Performance Analysis for Parallel Applications

Wei Wang

Achieving Desired Performance

- Ideally, parallel speedup should be linear
  - i.e., using *n* processors => *n* times faster than using 1 processor
- Reality is stark
  - Seldom do we have linear speedup
  - Sometimes using more processors means slow down
- It is important to be able to understand and analyze performance issues of parallel applications

Theoretical Performance Models

Theoretical Performance Models

- Theoretical performance models improve basic understandings on parallel computing
  - They also define the maximum and minimum possible speedups
- Theoretical models
  - Amdahl’s Law
  - Gustafson's law
  - Karp–Flatt metric
  - Speedup model with communication cost

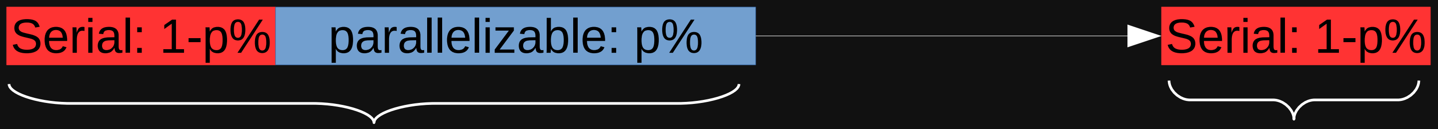
The Basic Speedup Definition

- Speedup of a parallel application is defined as

$$Speedup = \frac{\text{Sequential Execution Time}}{\text{Parallel Execution Time}}$$

Amdahl’s Law

- Intuition of Amdahl’s law:
  - If an application has a *p%* parallelizable part and a *(1-p%)* serial part,
    - the minimum execution time of the application is the execution time of the serial part
    - The maximum speedup is bounded by the serial part.



Original Execution Time

Minimum execution time.
Given enough parallel processors,
The paralleled-able part’s execution time may approach 0.

Amdahl's Law: Equation for Speedup

- The equation of Amdahl's Law:

$$Speedup = \frac{\text{Sequential Exec Time}}{\text{Parallel Exec Time}} = \frac{1 - p + p}{\left(1 - p\%\right) + \frac{p\%}{s}} = \frac{1}{\left(1 - p\%\right) + \frac{p\%}{s}}$$

  - *Speedup* is the overall speedup of the whole application after parallelization
  - *p%* is the percentage of the execution time of the parallelizable part
  - *s* is the speedup of the parallel part

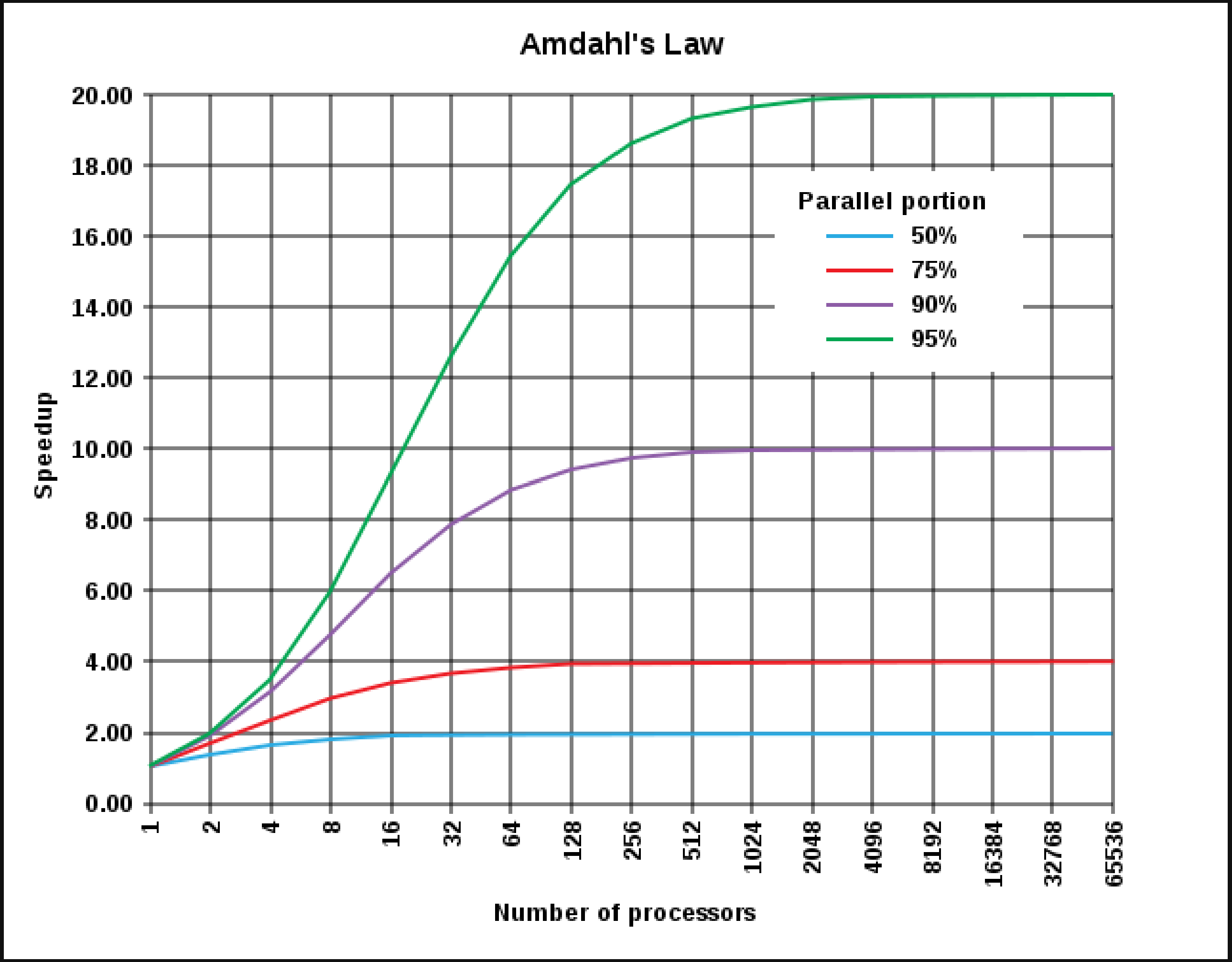
Amdahl's Law: The Limit on Speedup

- If the parallel part speedup approaches infinity:

$$\lim_{(s \to \infty)} Speedup = \frac{1}{1 - p\% + \frac{p\%}{s}} = \frac{1}{1 - p\%}$$

  - i.e., the maximum speedup is bounded by the execution time of the serial part

Another Visualization of Amdahl's Law



* Figure by Daniels220 at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6678551

Karp-Flatt Metric

- In practice, the ratio of serial is usually hard to know
- However,
  - *Speedup* can be easily measured
  - Parallel part speedup, *s*, is usually assumed to be the same as number of processors
- If both Speedup and s are known, then we can estimate the ratio of the serial part with Amdahl's law equation
- Karp-Flatt metric, *e*, is the name of the "serial part ratio" estimated based on experimentally measured *Speedup*

$$e = 1 - p\% = \frac{\frac{1}{Speedup} - \frac{1}{s}}{1 - \frac{1}{s}}$$

The Limitations of Amdahl’s Law and Karp-Flatt Metric

- Amdahl’s law defines the performance limit of a parallel application, with
  - A fixed problem size, i.e., the ratios of the parallel part and serial part are fixed.
    - In real life, problem size usually grows with system size
    - parallel part usually grows faster than serial part
    - This limitation is addressed by Gustafson’s law
- Amdahl’s law does not provide any insights about parallel overhead

Gustafson's Law

- Gustafson’s Law considers the growth of problem size
- Intuitively, Gustafson’s Law says that
  - if we can indefinitely scale up the problem size (and its parallelizable) with the increase of the processor count, the overall speedup of the whole application is the same as the speedup of the parallelizable part.
- Gustafson’s law reveals the true power of parallelization: we can solve much bigger problem within reasonable amount of time

Deriving Gustafson's Law

- Let (similar to Amdahl's law),
  - *p%* be the percentage of the execution time of the original workload size
  - *s* be the speedup of the parallel part
  - W be the original problem size on one processor
- With more processors, increase the workload size to

$$new\_workload = (1 - p\%) \times W + (s \times p\%) \times W$$

- The speed for the new workload is then

$$Speedup = \frac{\frac{(1-p\%)\times W}{1} + \frac{(s\times p\%)\times W}{1}}{\frac{(1-p\%)\times W}{1} + \frac{(s\times p\%)\times W}{s}} = 1 - p\% + s \times p\%$$

Maximum Speedup By Gustafson's Law

- If the parallelizable part is indefinitely large

$$\lim_{(p \to 1)} Speedp = 1 - p\% + s \times p\% = s$$

  - i.e., if the problem size is large enough, especially, if the parallelizable part is large enough, the overall speedup is the same as the speedup of parallelizable part

The Limitations of Gustafson's Law

- The sizes of some problems are limited.
  - e.g., the size of a chess game at a certain stage.
  - Essentially, these applications do not benefit much from large scale parallelization.
- Similar to Amdahl’s law, Gustafson’s law does not provide any insights about parallel overhead

Speedup Equation with Communication Cost

- Probably originated from Michael Quinn’s Parallel Computing text book
- Let *p%* be the percentage of the execution time of the original workload size
- Sequential execution time is (still),

$$\text{Sequential Execution Time} = 1 - p\% + p\% = 1$$

Speedup Equation with Communication Cost

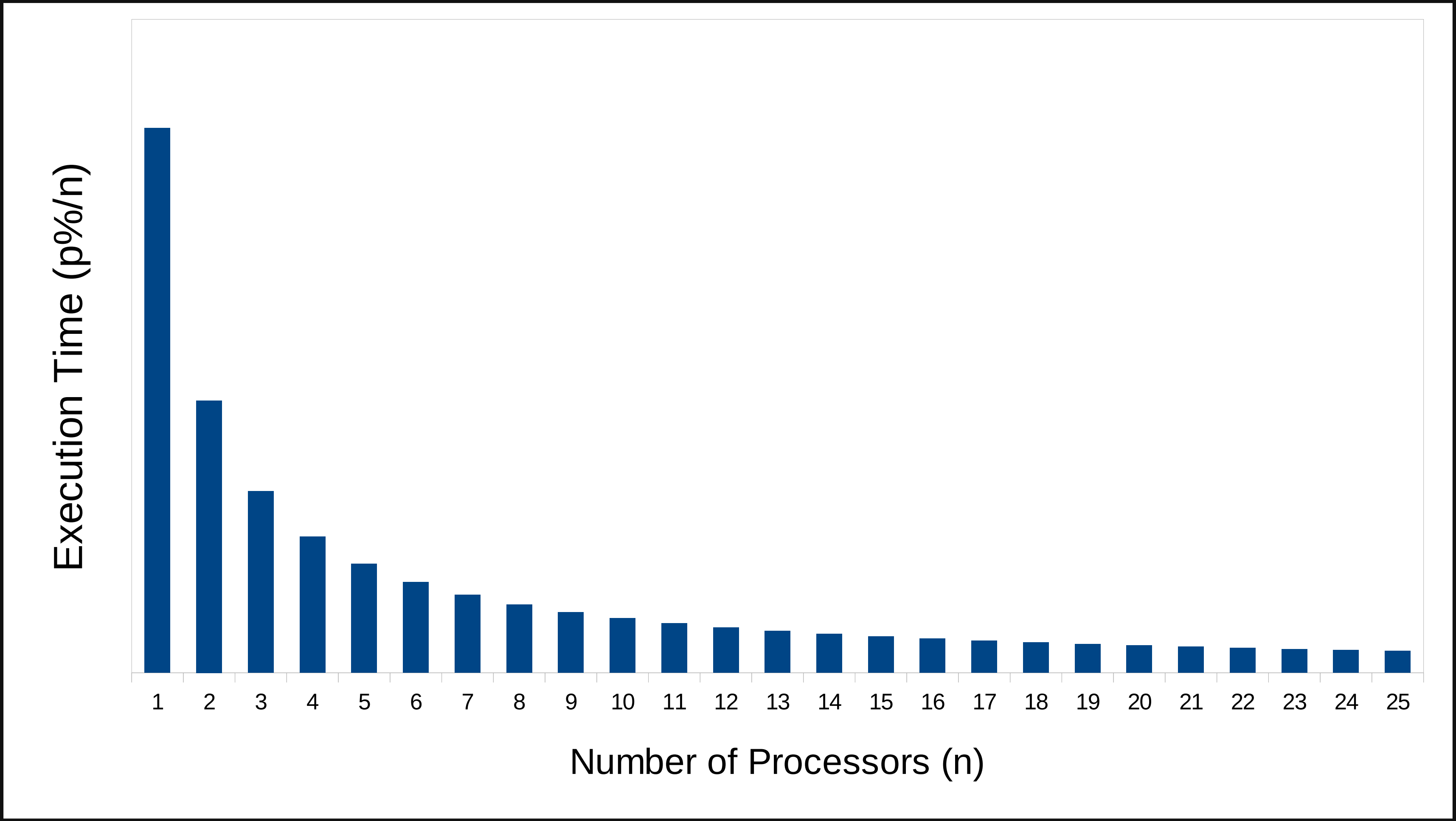
- Let
  - _n_ be the number of processors
  - _k(n)_ be the cost of communication on _n_ processors
- The parallel execution time is then (assuming the parallel computation has linear speedup)

$$\text{Parallel Execution Time} = 1 - p\% + \frac{p\%}{n} + k(n)$$

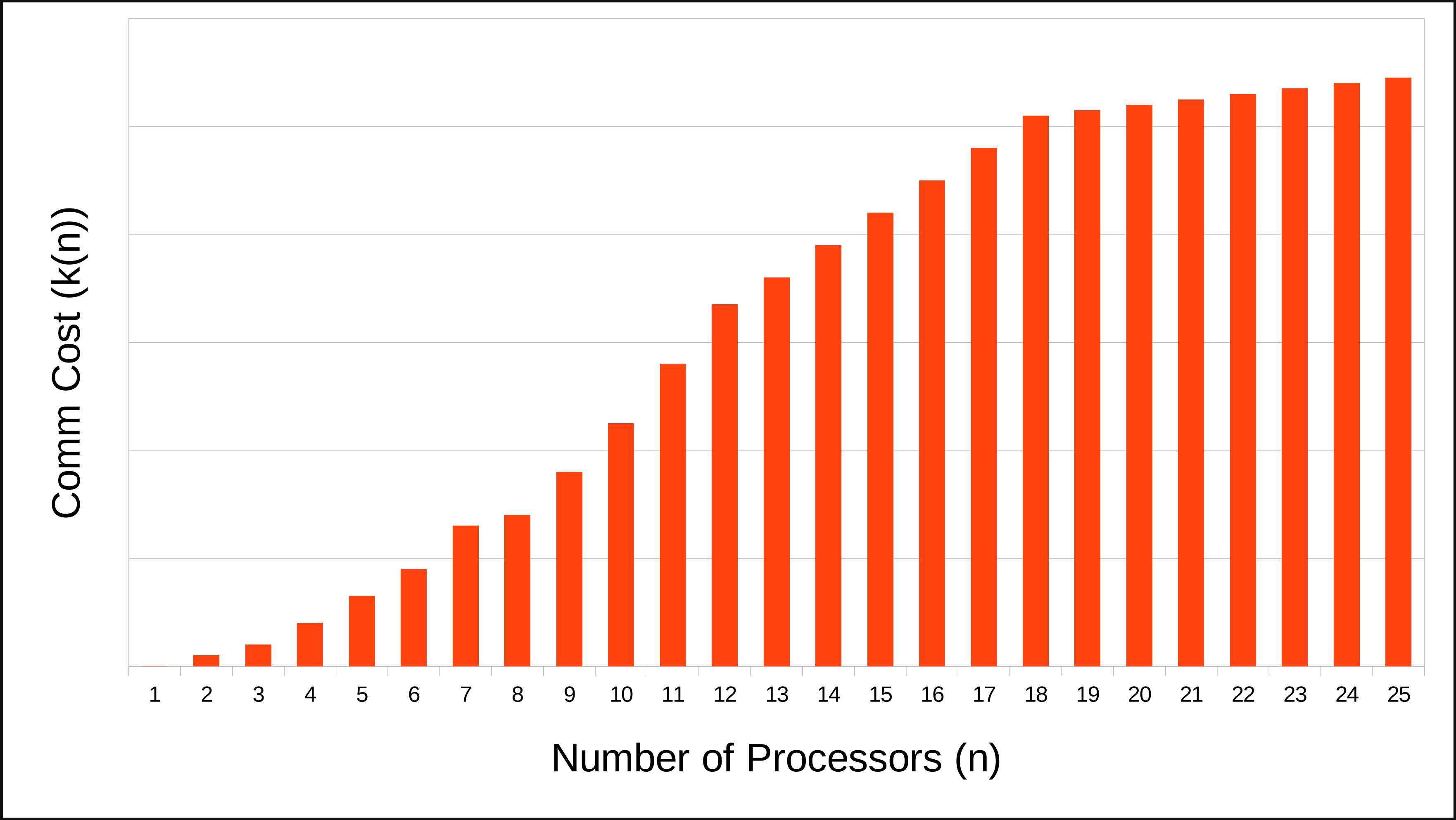
- Overall speedup is

$$Speedup = \frac{\text{Sequential Exec Time}}{\text{Parallel Exec Time}} = \frac{1}{1 - p\% + \frac{p\%}{n} + k(n)}$$

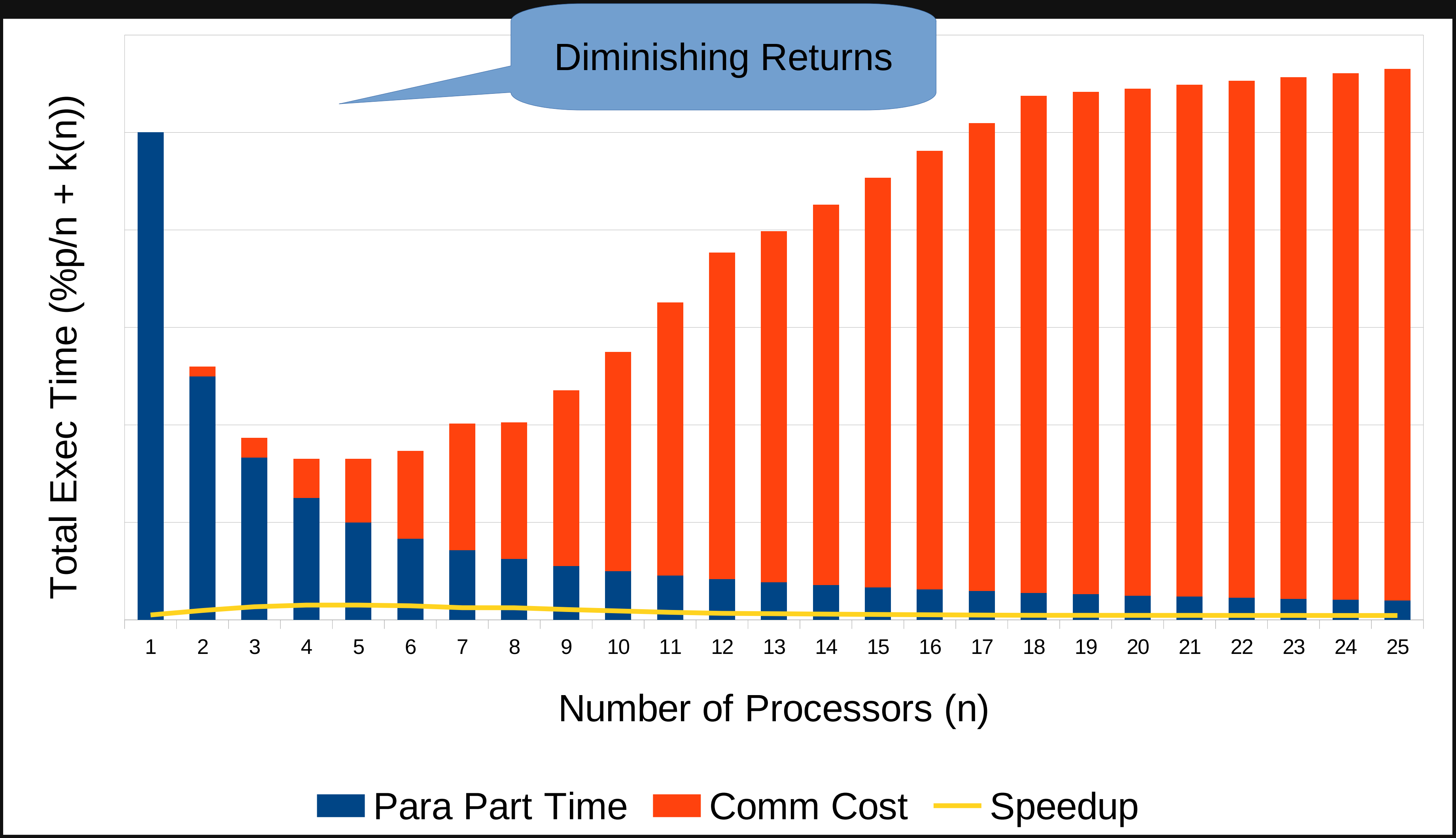
Plotting the Execution Time of the Parallel Part (p%/n)



Plotting the Communication Cost *k(n)*



Plotting the Parallel Execution time (%p/n + k(n)) and Speedup



Real Speedup with Communication Cost

- Due to the increasing communication cost,
  - real speedup is usually sub-linear
  - Speedup would reach the maximum with certain number of processors and adding more cores would only reduce speedup

Super-linear Speedup

- It is impossible to get super-linear speedup with our current theoretical models.
- However, super-linear speedup is occasionally possible in practice (very rare)
  - Super-linear speedup is usually caused by improved utilization and/or size of memory resources, such as cache or DRAMs
  - Super-linear is also possible in some algorithms employing exploratory decomposition