Scalability for realtime problems

Martin Schreiber

November 15, 2015

The performance of parallel programs and the feasibility to run efficiently on large-scale architectures is typically measured by weak and strong scalability plots. In this work, we briefly review these performance metrics, show significant disadvantages of these metrics and present a new way to measure scalability results for problems with (sub)realtime requirements.

Changelog

2023-02-26: Some cleanups

1 Introduction

The term scalability has its origins in the economics for describing how efficiently goods can be produced depending on the number of workers. It was used to measure how well the production of one product can be accelerated by increasing the number of workers.

Later, this term was adopted by the computer scientists to describe how well a problem (production outcome) can be solved for an increasing number of cores (workers). Two different laws were developed: The one by Amdahl which basically describes a strong scaling problem and the other one by Gustafson which describes a weak scaling problem. Both problem descriptions are nowadays used to describe the scalability of programs and they only take the workload per computing core into account. Both laws were only based on purely theoretical parts without consideration of the underlying problem. They in particular only distinguished between serial and parallelizable parts in the way how an algorithm can be parallelized.

In this work, we focus on simulations executed on an arbitrary grid. With the desire of running higher resolution simulations being one of the main driving force for the advance in super computers, this could be directly represented as a weak scalability problem: For an increase of resolution, the workload is kept constant at each core and the number of cores are increased. However, this does not account for real time requirements, since an increase of resolution typically also leads to an increase of required time step sizes. Therefore, the given problem

has to be able to scale on a larger number of cores which, again, is related to a strong scaling problem.

This work is on showing the relation between weak, strong and realtime scalability with the latter one introduced in this work.

2 Symbols

2.1 Weak and strong scaling:

Symbol	Description
C	Number of cores
C_B	Number of cores used as scalability baseline, typicall $C_B = 1$
$W_{\text{Total}}(N)$	Total workload per time step
$W_{\text{Core}}(N, c)$	Workload per core using c cores in total
N	Value describing problem size (e.g. $N^{\mathbf{2}}$ for 2D or $N^{\mathbf{3}}$ for 3D simulation)
$T(w, c)$	Wallclock time to solve a problem of total workload w on c cores
$S_W(c)$	Weak scalability on c cores
$S_S(c)$	Strong scalability on c cores

2.2 Realtime scaling

Symbol	Description
$dt(N)$	Timestep size depending on problem size (e.g., resolution)
$W_{\text{TotalSim}}(N)$	Total workload for entire simulation depending on problem size (e.g., resolution)
D	Dimension of problem

3 Weak and strong scalability

We first formalize weak and strong scalability as a staring point for our ongoing discussions. This scalability is based on a wallclock time given for a baseline of cores which we denote with the subscript B as C_B.

The wallclock time for an execution of a certain total baseline workload W_B on C_B cores (typically $C_B = 1$) is then given as

$$T_B := T(w = W_B, c = C_B).$$

3.1 Weak scalability

The weak scalability assumes a fixed workload per core, hence with the per-core workload

$$W^{\text{Weak}}_{\text{Core}}(N, c) := W_B$$

and total workload

$$W^{\text{Weak}}_{\text{Total}}(N, c) := c \cdot W_B$$

with c the number of cores. Here, we can either interpret it as fixing the workload per core or a proportional increase of the workload to the number of cores.

Its speedup is then defined as

$$S_{\text{Weak}}(c) := \frac{T\left(W_B \cdot \frac{c}{C_B}, C_B\right)}{T_B}$$

3.2 Strong scalability

The strong scalability is using a fixed total workload

$$W^{\text{Strong}}_{\text{Total}}(N, C_B) := W_B$$

which is distributed across all cores

$$W^{\text{Strong}}_{\text{Core}}(N, C_B) := \frac{W_B}{C_B}.$$

This results in a linear decrease of the workload per core for increasing the number of cores.

Its speedup is then defined as

$$S_{\text{Strong}}(c) := \frac{T\left(W_B, c\right)}{T_B}.$$

Obviously, achieving a strong scalability is much harder to achieve in this context.

4 Realtime scalability

It is obvious that the above mentioned scalability measurements are not related at all to the absolute wallclock time of executing a hyperbolic simulation. In this work, we focus on simulations of hyperbolic character. Here, we know that due to stability reasons, the time time step size is indirectly proportional to the resolution (at least for 1st order hyperbolic problems)

$$\Delta t(N) = C\frac{1}{N}$$

where C is some constant and N related to the resolution. Therefore, the higher the resolution, the more time steps need to be computed. We tabulate the increase in total workload increase depending on the resolution:

Dimensions	**Workload $W_{\textbf{TotalSim}}(N)$**
1	$W_{\text{TotalSim}}(N) = N\,N$
2	$W_{\text{TotalSim}}(N) = N^2\,N$
3	$W_{\text{TotalSim}}(N) = N^3\,N$
D	$W_{\text{TotalSim}}(N) = N^D\,N$
0	$W_{\text{TotalSim}}(N) = N$

Here, the total simulation workload $W_{\text{TotalSim}}(N)$ gives the workload over all time steps for a particular resolution N. The N^D factor accounts for the increase in resolution and the last factor N for the increase in number of time steps.

We can now set an overall wallclock time limitation $T(N, c)$ for the computations on a single core as before for a given workload (depending on the resolution) and number of cores:

$$T(N, C) := \frac{W_{\text{TotalSim}}(N)}{C} = \frac{W_{\text{TotalSim}}(N_B)\, N^{D+1}}{C}.$$

Next, we start developing the realtime performance model by introducing C_R as the cores considered for realtime scalability and N_R as the resolution-related parameter. First, since the per-core workload has to be the same for realtime reasons, we get

$$\begin{aligned} T(N_R, C_R) &= T(N_B, C_B) \\ \frac{W_{\text{TotalSim}}(N_B)\, N_R^{D+1}}{C_R} &= \frac{W_{\text{TotalSim}}(N_B)\, N_B^{D+1}}{C_B} \\ \frac{N_R^{D+1}}{C_R} &= \frac{N_B^{D+1}}{C_B}. \end{aligned}$$

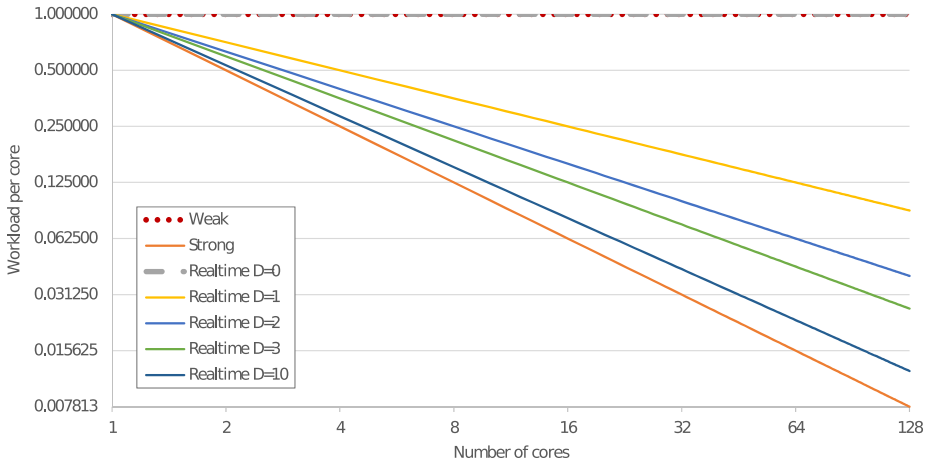
Hence, for a given resolution N, the number of cores are given by

$$C_R := \frac{N_R^{D+1}\, C_B}{N_B^{D+1}} = N_R^{D+1} \frac{C_B}{N_B^{D+1}}$$

or based on the number of cores C_R, the resolution N_R is given by

$$N_R := N_B \left(\frac{C_R}{C_B}\right)^{(D+1)^{-1}}.$$

This results in the following workload distribution for weak, strong and realtime scaling with the last one depending on the dimension of the underlying problem. Both axes are given in log-scaling.



Here, we can observe that the workload per core results (as well known) is constant for weak scaling and decreases reciprocal to the number of cores for strong scaling. Regarding the realtime scaling, we can observe, that for D=0 such as it is the case for an ODE, this does not change anything since the time step size is independent of the resolution. For D=1, the workload per core is by far not as much decreasing as it is the case for the strong scaling. For problems with a higher dimension, hence $D \to \infty$, the workload is not as fast decreasing as typically expected for the strong scaling.

Therefore, the real circumstance to which the HPC community for scalability is exposed to is not a strong scalability issue, but a mix between strong and weak scalability. All these plots show only the optimal scalability for the considered problem size.

5 Realtime scalability with Amdahl and Gustafson

[TODO] show relationship to Amdahl and Gustafson