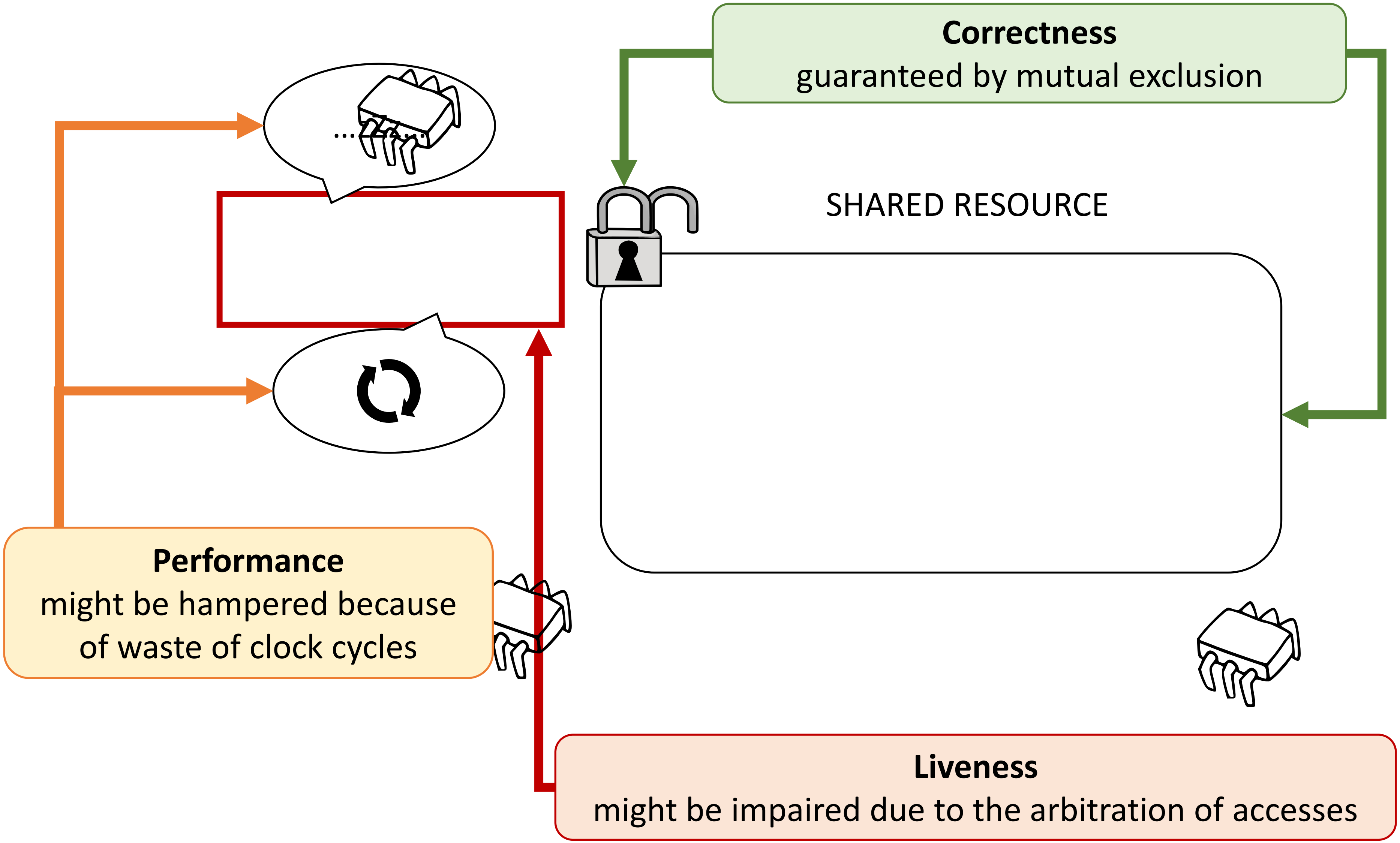
Programmazione concorrente

Laurea Magistrale in Ingegneria Informatica
Università Tor Vergata
Docente: Romolo Marotta

Properties of Concurrent Programs

1. Scalability
2. Correctness
3. Progress

On concurrent programming



What do we want from parallel programs?

- **Safety:** *nothing wrong happens* (Correctness)
  - parallel versions of our programs should be correct as their sequential implementations
- **Liveliness:** *something good happens eventually* (Progress)
  - if a sequential program terminates with a given input, we want that its parallel alternative also completes with the same input
- **Performance**
  - we want to exploit our parallel hardware

A bit of terminology

- Hardware
  - Processor
  - CPU
  - CPU-Core
  - Logical Core
  - Hardware thread
- Software
  - Process
  - Thread
  - Fiber
  - Task
- Programs
  - Sequential
  - Concurrent
  - Parallel
  - Distrubuted
- Memory
  - Shared
  - Distributed

The system model

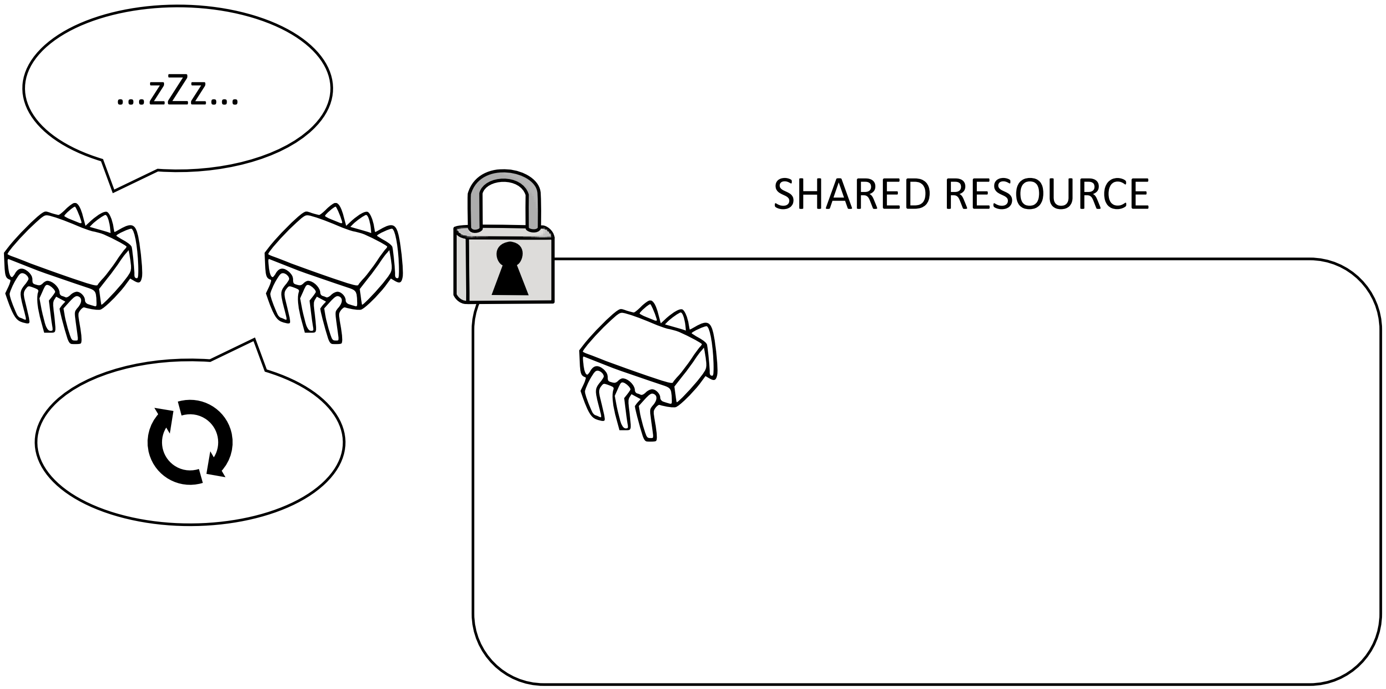
- Threads (aka processes)
- Cores (aka cpus)
- Shared memory
- Arbitrary long asynchronous delays
- Scheduler
  - A system component that decides which/when a thread runs on a given core

Scalability

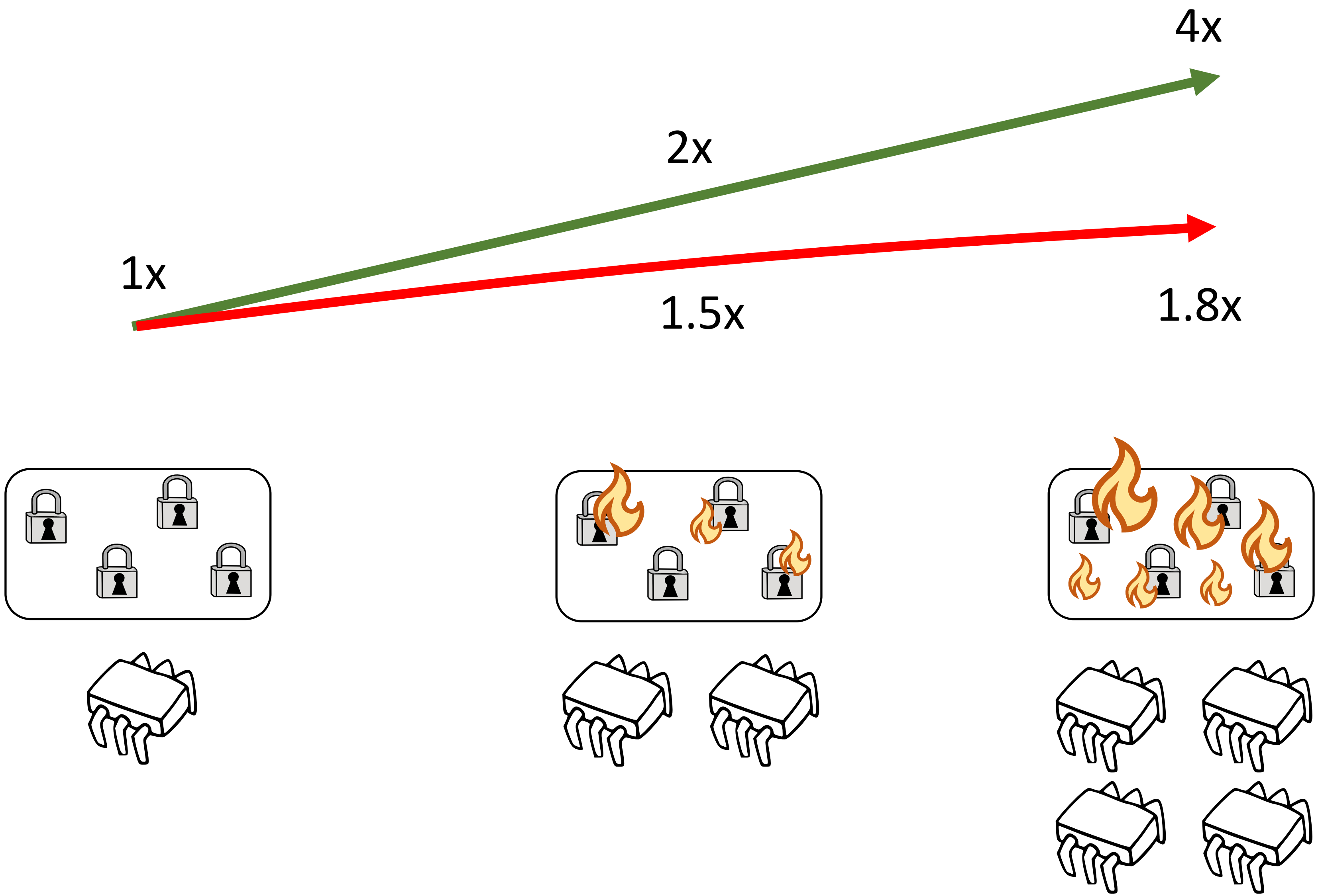
Correctness conditions

Progress conditions

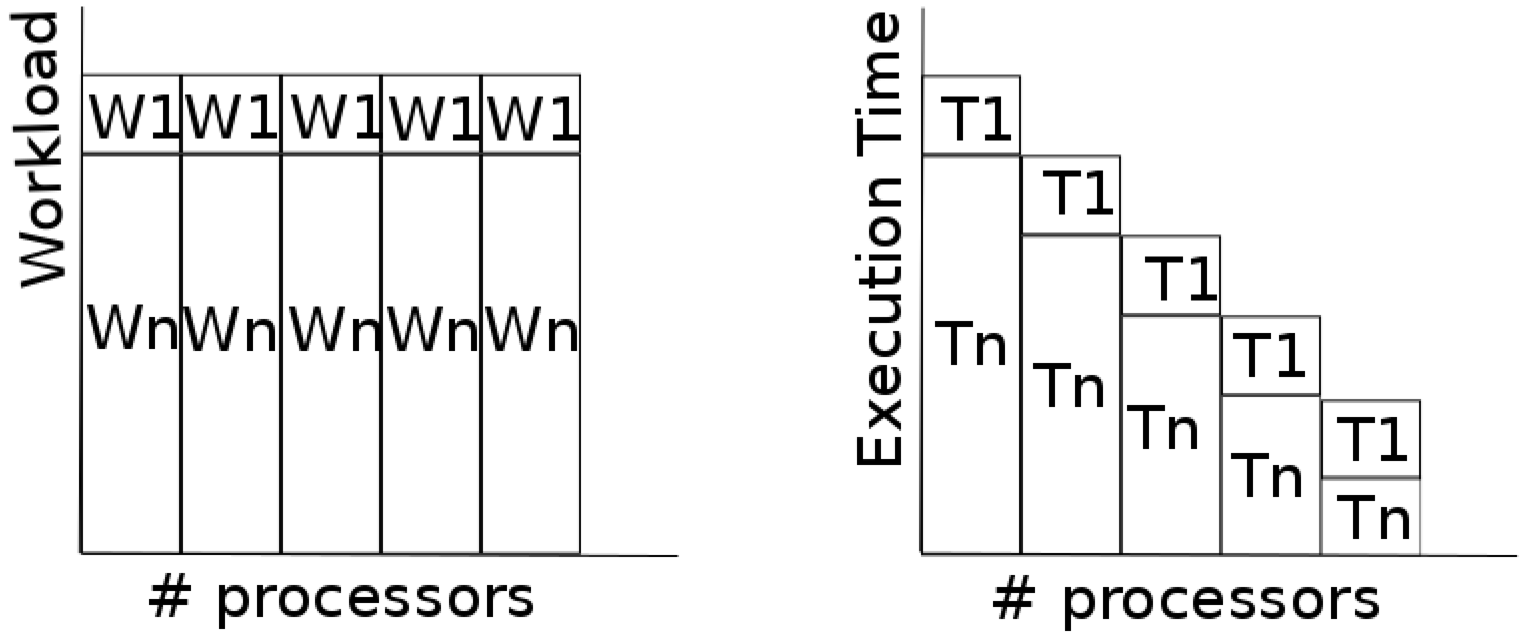
The cost of synchronization



The cost of synchronization



Amdahl Law – Fixed-size Model (1967)



Amdahl Law – Fixed-size Model (1967)

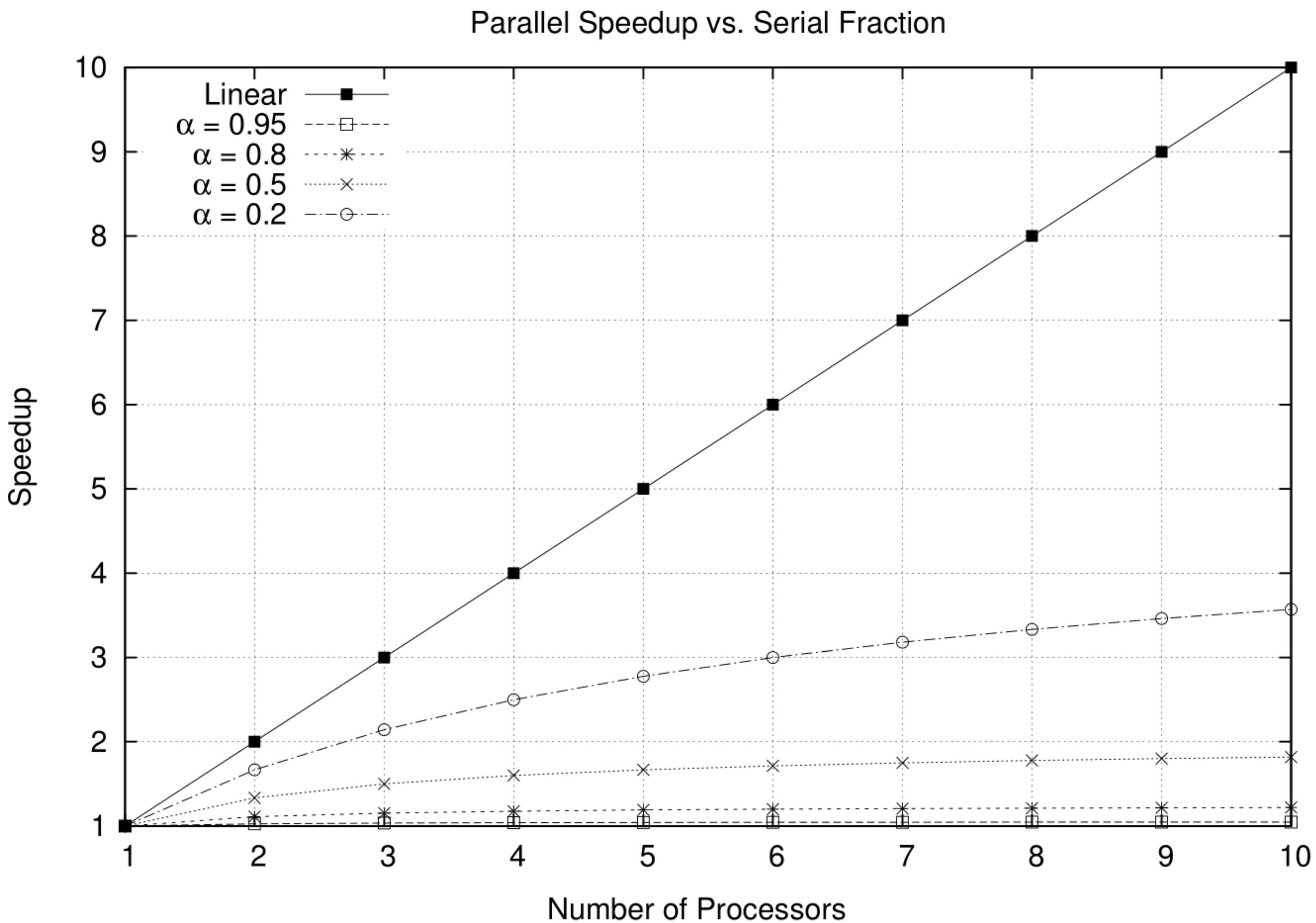
- The workload is fixed: it studies how the behavior of the same program varies when adding more computing power

$$S_{Amdahl} = \frac{T_s}{T_p} = \frac{T_s}{\alpha T_s + (1-\alpha)\frac{T_s}{p}} = \frac{1}{\alpha + \frac{(1-\alpha)}{p}}$$

- where:
  - $\alpha \in [0,1]$: Serial fraction of the program
  - $p \in N$: Number of processors
  - T_s : Serial execution time
  - T_p : Parallel execution time
- It can be expressed as well vs. the parallel fraction

$$P = 1 - \alpha$$

Amdahl Law – Fixed-size Model (1967)



How real is this?

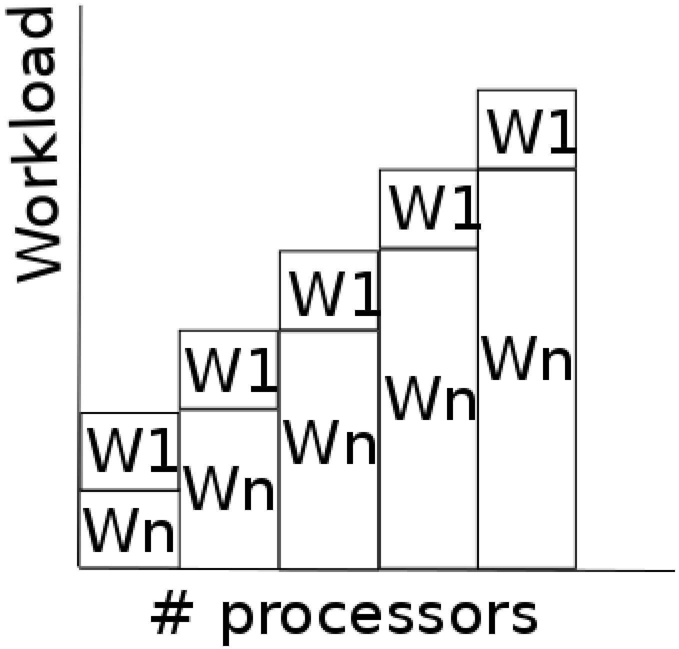
$$\lim_{p\to\infty} S_{Amdahl} = \lim_{p\to\infty} \frac{1}{\alpha + \frac{(1-\alpha)}{p}} = \frac{1}{\alpha}$$

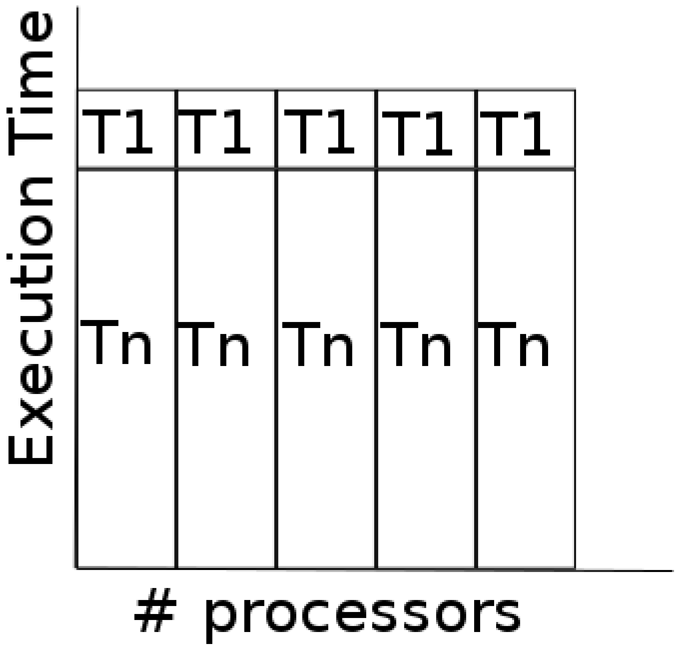
- If the sequential fraction is 20%, we have:

$$\lim_{p\to\infty} S_{Amdahl} = \frac{1}{0.2} = 5$$

- Speedup 5 using infinite processors!

Fixed-time model





Gustafson Law—Fixed-time Model (1989)

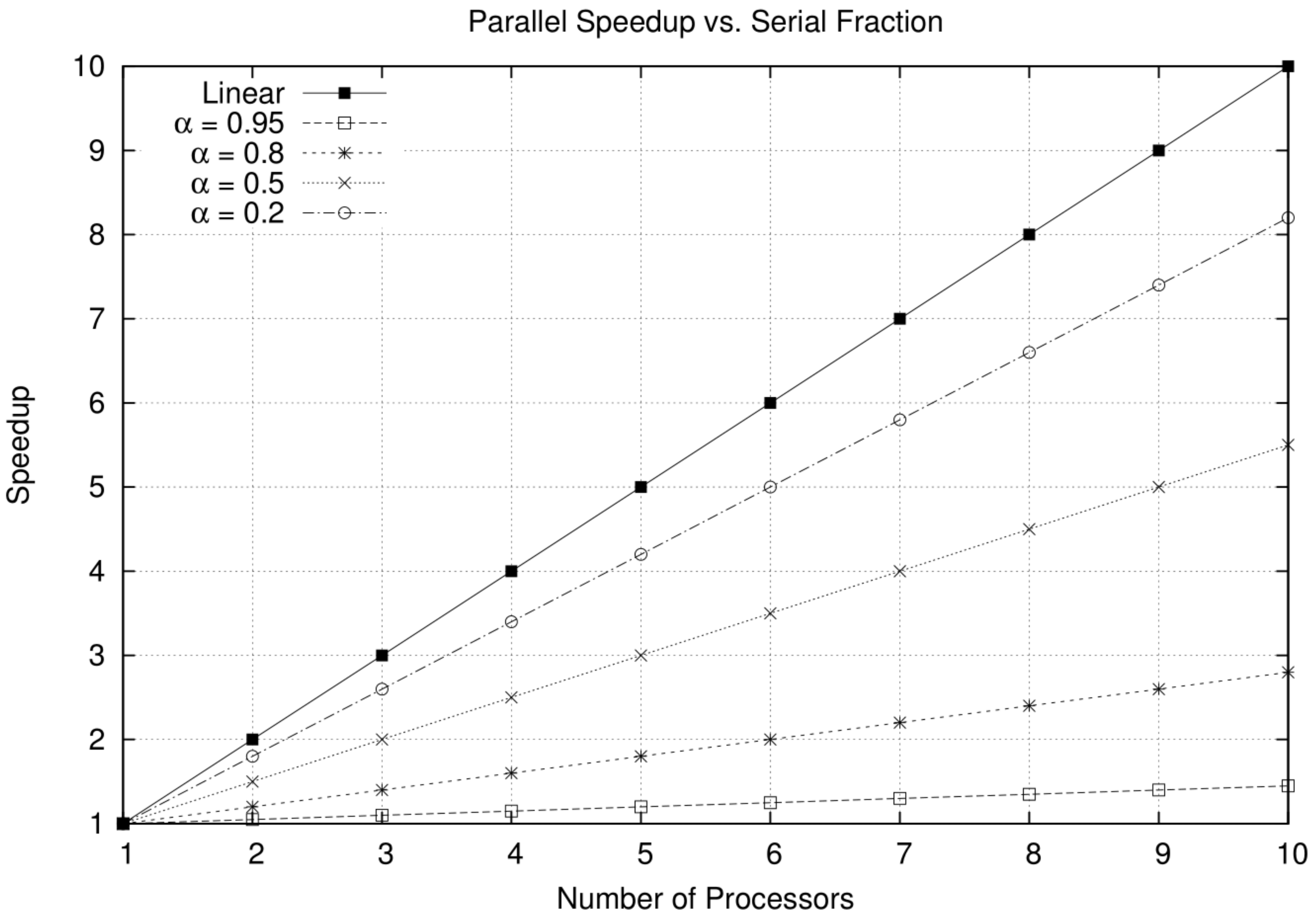
- The execution time is fixed: it studies how the behavior of the *scaled* program varies when adding more computing power

$$W' = \alpha W + (1 - \alpha) p W$$

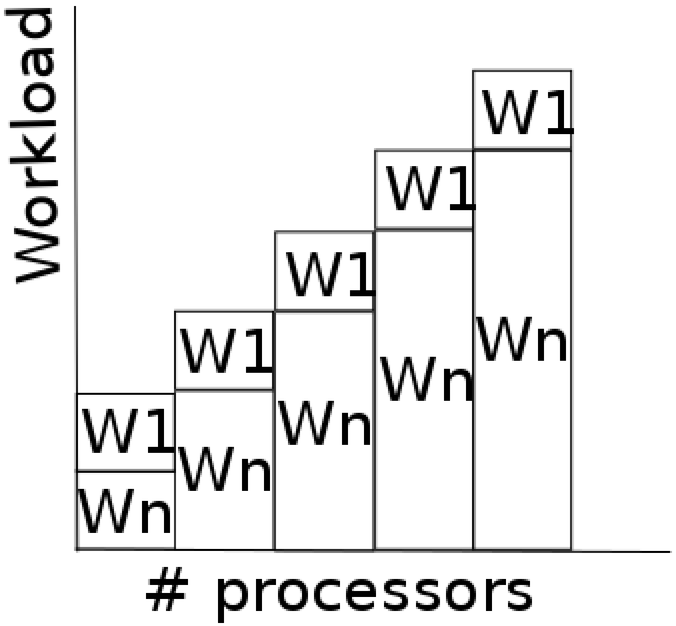
$$S_{Gustafson} = \frac{W'}{W} = \alpha + (1 - \alpha) p$$

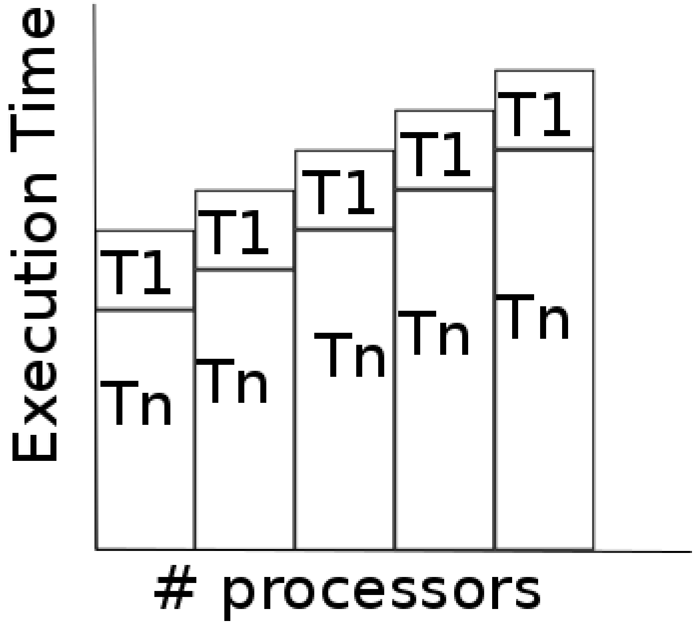
- where:
  - $\alpha \in [0,1]$: Serial fraction of the program
  - $p \in N$: Number of processors
  - W : Original workload
  - W' : Scaled workload

Speed-up according to Gustafson



Memory-bounded model





Sun Ni Law—Memory-bounded Model (1993)

- The workload is scaled, bounded by memory

$$S_{Sun-Ni} = \frac{sequential\ time\ for\ W^*}{parallel\ time\ for\ W^*}$$

$$S_{Sun-Ni} = \frac{\alpha W + (1-\alpha)G(p)W}{\alpha W + (1-\alpha)G(p)\frac{W}{p}} = \frac{\alpha + (1-\alpha)G(p)}{\alpha + (1-\alpha)\frac{G(p)}{p}}$$

- where:
  - $G(p)$ describes the workload increase as the memory capacity increases
  - $W^* = \alpha W + (1-\alpha)G(p)W$

Speed-up according to Sun Ni

$$S_{Sun-Ni} = \frac{\alpha + (1-\alpha)G(p)}{\alpha + (1-\alpha)\frac{G(p)}{p}}$$

- If $G(p) = 1$

$$S_{Amdahl} = \frac{1}{\alpha + \frac{(1-\alpha)}{p}}$$

- If $G(p) = p$

$$S_{Gustafson} = \alpha + (1-\alpha)p$$

- In general, $G(p) > p$ gives a higher scale-up

Superlinear speedup

- Can we have a Speed-up > p ? Yes!
  - Workload increases more than computing power (G(p) > p)
  - Cache effect: larger accumulated cache size. More or even all of the working set can fit into caches and the memory access time reduces dramatically
  - RAM effect: enables the dataset to move from disk into RAM drastically reducing the time required, e.g., to search it.

Scalability

- Efficiency

$$E = \frac{speedup}{\#processors}$$

- **Strong Scalability**: If the efficiency is kept fixed while increasing the number of processes and maintain fixed the problem size

- **Weak Scalability**: If the efficiency is kept fixed while increasing at the same rate the problem size and the number of processes