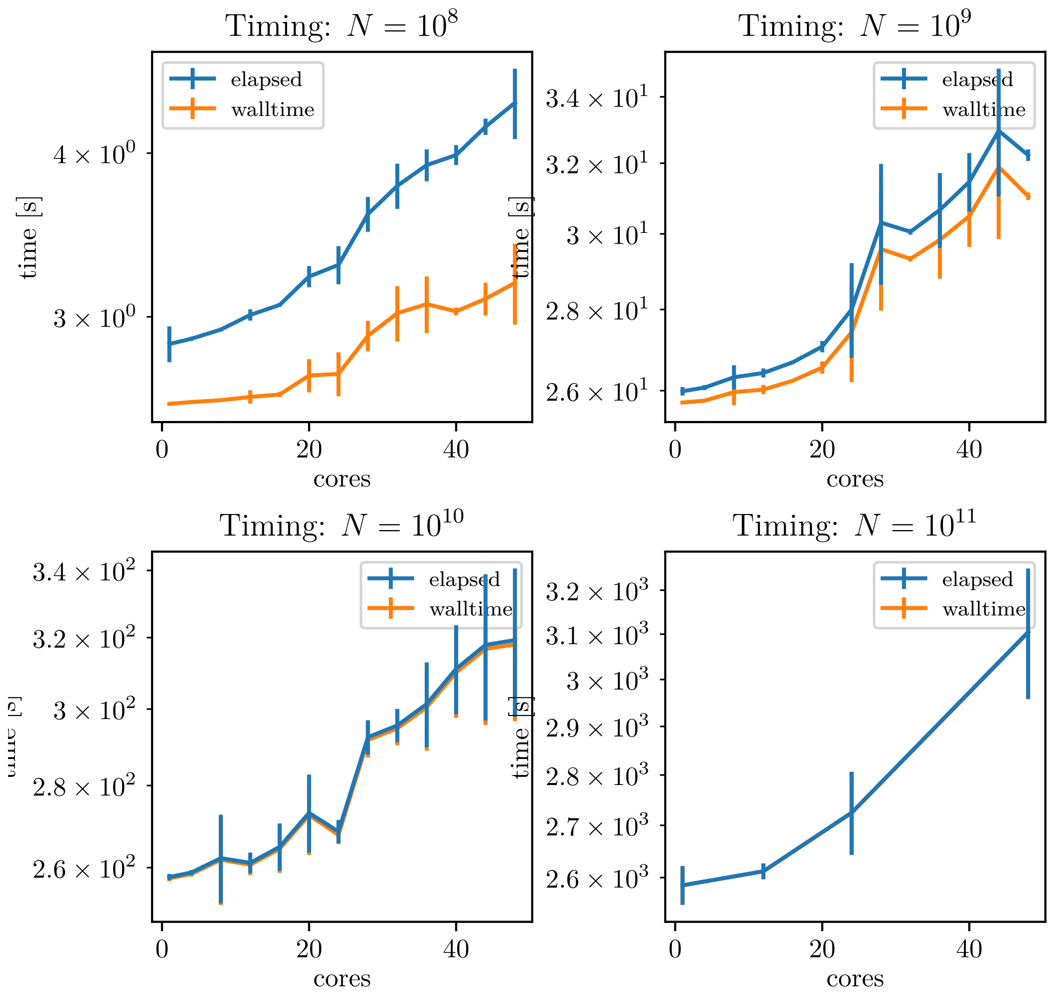

Figure 11: Absolute timing. Notice that there is a misbehaving for $N = 10^{10}$ and $P = 24$. In the other points the curves are monotonous within the errors.

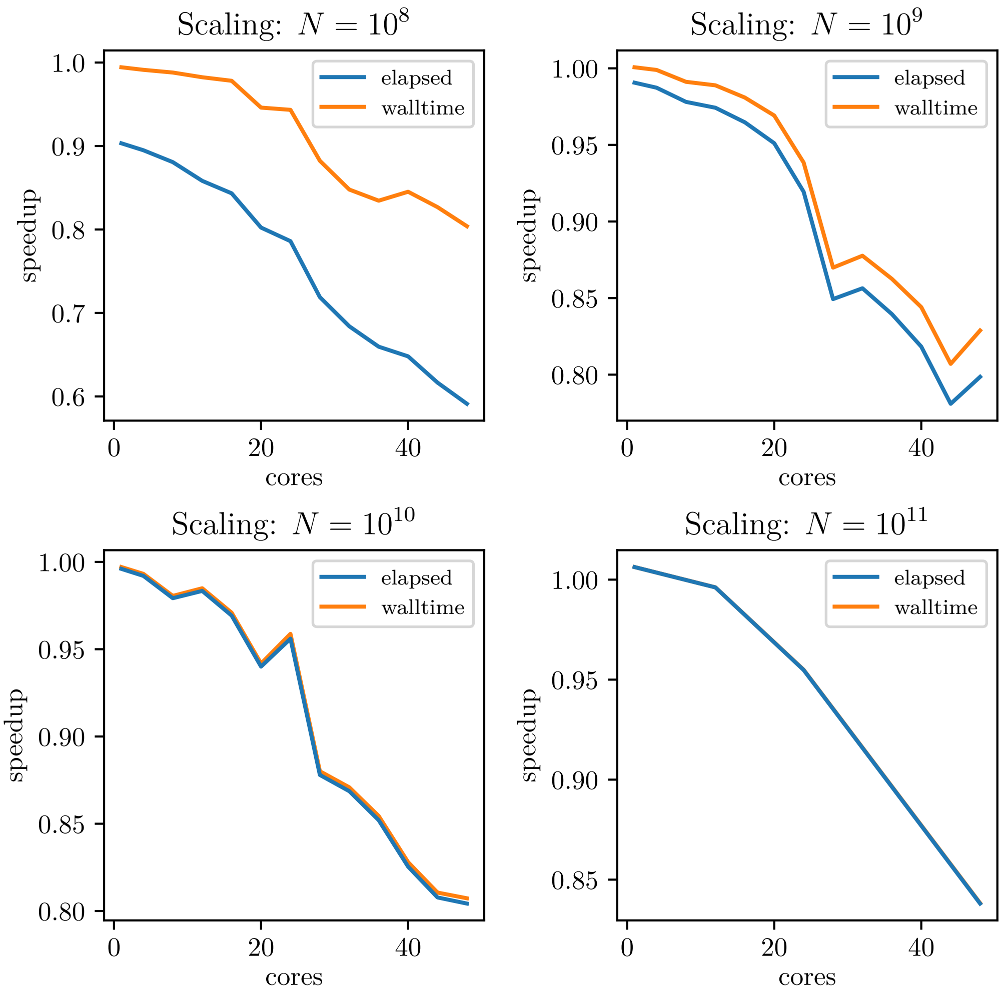

Figure 12: Speedup.