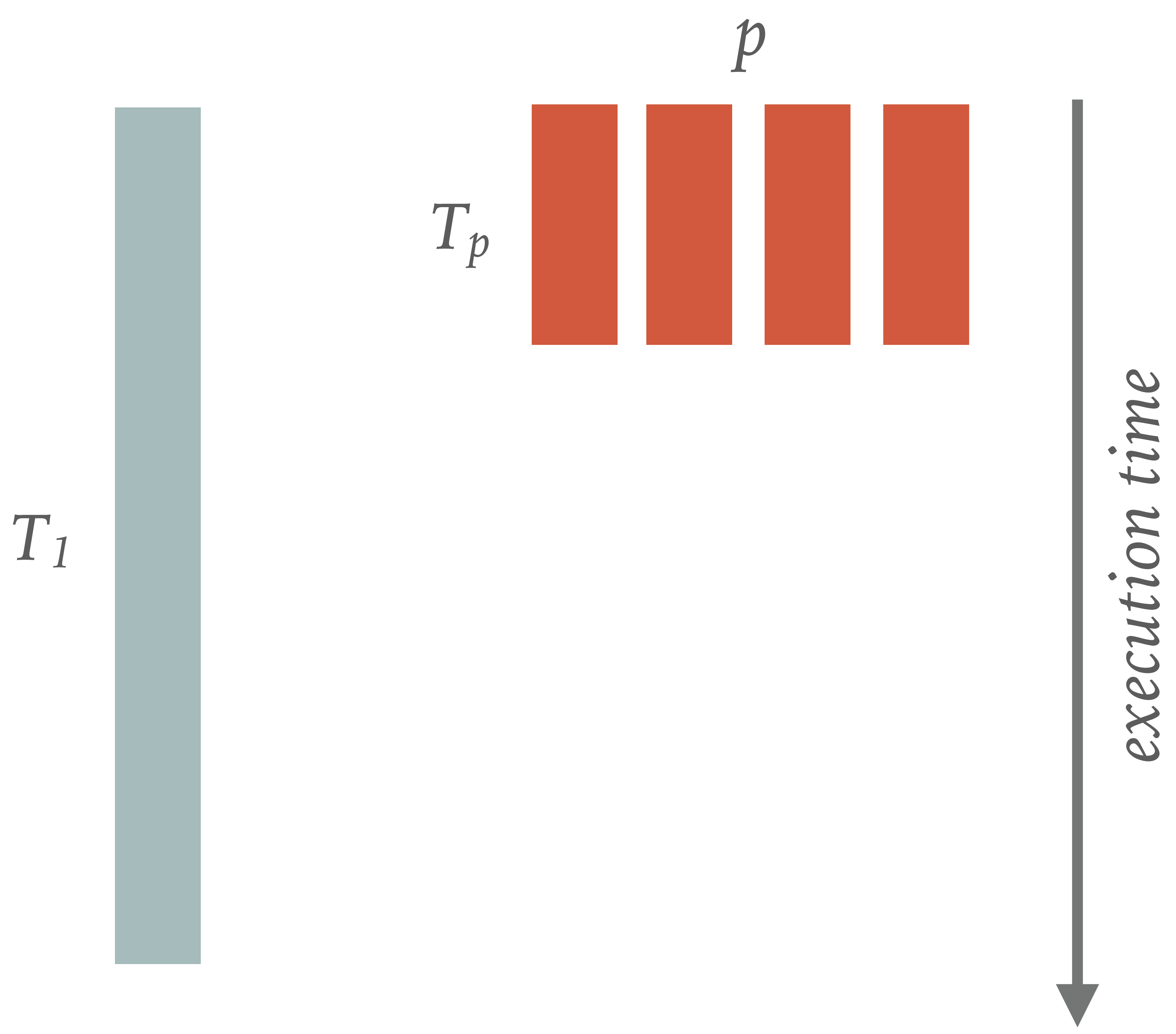
HOW TO MEASURE PERFORMANCE OF A PARALLEL CODE?

T_1 - *execution time of a serial algorithm*

p - *number of processes used*

T_p - *execution time of a parallel algorithm on p processes*

$S_p = \frac{T_1}{T_p}$ - ***speedup*** *of a parallel algorithm using p processes*

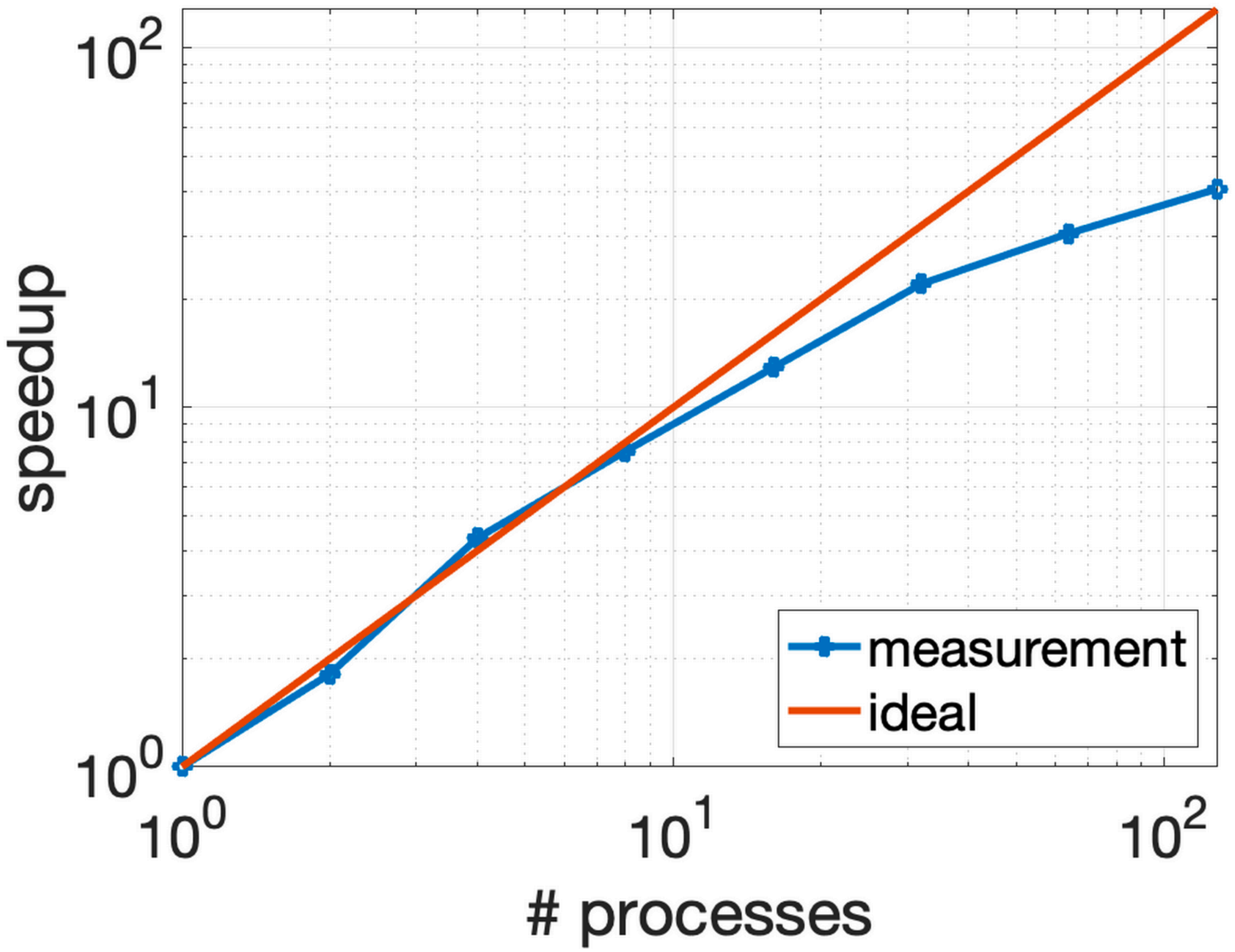


HOW TO MEASURE PERFORMANCE OF A PARALLEL CODE?

$$S_p = \frac{T_1}{T_p}$$

Ideally, $T_p = \dfrac{T_1}{p}$

which leads to a ***linear speedup****:* $S_p = p$



AMDAHL'S LAW

What if not all the code can be parallelized?

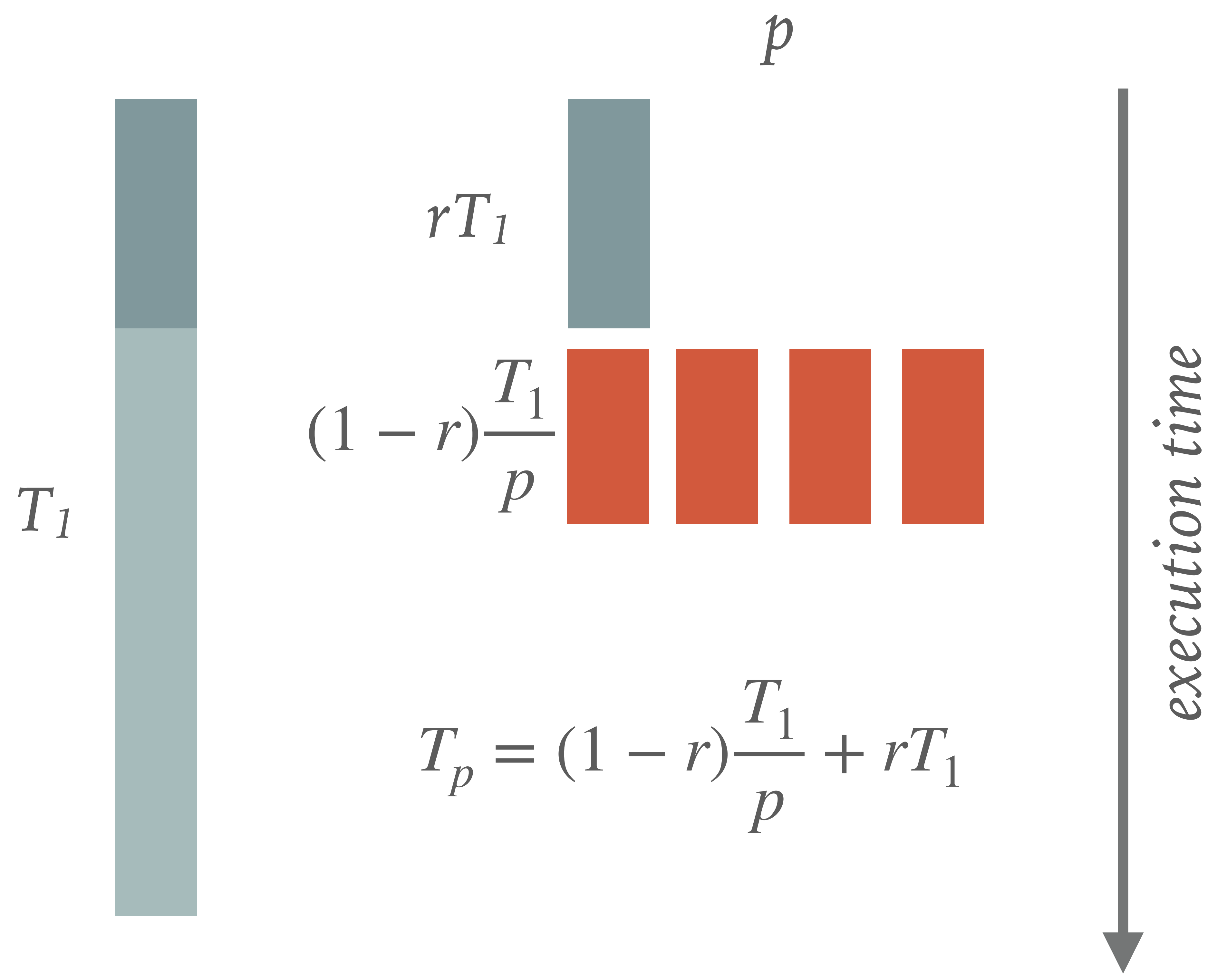
r - part of the code which remains serial

In that case, speedup looks as follows:

$$S_p = \frac{T_1}{T_p} = \frac{T_1}{(1-r)\frac{T_1}{p} + rT_1}$$

So even for a very large p we get:

$$\lim_{p\to\infty} S_p = \lim_{p\to\infty} \frac{T_1}{(1-r)\frac{T_1}{p} + rT_1} = \frac{1}{r}$$



AMDAHL'S LAW

But we did not even account for communication overhead…

Speedup then becomes:

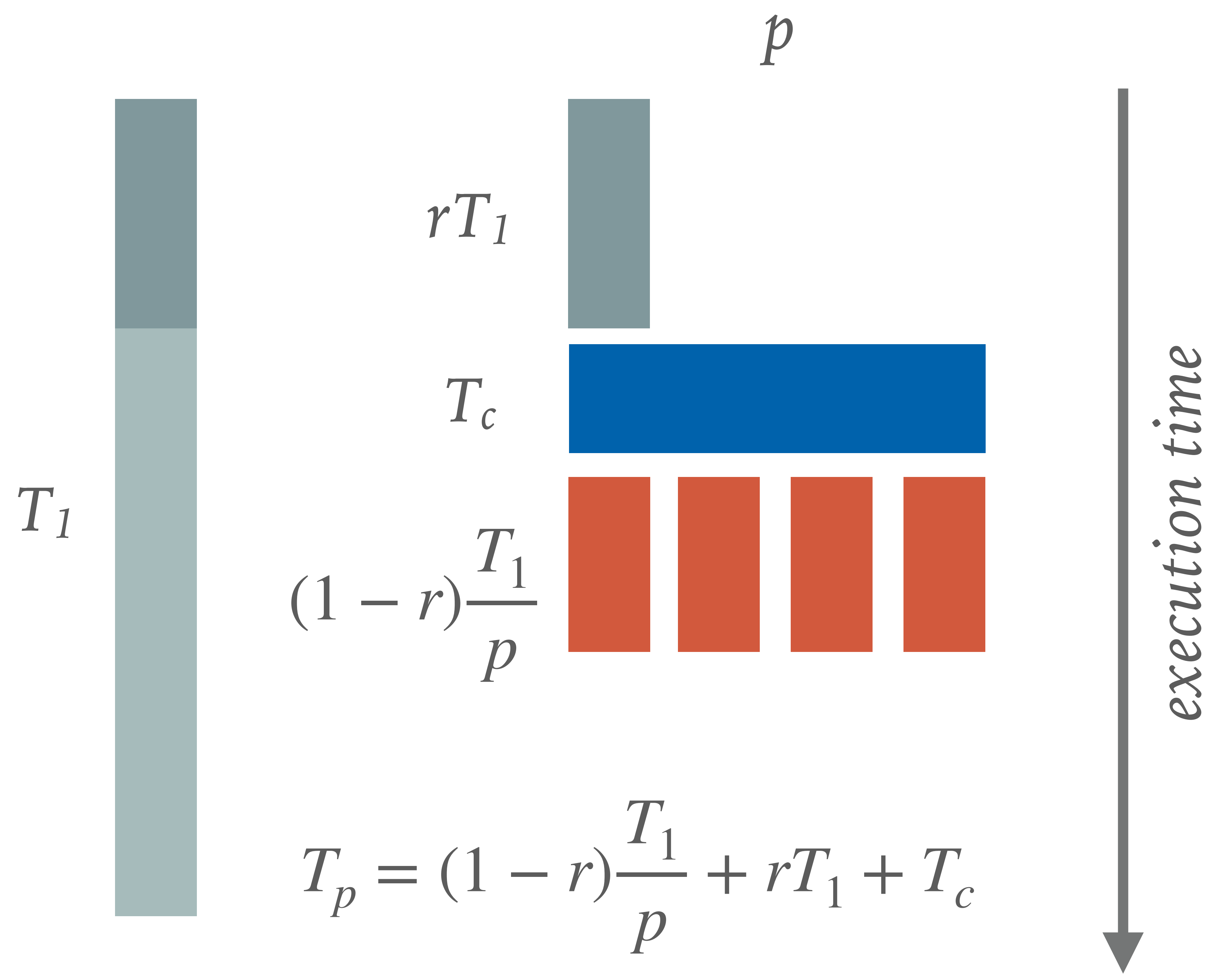
$$S_p = \frac{T_1}{(1-r)\frac{T_1}{p} + rT_1 + T_c}$$

Assuming perfectly parallel program (r=0):

$$S_p = \frac{T_1}{\frac{T_1}{p} + T_c}$$

For scalability S_p to be close to p (linear), we require:

$$T_c \ll \frac{T_1}{p} \qquad \text{or} \qquad p \ll \frac{T_1}{T_c}$$



SCALABILITY

Scalability - ability to maintain efficiency with increasing process count

Strong scaling:

increasing number of processes does not significantly decrease efficiency for a constant problem size

__How fast__ can I run a problem of constant size given more computational resources?

Weak scaling:

increasing number of processes does not significantly decrease efficiency when we increase problem size commensurately to the number of processes

__How big__ a problem can I run given more computational resources?

SCALABILITY

Strong scaling:

increasing number of processes does not significantly decrease efficiency for a constant problem size

nproc	N	time
1	8000	10 s
2	8000	5 s
4	8000	2.5 s
8	8000	1.25 s

Weak scaling:

increasing number of processes does not significantly decrease efficiency when we increase problem size commensurately to the number of processes

nproc	N	time
1	1000	1.25 s
2	2000	1.25 s
4	4000	1.25 s
8	8000	1.25 s

EFFICIENCY

Strong scaling:

increasing number of processes does not significantly decrease efficiency for a constant problem size

Weak scaling:

increasing number of processes does not significantly decrease efficiency when we increase problem size commensurately to the number of processes

Efficiency:

Work per process decreases commensurately with number of processes

$$E_p = \frac{S_p}{p} = \frac{T_1}{pT_p}$$

Efficiency:

Work per process remains constant regardless of the number of processes.

$$E_p = \frac{T_1}{T_p}$$

SCALABILITY

Strong scaling:

increasing number of processes does not significantly decrease efficiency for a constant problem size

Weak scaling:

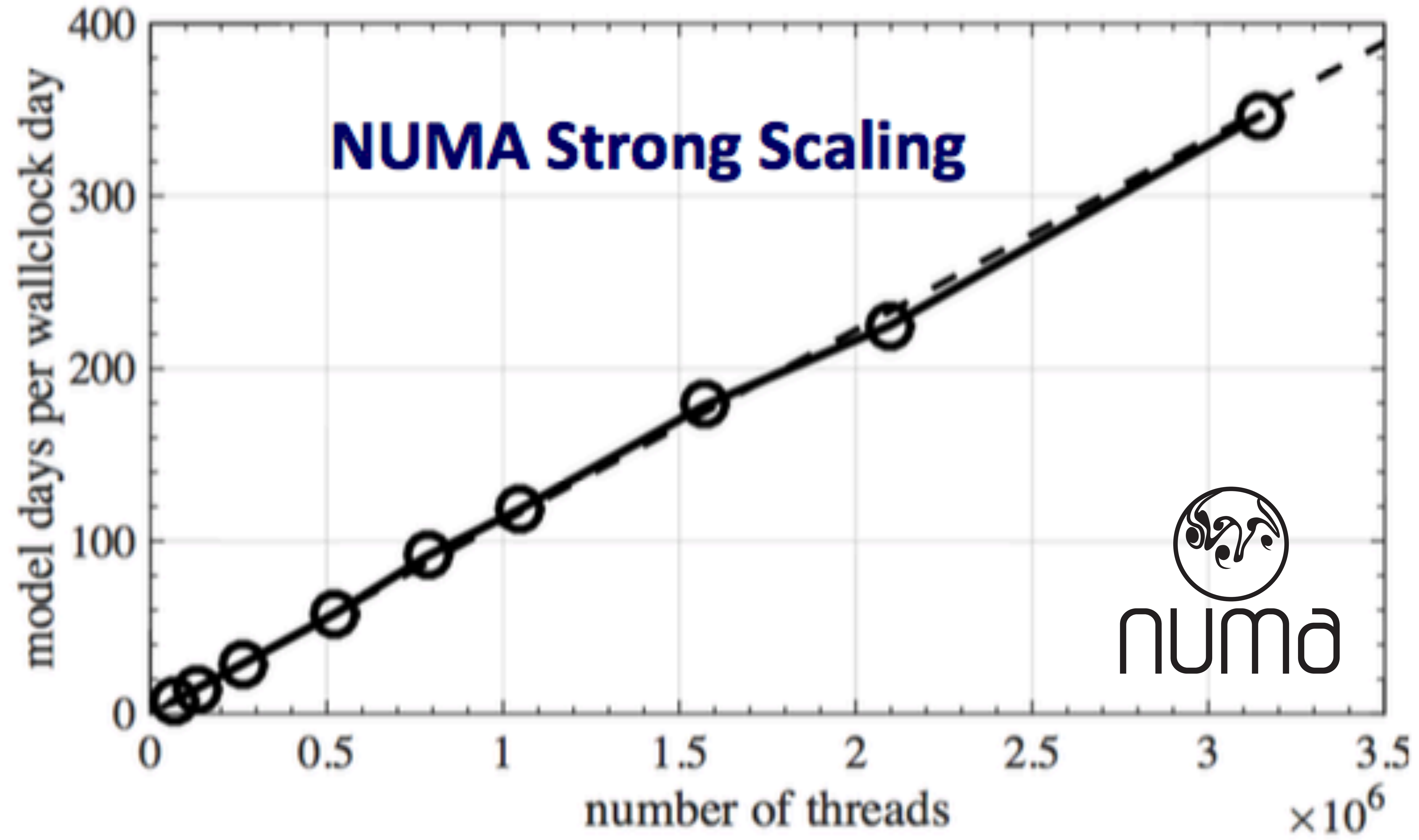
increasing number of processes does not significantly decrease efficiency when we increase problem size commensurately to the number of processes

		problem size			
		1000	2000	4000	8000
nproc	1	1.25	2.5	5	10
	2	0.625	1.25	2.5	5
	4	0.312	0.625	1.25	2.5
	8	0.156	0.312	0.625	1.25

Strong scaling

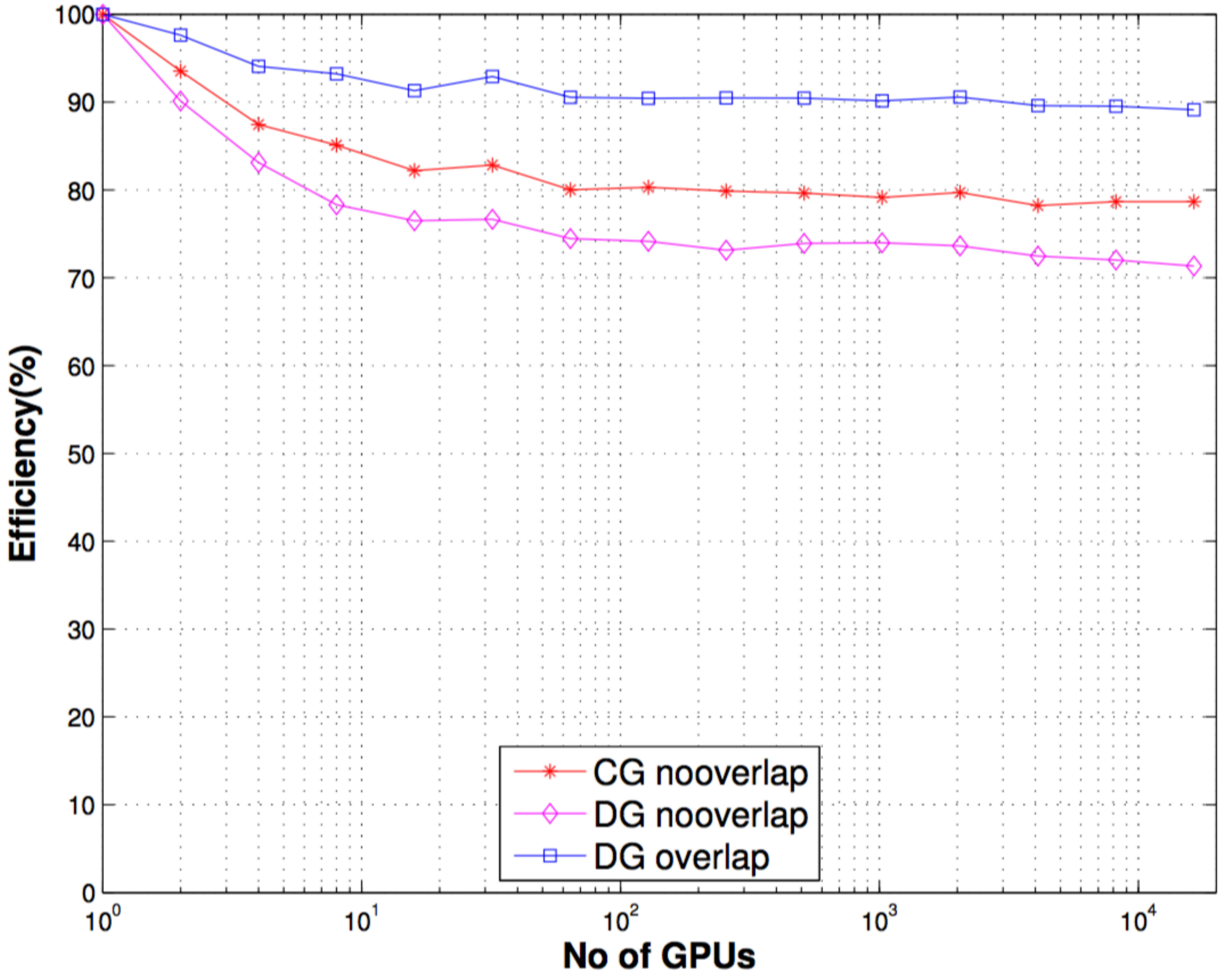
Weak scaling

SO, ARE WE DOOMED TO FAIL…?



Müller, A., Kopera, M. A., Marras, S., Wilcox, L. C., Isaac, T., & Giraldo, F. X. (2015). Strong scaling for numerical weather prediction at petascale with the atmospheric model NUMA. *The International Journal of High Performance Computing Applications*.

SO, ARE WE DOOMED TO FAIL…?



Abdi, D. S., Wilcox, L. C., Warburton, T. C., & Giraldo, F. X. (2019). A GPU-accelerated continuous and discontinuous Galerkin non-hydrostatic atmospheric model. *The International Journal of High Performance Computing Applications*, *33*(1), 81-109.