High Performance Computing for Science and Engineering I

Strong and Weak Scaling

Fabian Wermelinger
Computational Science & Engineering Laboratory

OUTLINE

- **Amdahl’s Law — Strong Scaling**
  - Fixed Problem Size
  - How much does parallelism reduce the execution time of a problem?

- **Gustafson’s Law — Weak Scaling**
  - Fixed Execution Time
  - How much longer does it take for the problem without parallelism?

Amdahl's Law — Strong Scaling Analysis

I wrote a shared memory code. How well does my code run in parallel?

Recall Amdahl's Law from the first lecture:

$$S_p = \frac{1}{f + \frac{1-f}{p}}$$

Speedup S with p processors

Amdahl's Law — Strong Scaling Analysis

I wrote a shared memory code. How well does my code run in parallel?

Recall Amdahl's Law from the first lecture:

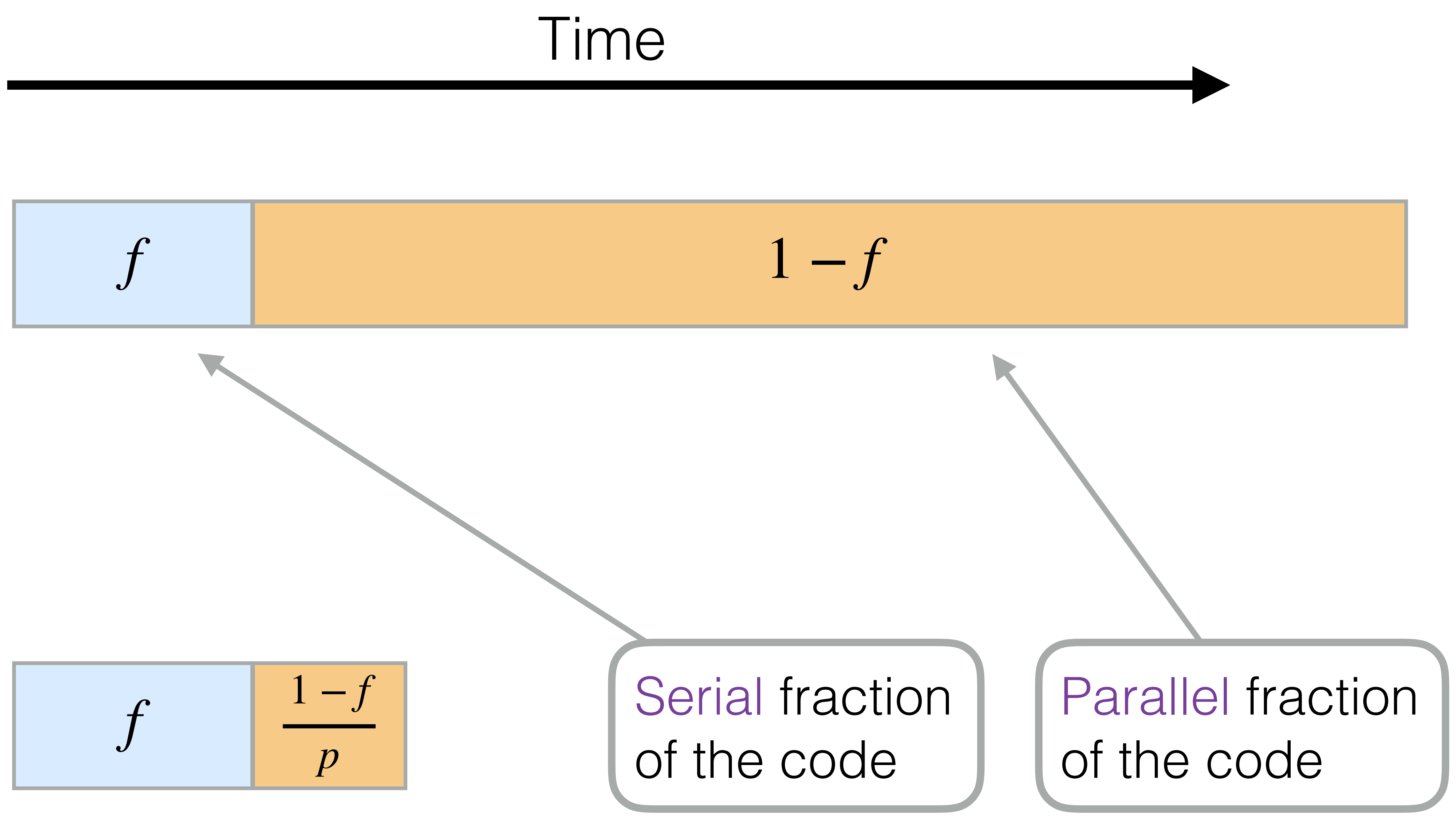
$$S_p = \frac{1}{f + \frac{1-f}{p}}$$

Serial execution

Parallel execution (p processors)

Speedup S with p processors

In a picture:



Serial fraction of the code

Parallel fraction of the code

Amdahl's Law — Strong Scaling Analysis

Implicit assumptions in Amdahl's Law:

- Fixed problem size
  - Makes sense if *p* is relatively small
  - Often we want to keep the execution time constant and increase the problem size (weak scaling)
- Negligible communication cost
  - The number of processors *p* should be small
- All-or-None parallelism
  - A more realistic model would be:

$$S_p = \frac{1}{f_1 + \sum_{i=2}^{p} \frac{f_i}{i}} \quad \text{with} \quad \left(\sum_{i=1}^{p} f_i = 1 \right)$$

Problems for which those assumptions are reasonable can use this model for performance analysis. Such analysis is called **Strong Scaling**.

Recall: Shared memory architectures can not implement a large number of processors due to limitations on the memory bus as well as related cost issues. Communication cost using shared memory is still relatively low compared to distributed memory models.

Amdahl's Law — Strong Scaling Analysis

Implication of fixed problem size:

Speed of a certain task: $\frac{w}{t}$

w: associated **work** (problem size)

t: **time** needed to complete the work

Speed for serial task: w/t_1

Speed for parallel task: w/t_p

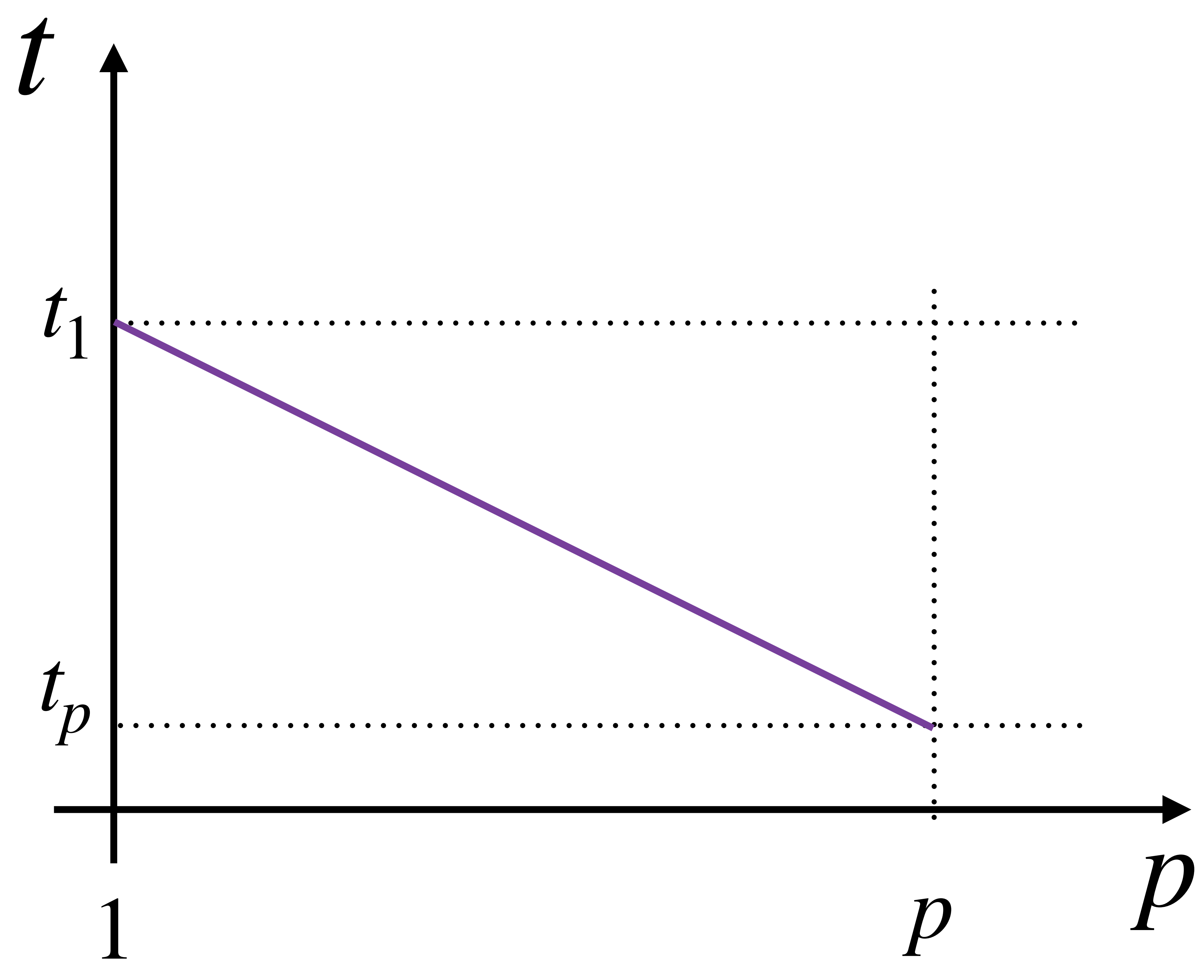
Strong scaling speedup:

$$S_p = \frac{w/t_p}{w/t_1} = \frac{t_1}{t_p}$$

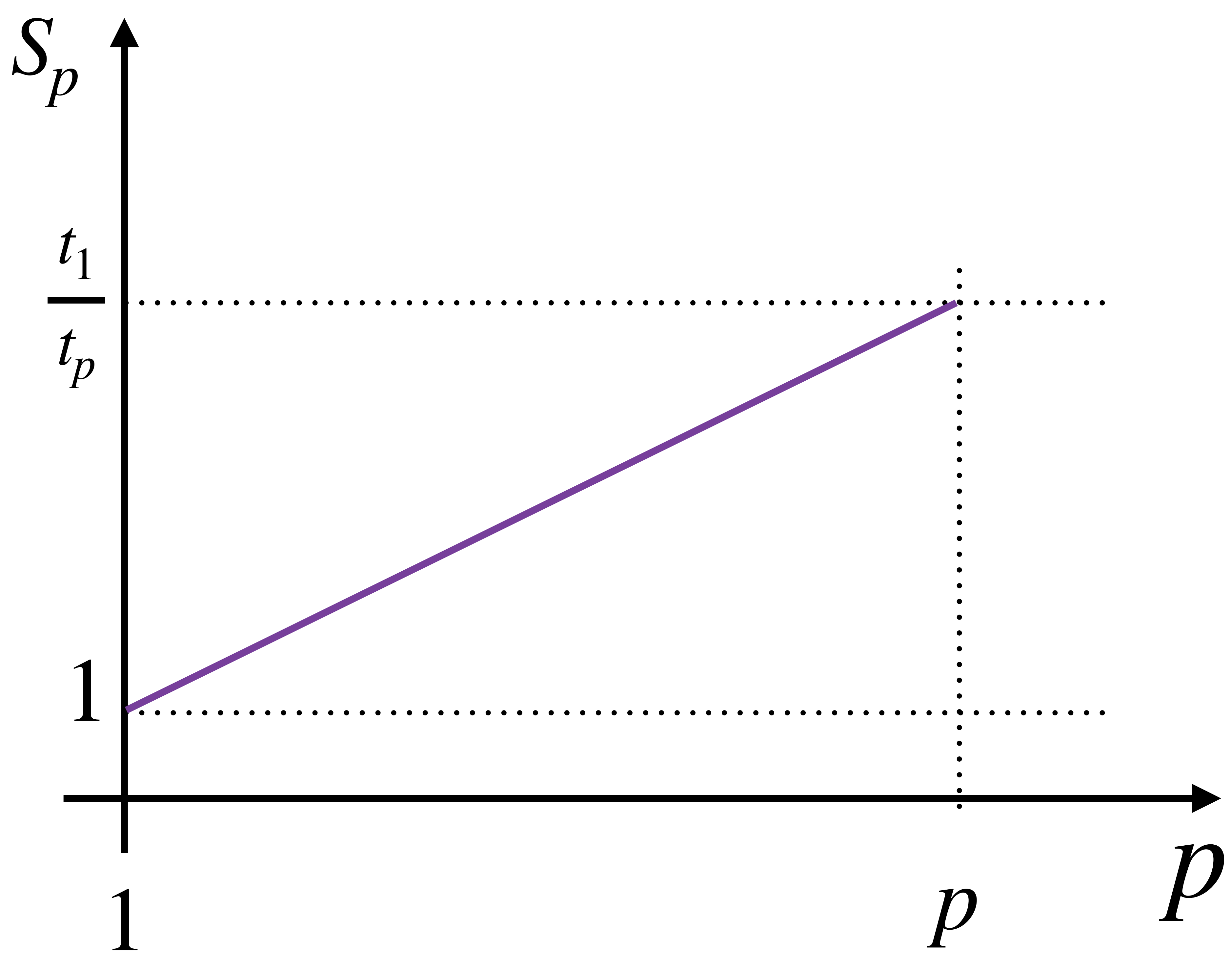
Amdahl's Law — Strong Scaling Analysis

Presenting Strong Scaling data:

Execution time

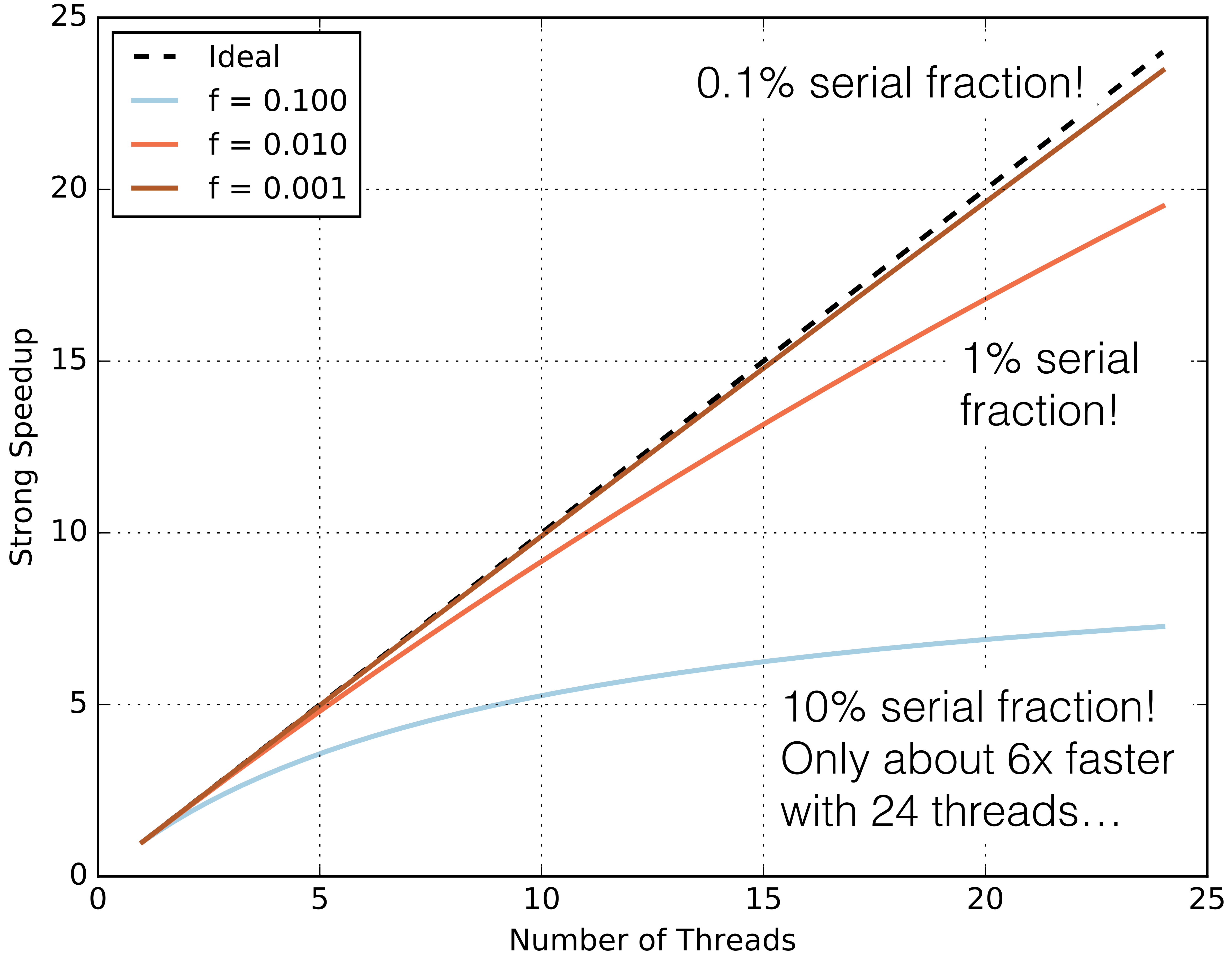


Strong Speedup



Amdahl's Law — Strong Scaling Analysis

Implication of serial fraction f:



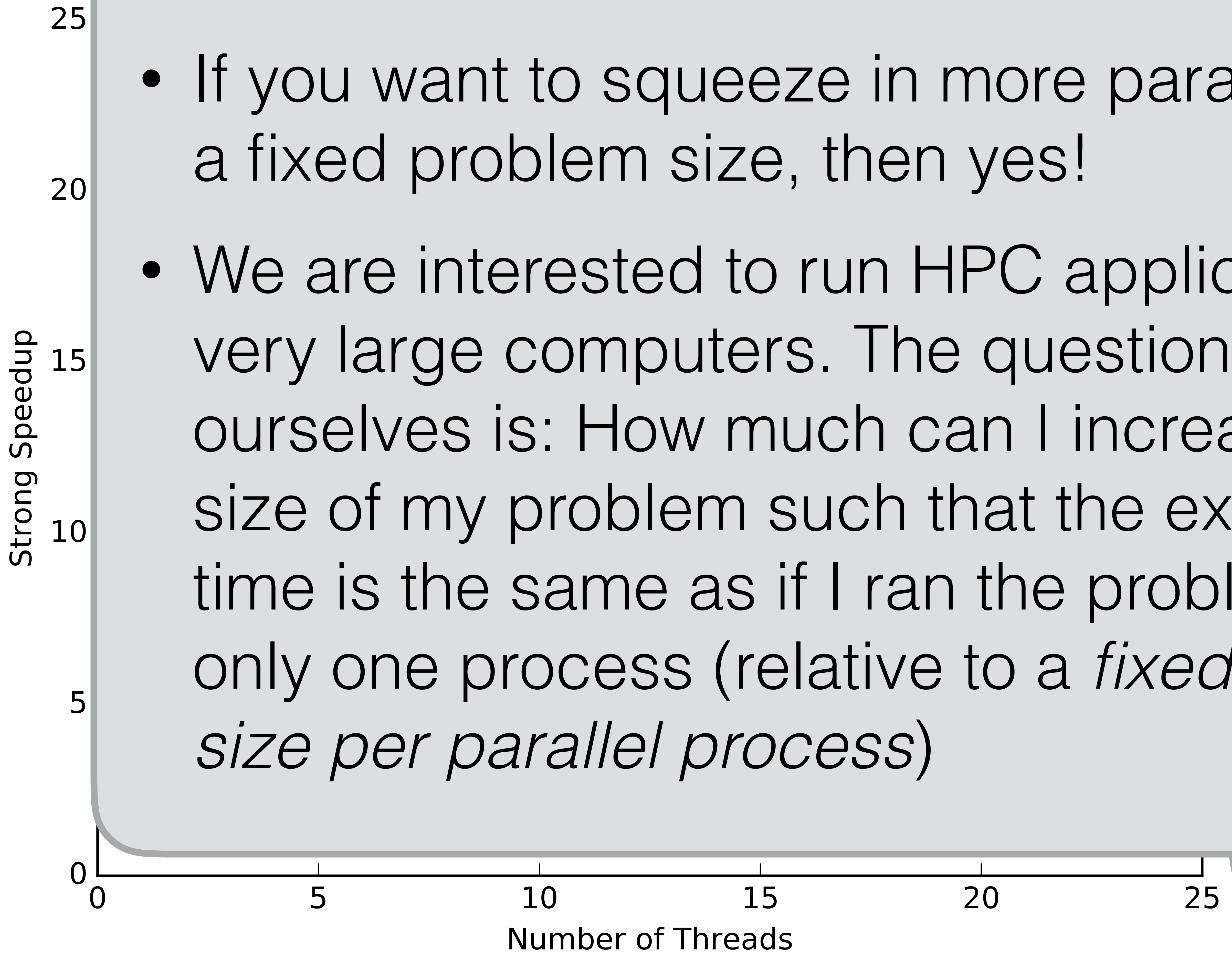
The serial fraction implies a performance upper-bound:

$$\lim_{p\to\infty} S_p = \frac{1}{f}$$

Even with an infinite amount of processors, this is the best we could do. Strong scaling analysis is very sensitive towards the serial fraction. Communication overhead (e.g. synchronization) further degrades performance.

Amdahl's Law — Strong Scaling Analysis

Im



So we are doomed?

- If you want to squeeze in more parallelism for a fixed problem size, then yes!
- We are interested to run HPC applications on very large computers. The question we ask ourselves is: How much can I increase the size of my problem such that the execution time is the same as if I ran the problem with only one process (relative to a *fixed problem size per parallel process*)

he serial fraction implies a erformance upper-bound:

$$\lim_{p\to\infty} S_p = \frac{1}{f}$$

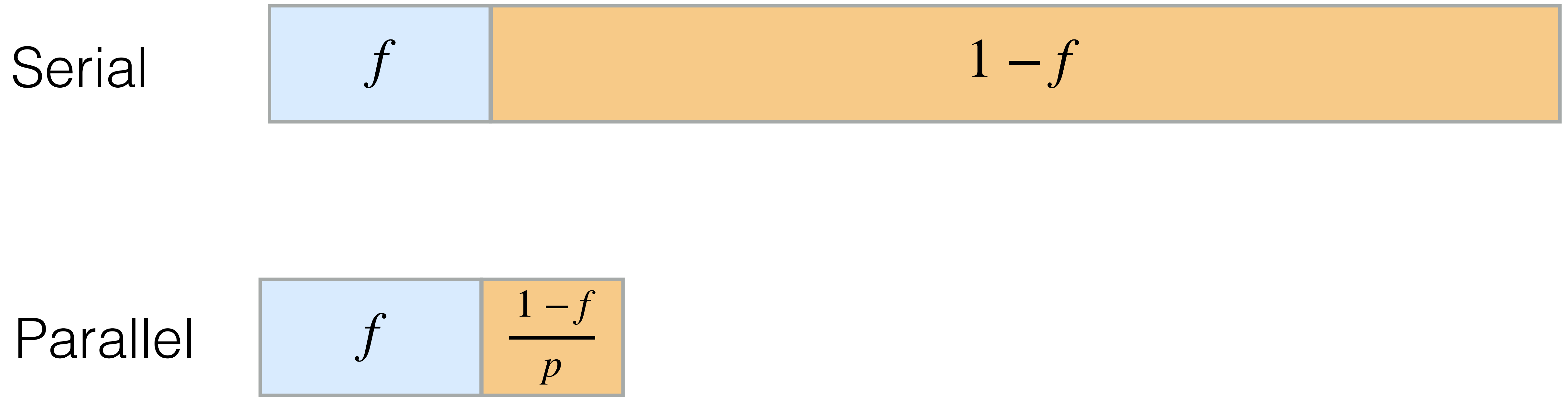
n infinite amount of processors, this we could do. Strong scaling analysis sitive towards the serial fraction. ation overhead (e.g. synchronization) degrades performance.

Gustafson's Law — Weak Scaling Analysis

Amdahl's Law:

Time

Problem size: $f + (1-f) = 1$

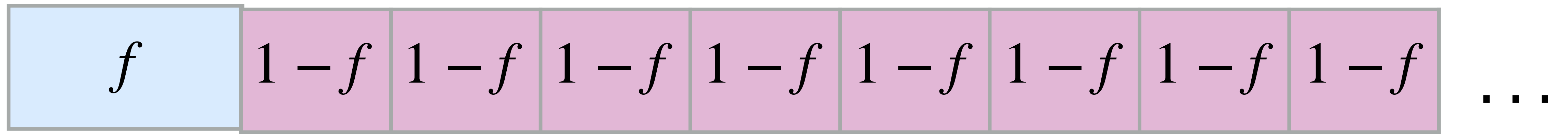


How much faster am I with p processors for fixed problem size?

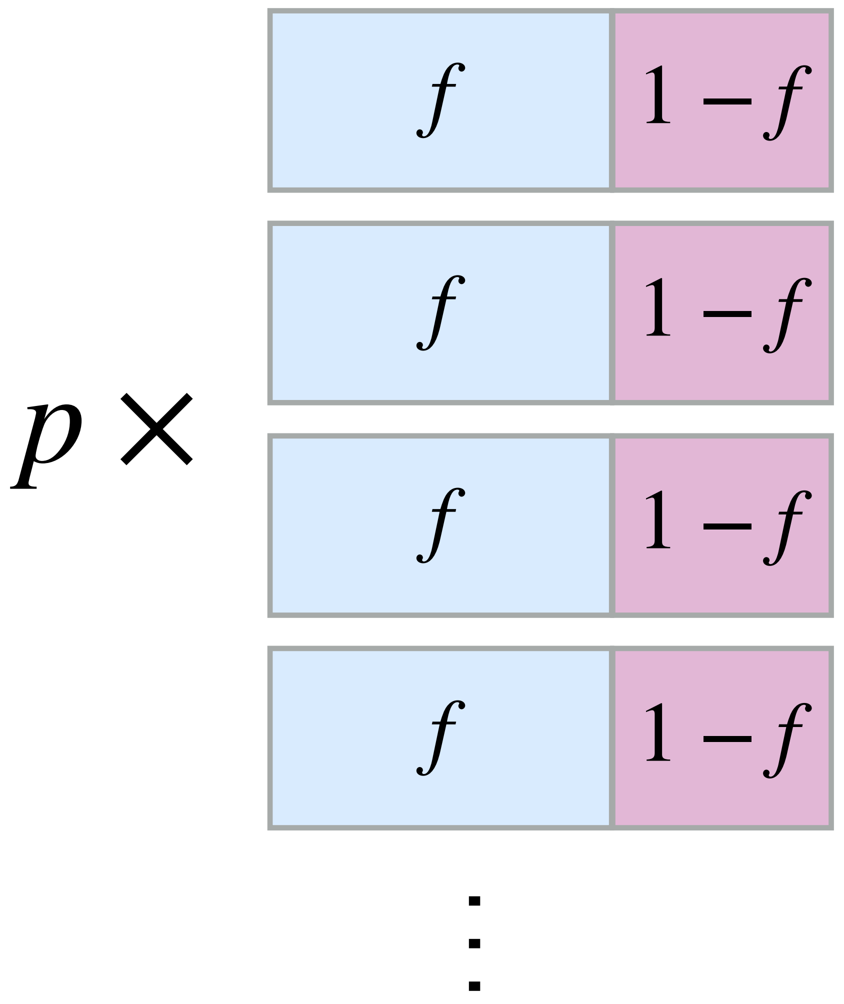
Gustafson's Law:

Time

Problem size: $f + (1-f) + (1-f) + \ldots = f + p(1-f) > 1$



Problem size per process: $f + (1-f) = 1$



Work load *per process* is constant!

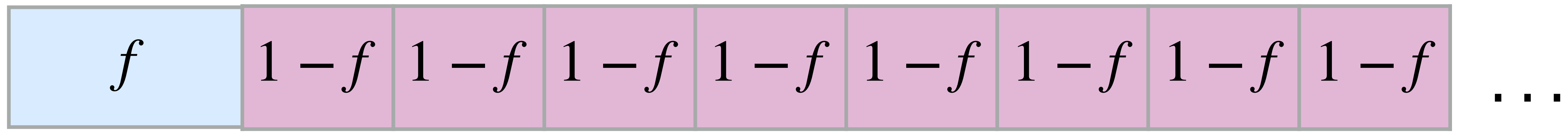
How much longer does it take for a given workload when parallelism is absent?

Gustafson's Law — Weak Scaling Analysis

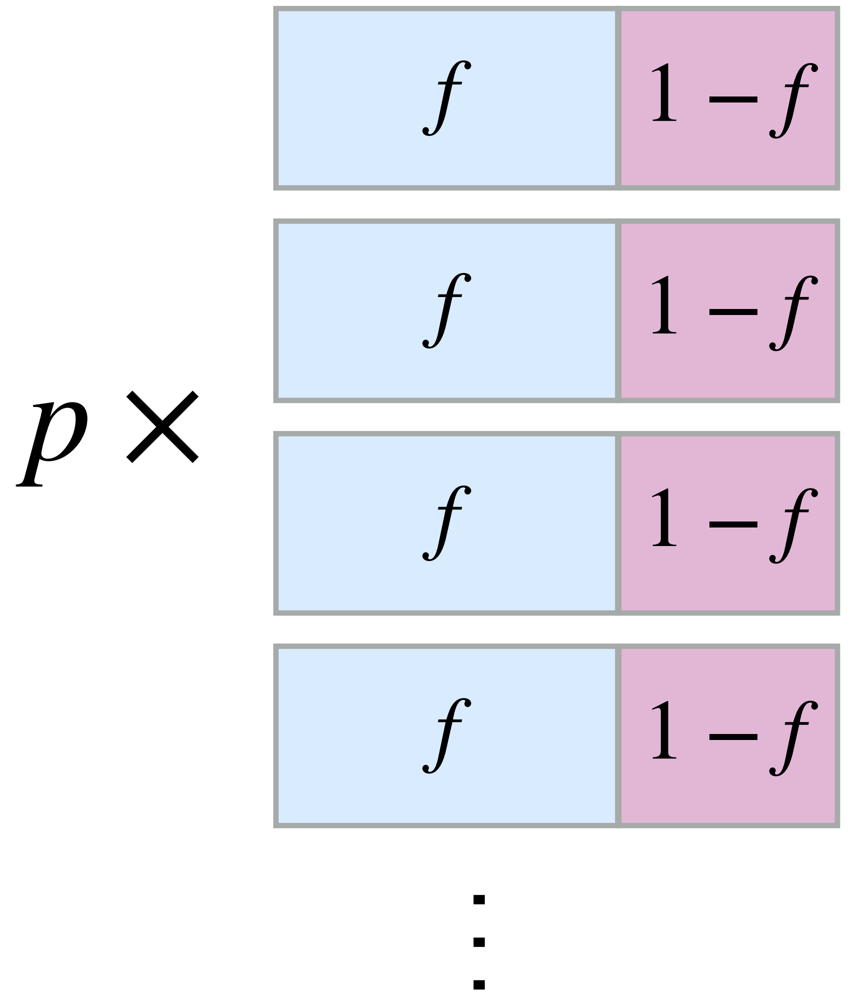
Gustafson's Law:

Time

Problem size: $f + (1-f) + (1-f) + \ldots = f + p(1-f) > 1$



Problem size per process: $f + (1-f) = 1$



Work load *per process* is constant!

How much longer does it take for a given workload when parallelism is absent?

Speedup:

$$S_p = f + p(1-f)$$

Serial fraction is *unaffected* by parallelization!

Main question: How well does the *parallel fraction scale* among p processors?

Gustafson's Law — Weak Scaling Analysis

Implication of fixed problem size per process:

Speed of a certain task: $\boxed{\dfrac{w_1}{t}}$

w_1: associated **work** (problem size *per process*)

t: **time** needed to complete the work

Speed for serial task: w_1/t_1

Speed for parallel task: pw_1/t_p

Weak scaling speedup:

$$S_p = \frac{pw_1/t_p}{w_1/t_1} = p\frac{t_1}{t_p}$$

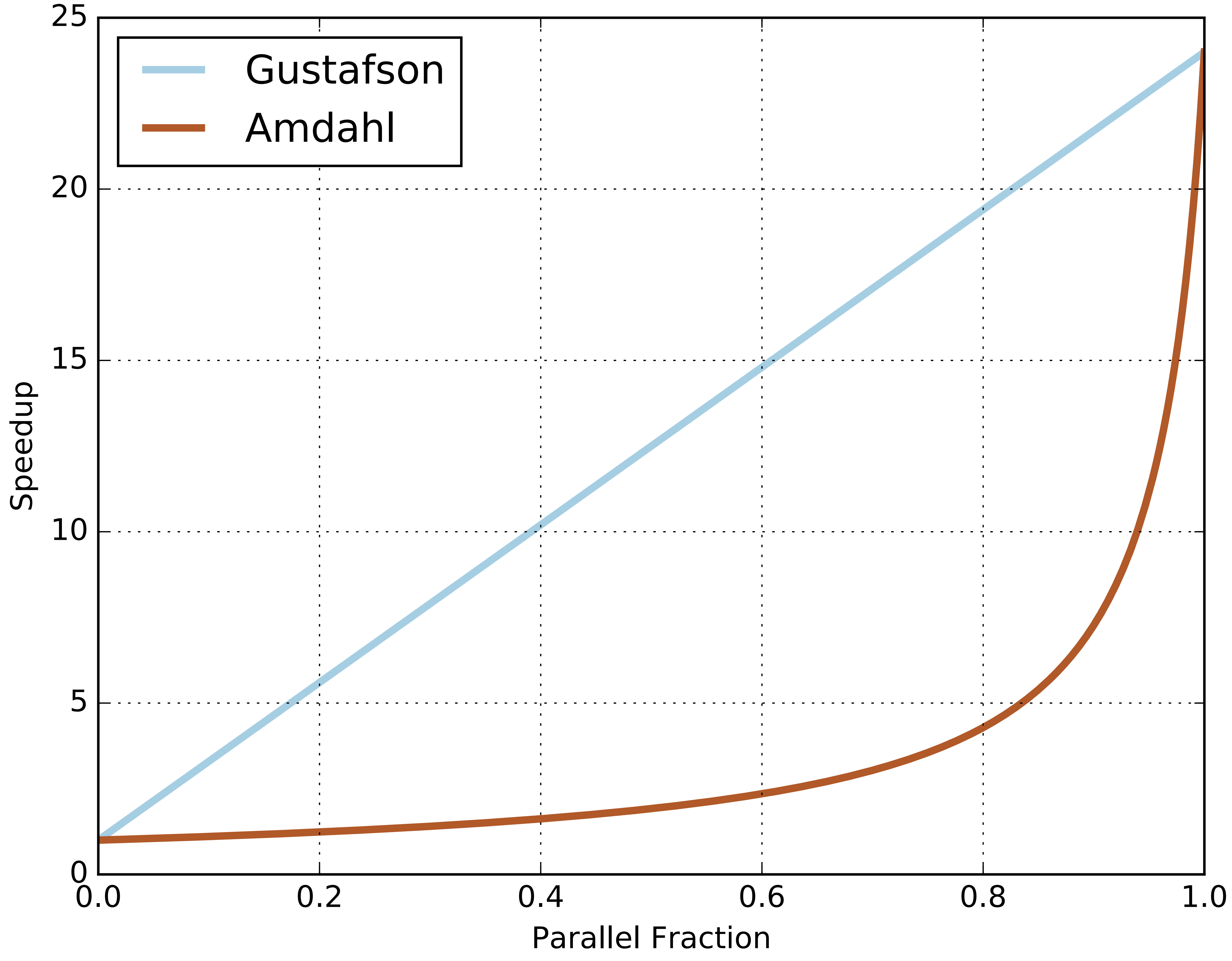
Weak scaling efficiency:

$$\boxed{E_w = \frac{S_p}{p} = \frac{t_1}{t_p}}$$

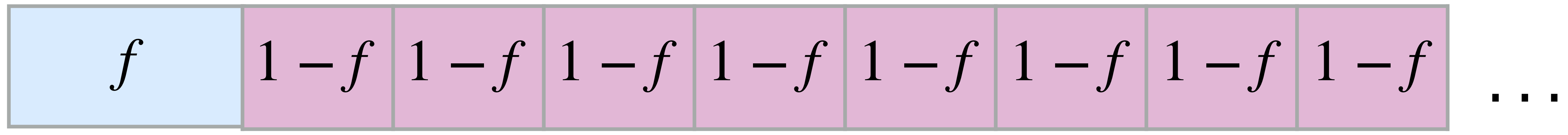
Gustafson's Law — Weak Scaling Analysis

Gustafson's Point of View:

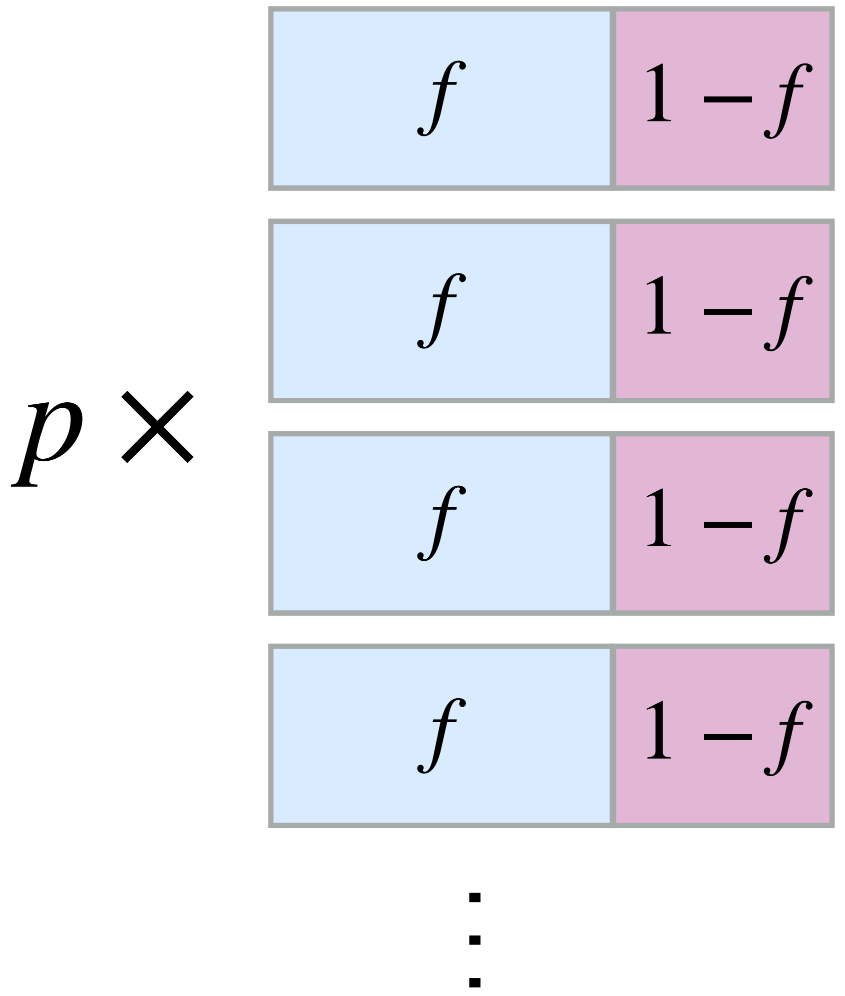
(Example shown for $p = 24$ processors)



Problem size: $f + (1-f) + (1-f) + \ldots = f + p(1-f) > 1$



Problem size per process: $f + (1-f) = 1$

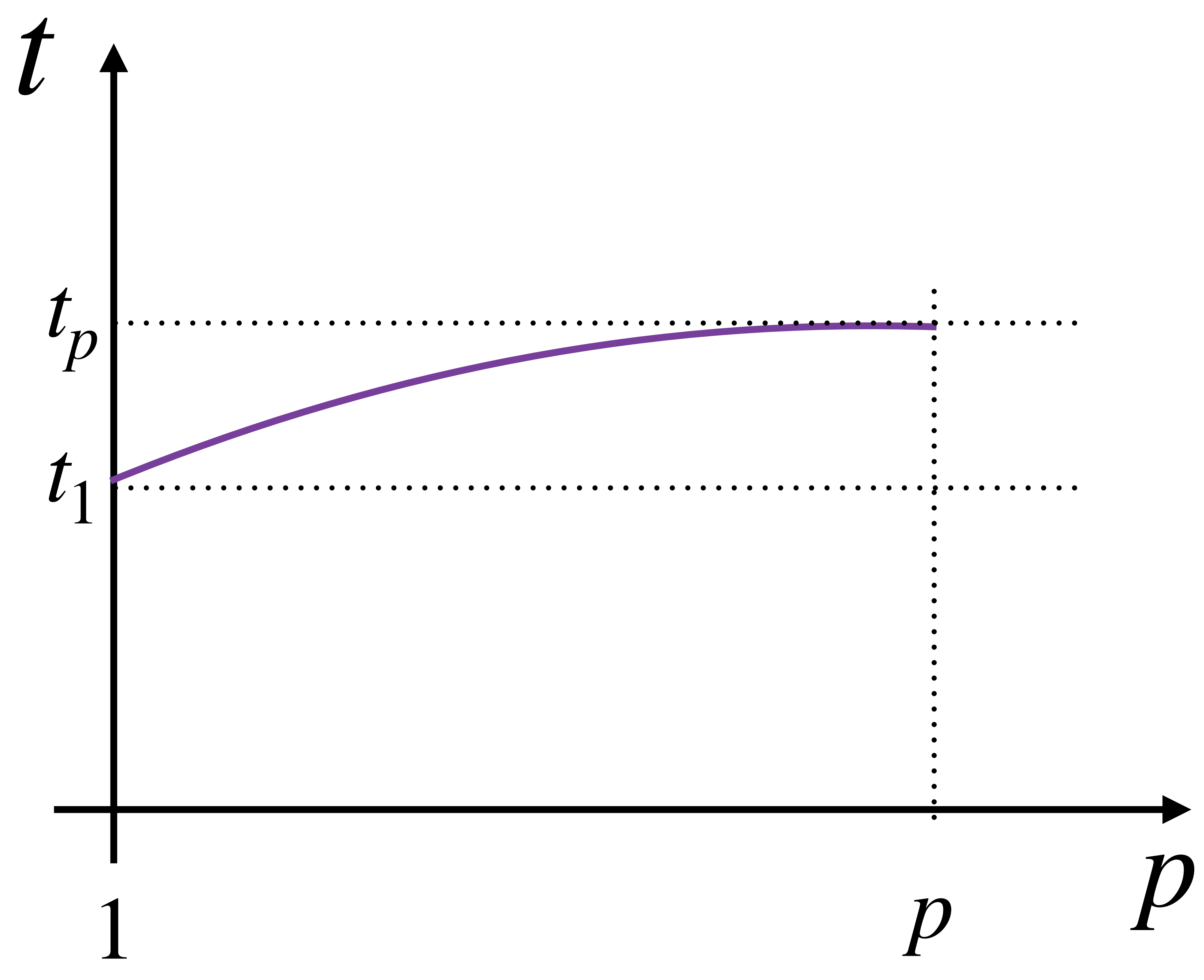


Gustafson's Law scales relative to the **parallel fraction** of a code. The serial fraction *does not affect the scaling*.

Gustafson's Law — Weak Scaling Analysis

Presenting Weak Scaling data:

Execution time



Weak efficiency