Speedup can be defined as S = Ts / Tp where Ts is serial time and Tp is parallel time.

Efficiency = S / p where S is speedup and p is the number of processes. For example, if we get the same speedup with 6 and 7 processors, we use 6 processors. It is more efficient. Because every new processor comes with costs or overhead in parallel program. This is why we can not get linear speedup in general cases.

Scalability: In a parallel programming, when we increase the number of processes and the size of input, and we get the same efficiency, then we can say that it is scalable.

Amdahl: Amdahl says that if we can not parallelize serial programs completely, the speedup will be very limited. Even if we have for example 1000 cores, our speedup will be limited independently from core amount.

Suppose you have a sequential code and that a fraction f of its computation is parallelized and run on N processing units working in parallel, while the remaining fraction 1-f cannot be improved, i.e., it cannot be parallelized. Amdahl’s law states that the speedup achieved by parallelization is

[enter image description here](https://i.stack.imgur.com/DRZlm.gif)

Gustafson: Amdahl’s point of view is focused on a fixed computation problem size as it deals with a code taking a fixed amount of sequential calculation time. Gustafson's objection is that massively parallel machines allow computations previously unfeasible since they enable computations on very large data sets in fixed amount of time. In other words, a parallel platform does more than speeding up the execution of a code: it enables dealing with larger problems.

Suppose you have an application taking a time ts to be executed on N processing units. Of that computing time, a fraction (1-f) must be run sequentially. Accordingly, this application would run on a fully sequential machine in a time t equal to

[enter image description here](https://i.stack.imgur.com/sLkyh.gif)

The pitfall of these rather optimistic speedup and efficiency evaluations is related to the fact that, as the problem size increases, communication costs will increase, but increases in communication costs are not accounted for by Gustafson’s law.

Scalability: Suppose we have fixed size processes and fixed size inputs in a parallel program and we get some certain amount of efficiency in this case. When we increase the amount of problem(processes and input), if we can get same amount of efficiency, than we can say that this parrallel program can be scalable.