Scalability:
Strong and weak scaling
-
Geray Karademir

USM Code Coffee
05.05.2025

Intro

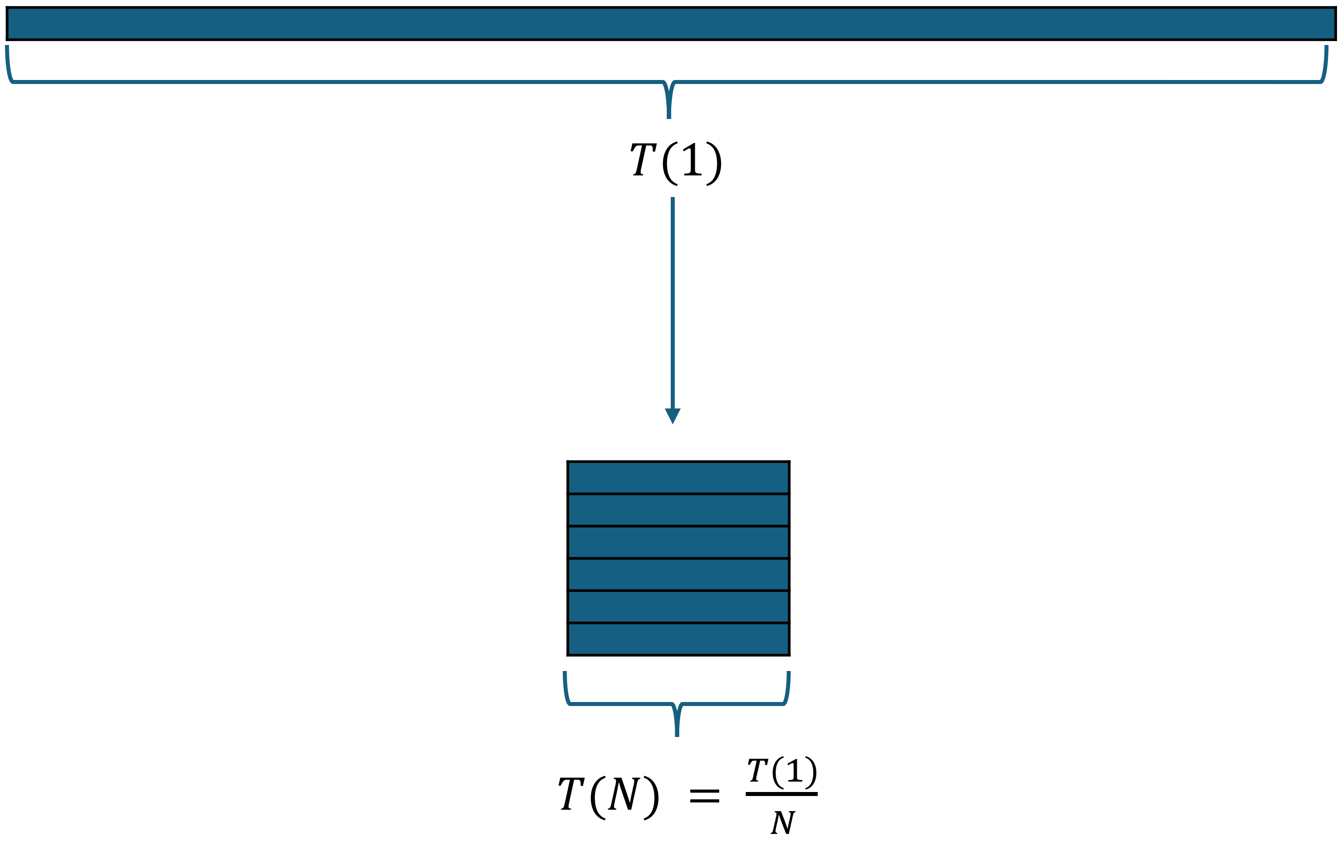
- Parallelization is key to solve large problems in a reasonable amount of time and the USM/LMU offers substantial computational resources to develop, test and run code up to small numbers of nodes.

- How do you verify if your code is capable of using these resources?

- What's the best number of resources given my code and the problem size?

Idealised view

- Running on one worker:
- Running on N workers:

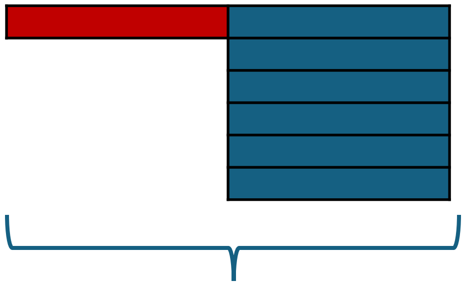


Idealised view

- Running on one worker:

$$T(1) = s + p$$

- Running on N workers:



$$T(N) = s + \frac{p}{N}$$

Scalability metrics

How much faster do you become when using N workers?

→ Speedup: $S(N) = \frac{T(1)}{T(N)}$

How efficient do these N workers perform?

→ Effiency: $\varepsilon(\mathrm{N}) = \frac{S(N)}{N}$

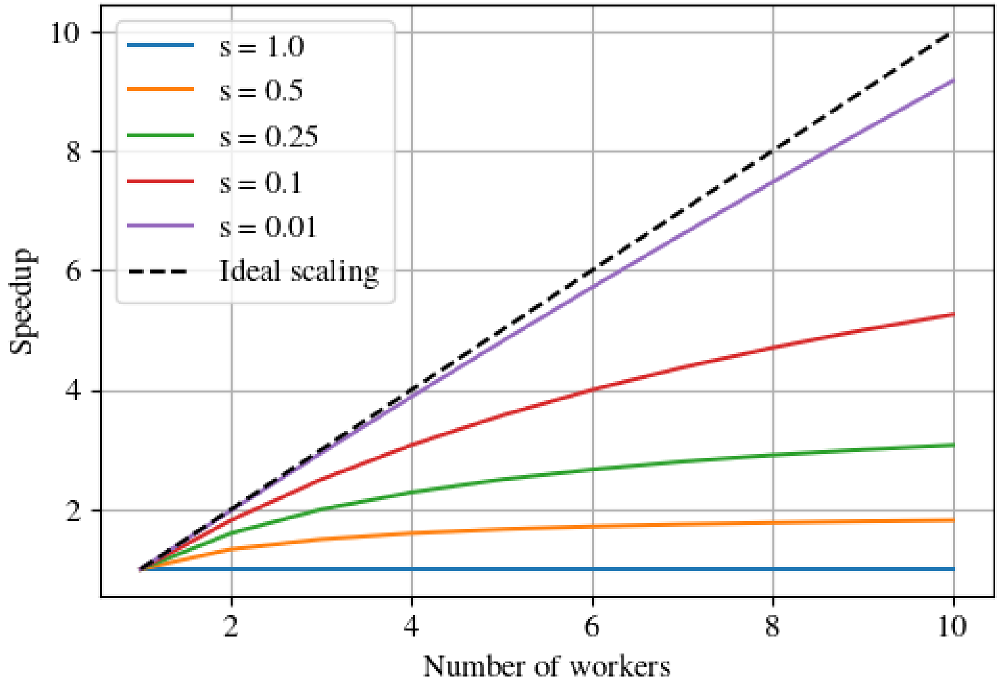
Warning! These don't tell you a lot about your performance! Bad code can scale very well!

Amdahl's Law (1967) –"Strong Scaling"

Amdahl's Law demonstrates the theoretical maximum speedup of an overall system assuming a fixed workload.

$$S(N) = \frac{T(1)}{T(N)} = \frac{1}{s + \frac{1-s}{N}}$$

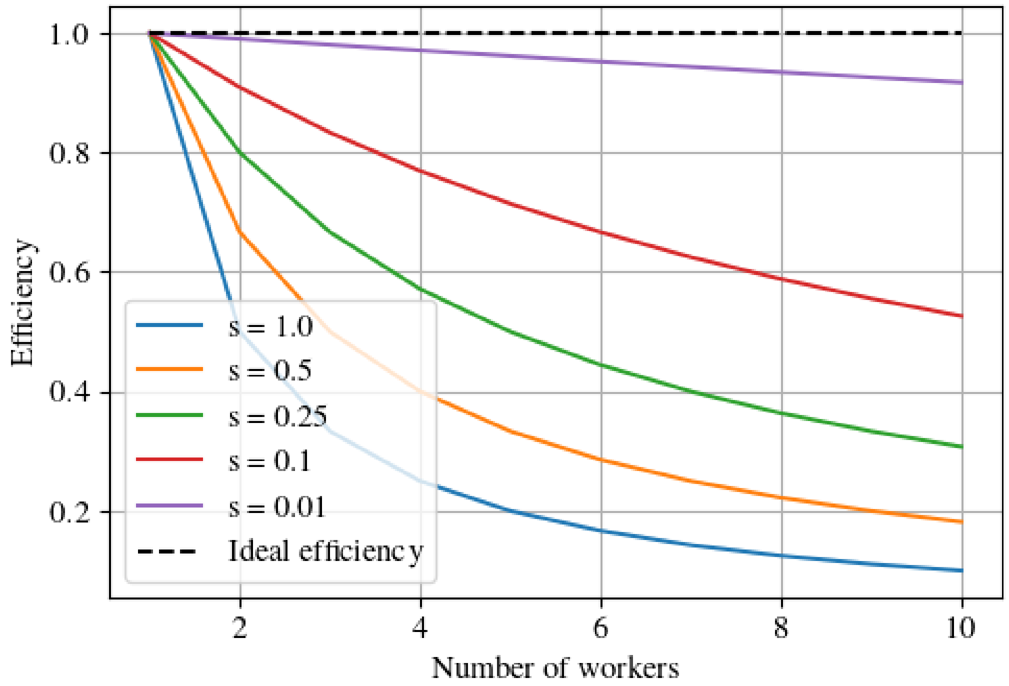
➔The asymptotic limit is the serial part.



Amdahl’s Law (1967) –“Strong Scaling”

Efficiency:

$$\varepsilon(N) = \frac{T(1)*N_1}{T(N_i)*N_i} = \frac{1}{s(N-1)+1}$$

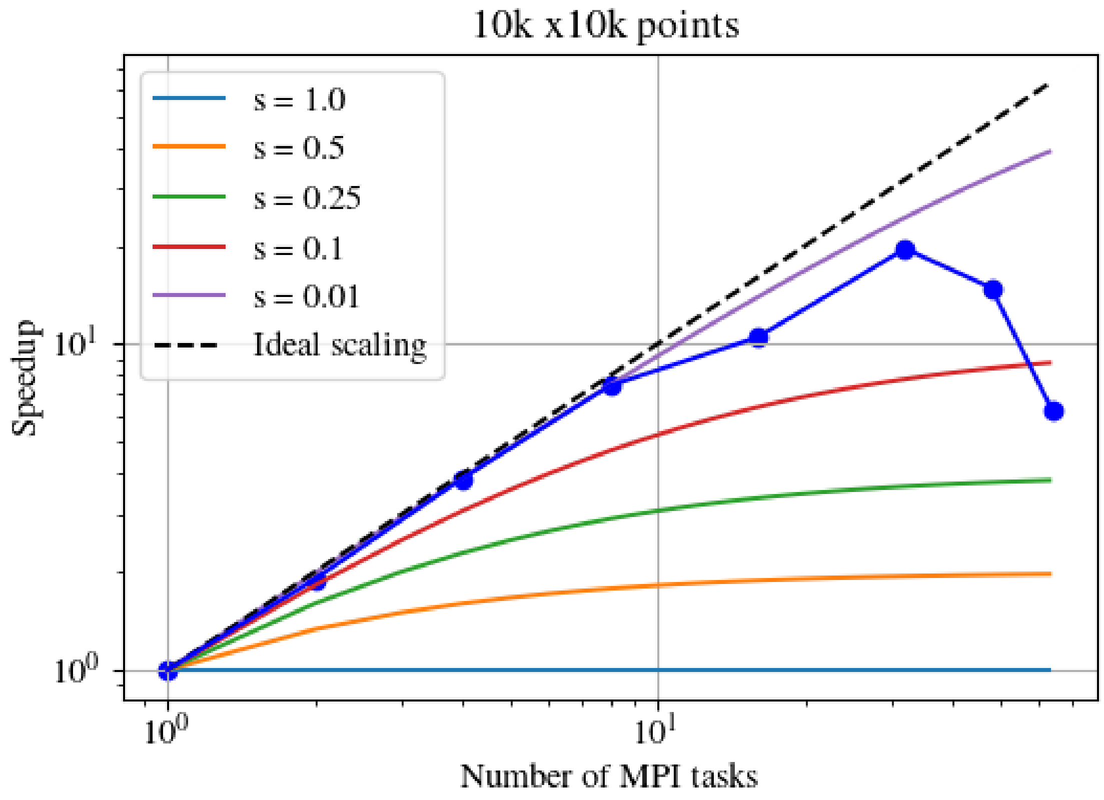


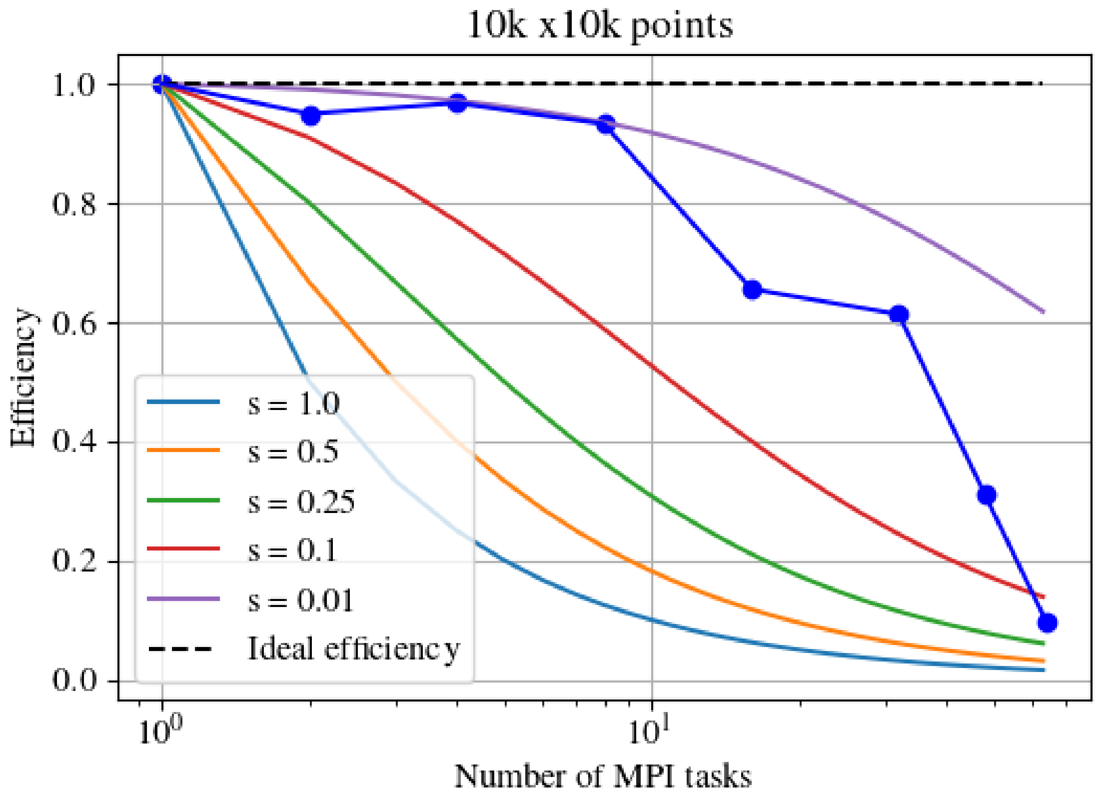
➔We ignore synchronization time, communication overhead, dependencies, wait times, sync points and resource bottlenecks.

Example Code:

- MPI parallelised code which calculates the distances of one dataset with all points of another and creates a distance distribution.
- Similar to calculating angular cross/autocorrelation functions.
- Lets have a look at the code!

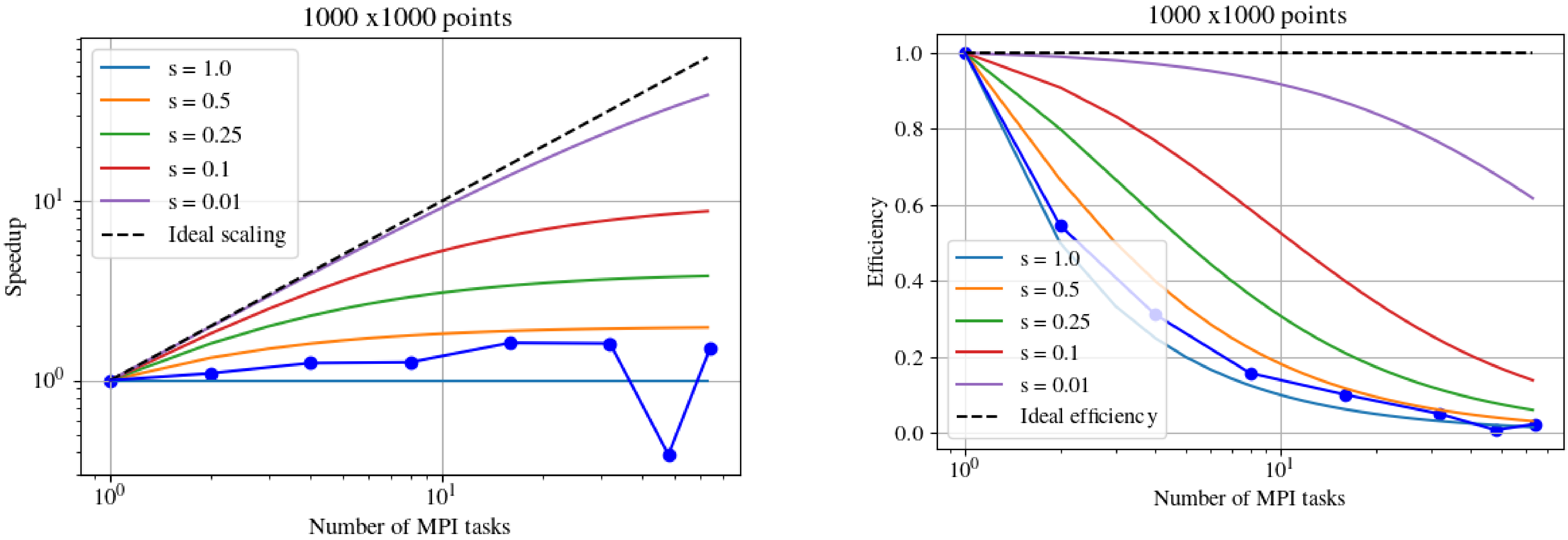
Example code



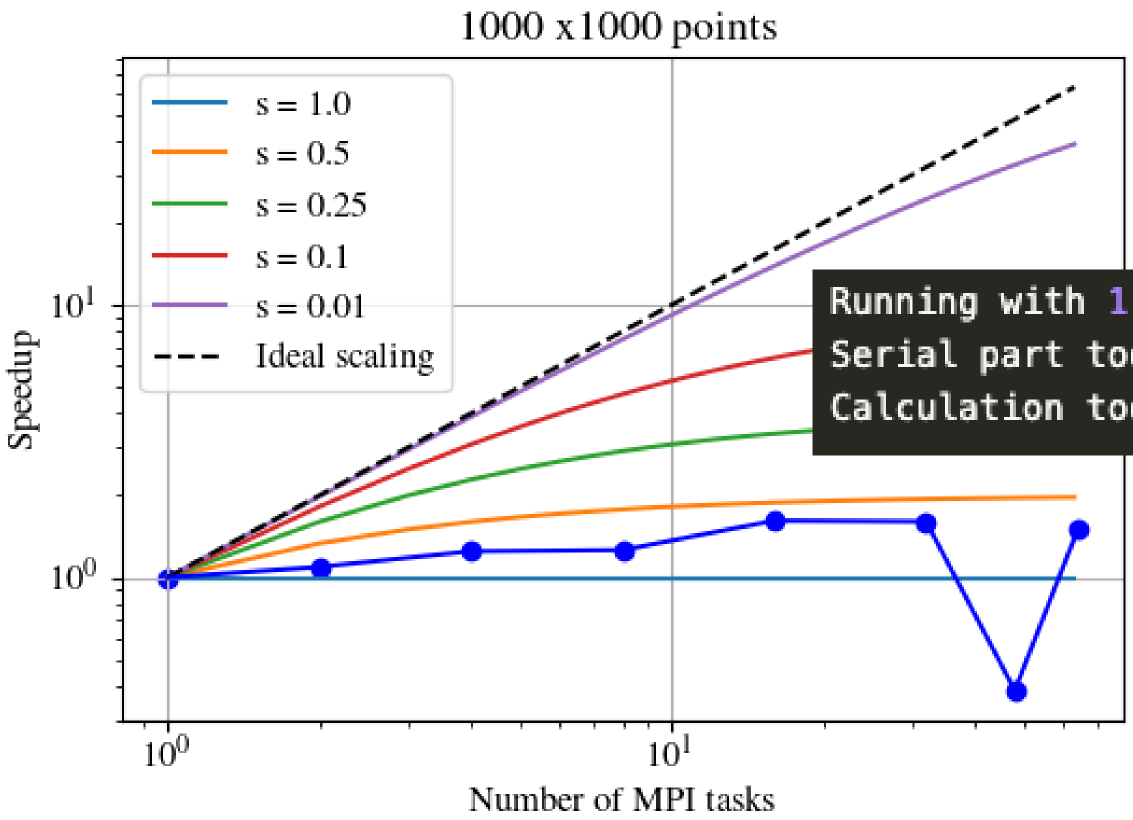


But what if I want to solve problems with different sizes?

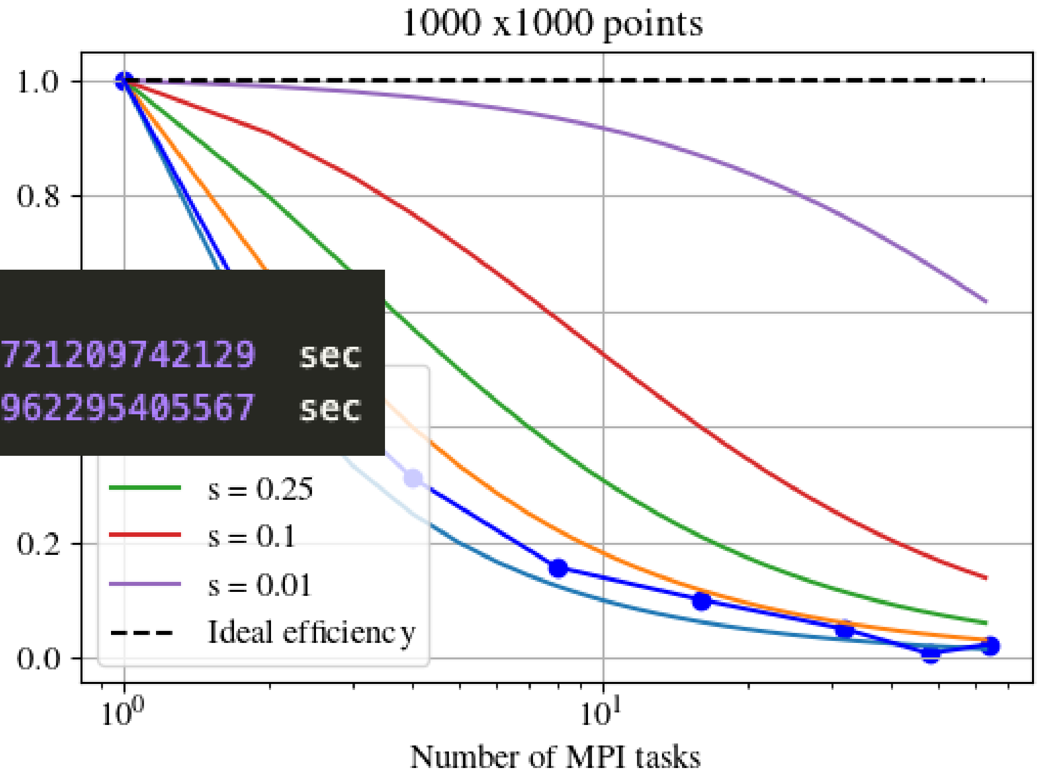
Example code



Example code



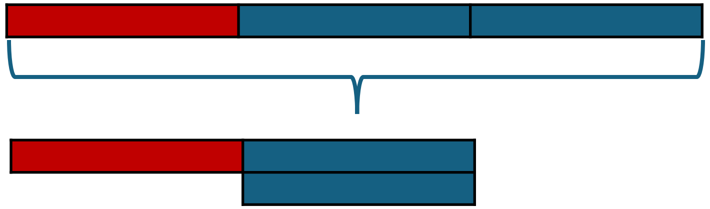

```
Running with 1 Processes
Serial part took 0.11114721209742129  sec
Calculation took 0.18124962295405567  sec
```



Gustafson's Law (1988) – "Weak scaling"

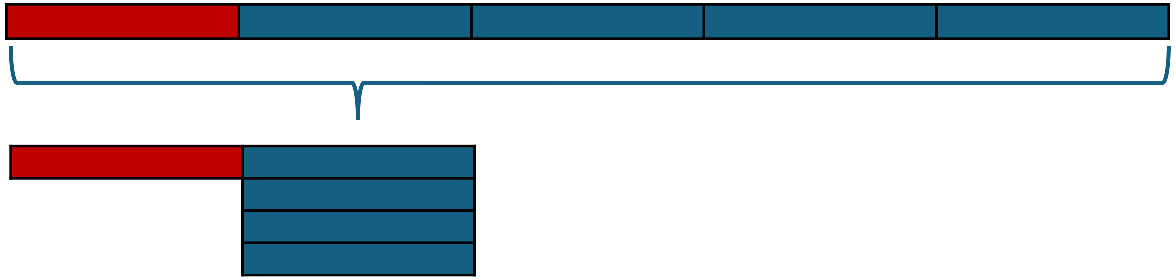
Most of the time when problem size increases mainly the parallel part increases.

➔Speedup increases with problem size!



Weak scaling assumes:

- Constant execution time
- Increase in throughput

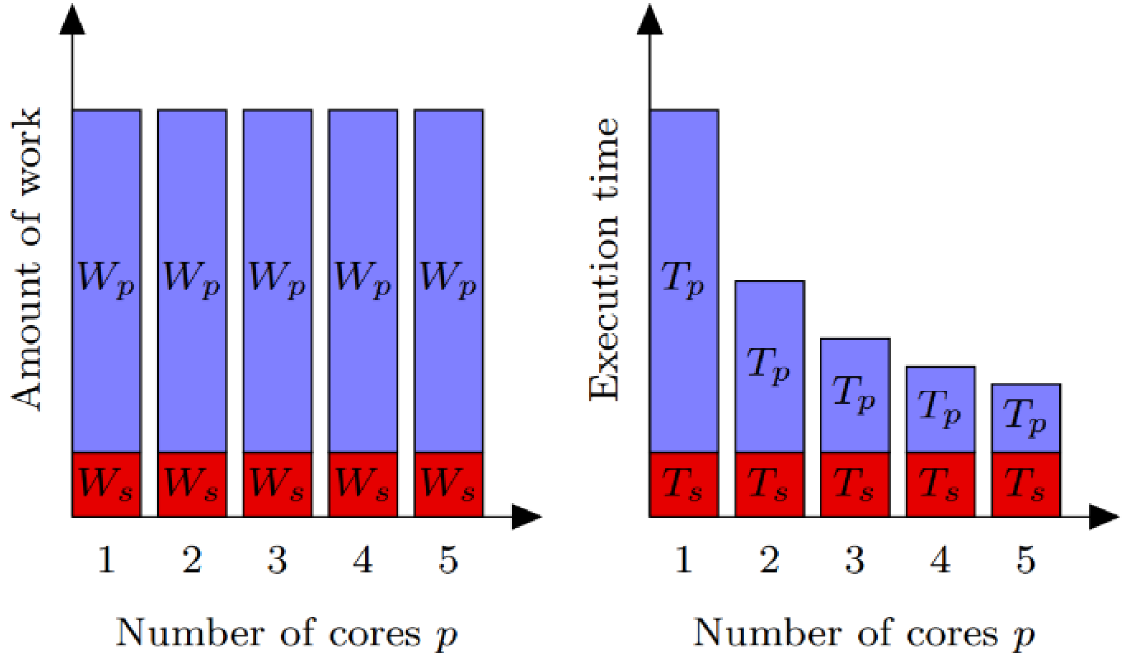


Weak scaling efficiency:

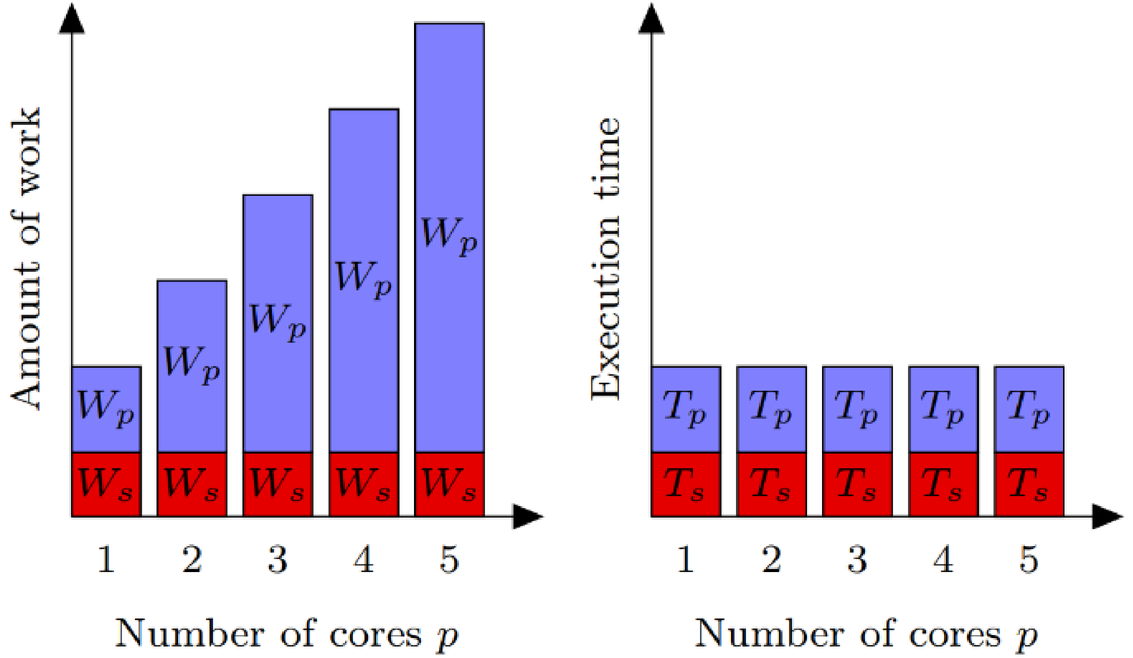
- $\varepsilon(N) = \frac{T(1)}{T(N)}$

(As speedup for strong scaling)

Ahmdal vs Gustafson

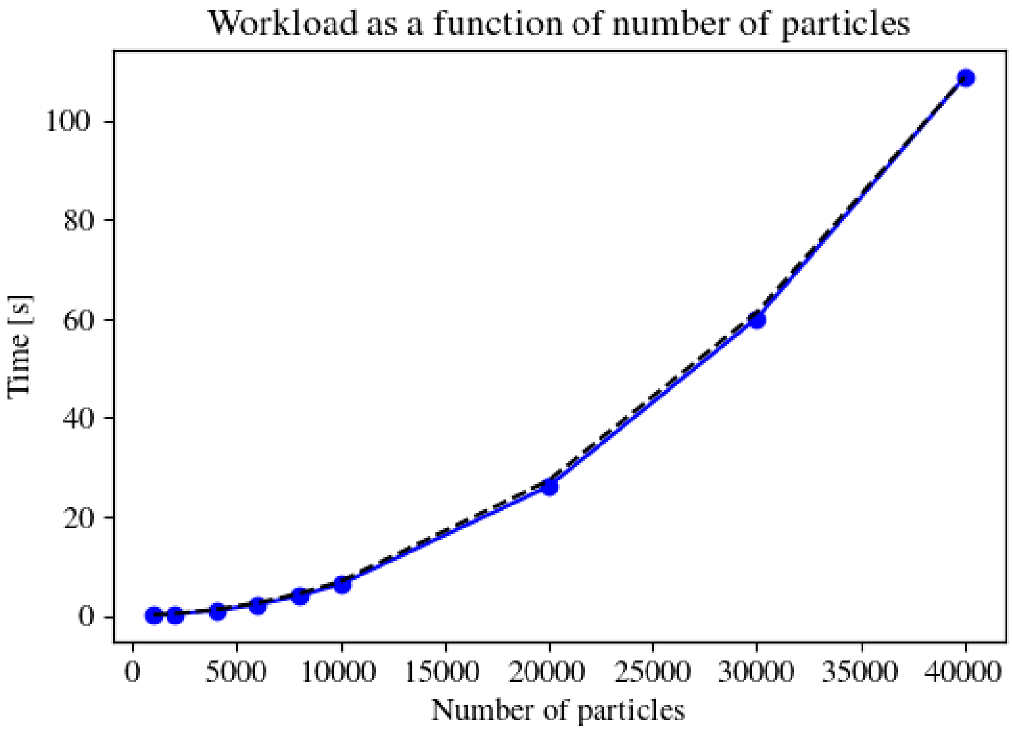


Ahmdal's law



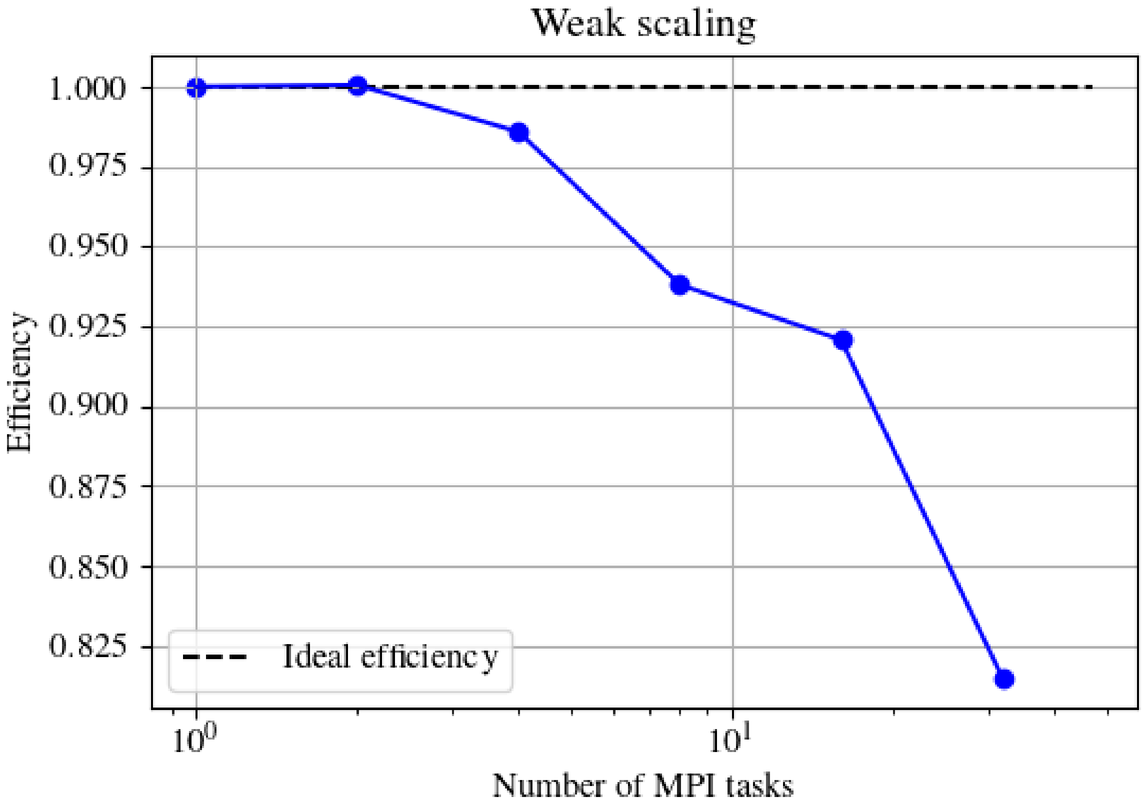
Gustafson's law

Example Code



Example Code

MPI Tasks	Runtime [s]	Npart
1	6.2683	10.000^2
2	6.2649	14.142^2
4	6.3582	20.000^2
8	6.6825	28.284^2
16	6.8083	40.000^2
32	7.6930	56.568^2



Conclusion

- Strong and weak scaling tests are essential when aiming to run a code on a larger resources.

- Strong scaling will tell you the optimal range of resources to use for your problem.

- Weak scaling will tell you for what kind of workloads your code is able to scale.

Sorces

- Gene M. Amdahl: "*Validity of the single processor approach to achieving large scale computing capabilities*". In Proceedings of the April 18-20, 1967, spring joint computer conference (AFIPS '67 (Spring)). Association for Computing Machinery, New York, NY, USA, 483–485. DOI:10.1145/1465482.1465560

- John L. Gustafson: "*Reevaluating Amdahl's law.*" Commun. ACM 31, 5 (May 1988), 532–533

- B.H.H. Juurlink and C. H. Meenderinck. 2012. "*Amdahl's law for predicting the future of multicores considered harmful.*" SIGARCH Comput.Archit.News 40, 2 (May 2012), 1-9.DOI:https://doi.org/10.1145/2234336.2234338