Loss of Parallel Efficiency

The Sequential Fraction

Amdahl's Law theoretically shows that an application with a sequential fraction has its acceleration limited by it:

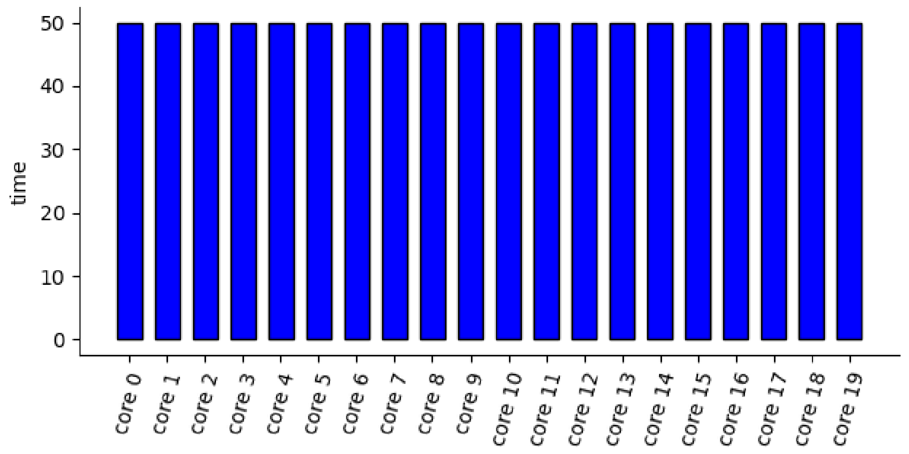
$$Sp(p, F_s) = \frac{1}{F_s}$$

Load Imbalance

- if not all cores have the same workload, the execution time will be the one of the last core to finish.
- in the extreme case where only one core is working, it is equivalent to a sequential computation, having $F_s = 1$

Load Imbalance (Example)

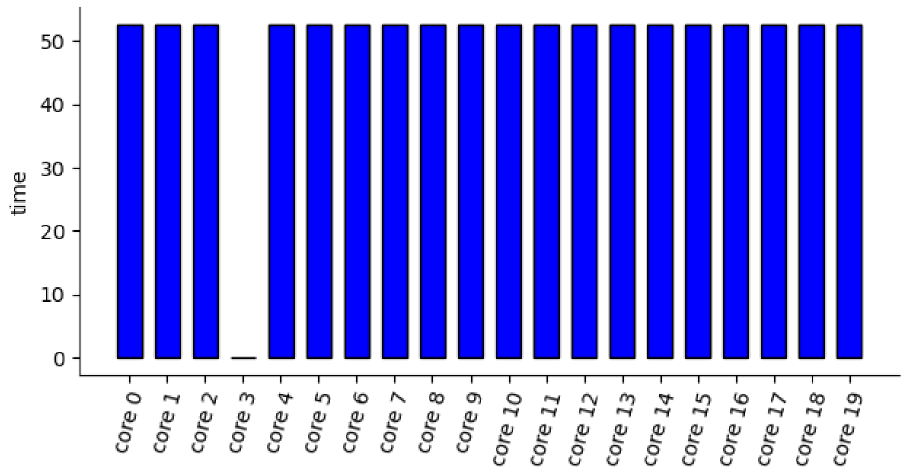
20-core computer / perfectly balanced workload



$$Sp(20) = 20$$

Load Imbalance (Example)

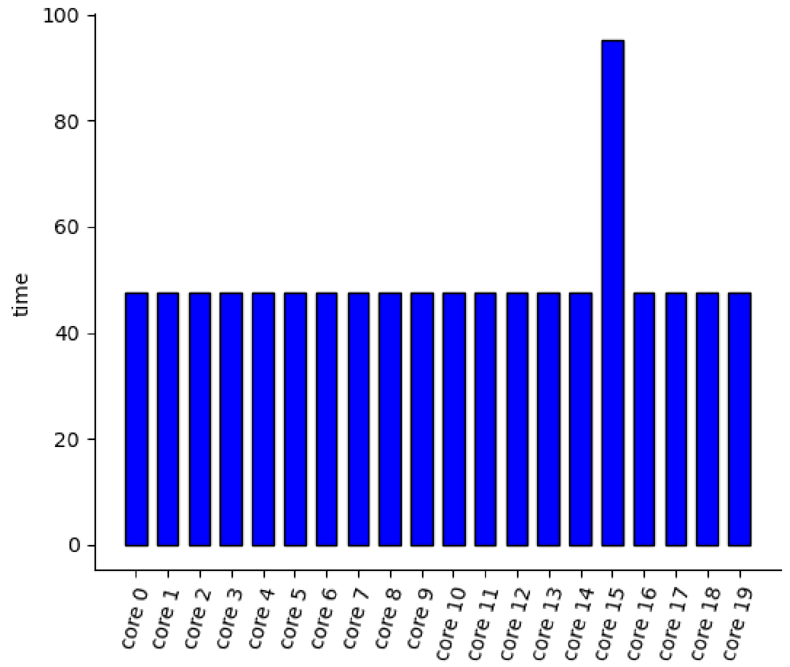
20-core computer / one core having nothing to do



$$Sp(20) = 19$$

Load Imbalance (Example)

20-core computer / one core with twice the workload



$$Sp(20) = 10.5$$

Interactions

During a computation, there are typically interactions between the cores:

- communications in distributed memory
- concurrent memory access (False-sharing) in shared memory

These interactions represent an **overhead** and **degrade** performance based on **their number** and **their volume**.

Interactions: Application Granularity

Definition: Application granularity is the ratio of the time the application spends computing to the time it spends communicating.

$$Ga = \frac{Tccomp}{Tcomm}$$

The larger Ga, the better the parallel performance.

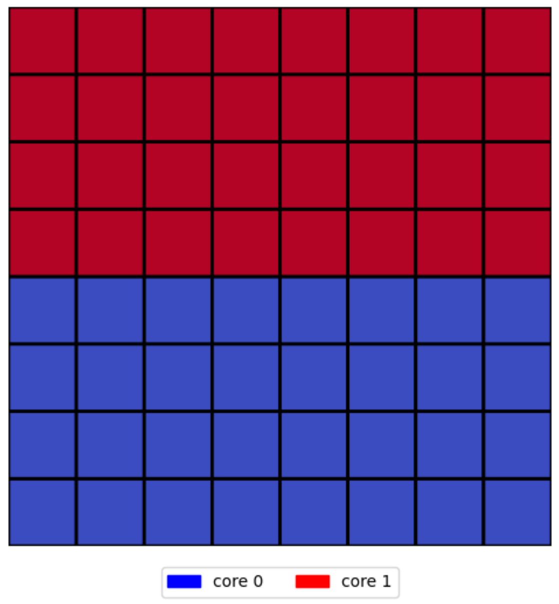
Interactions: Partitioning

The goal of partitioners is to divide a domain while trying to balance the workload and minimize interactions.

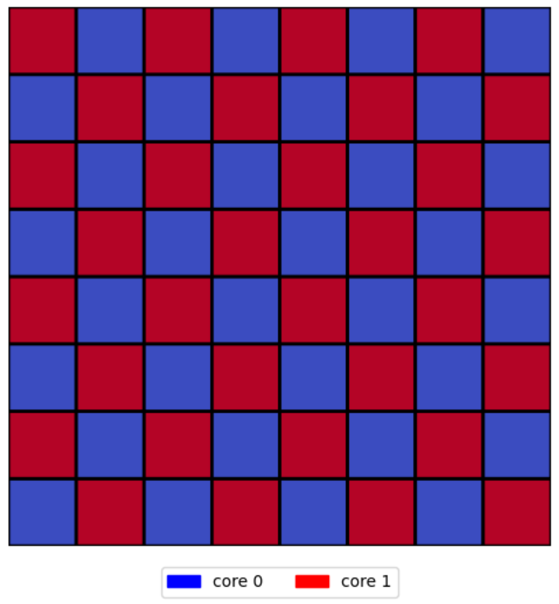
Interactions: Partitioning

Exemple : Dividing a square into two equal parts

Good Partition

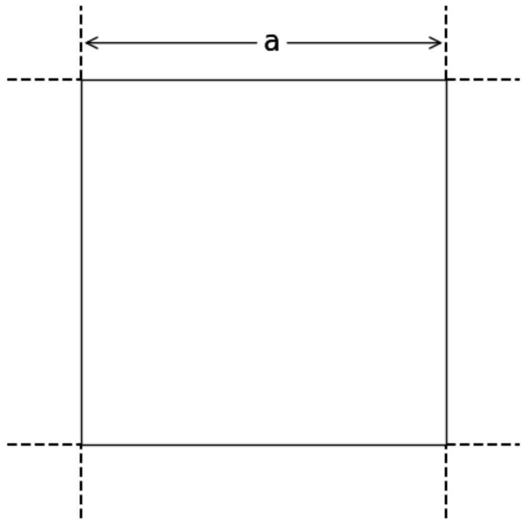


Bad Partition



Interactions: Size matters

Example: For a square with side length a



- Computation cost proportional to the area: a^2
- Interaction cost proportional to the perimeter: $4a$

Interactions: Size matters

a	**area**	**prerimeter**	Ga
0.5	0.25	2	0.125
1	1	4	0.25
2	4	8	0.5
4	16	16	1
8	64	32	2

Execution at a Very Large Scale

With a large number of cores, other sources can deteriorate parallel performance:

- Collective communications (such as MPI_Alltoall)
- Input/Output
- Application launching (including MPI_Init)

Main Losses of Parallel Efficiency (Summary)

- The sequential fraction
- Load imbalance
- Application granularity too low (too many interactions compare to computation)
  - poor domain partitioning
  - a too small domain
- Very large scale computation