Performance Evaluation

"The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times;

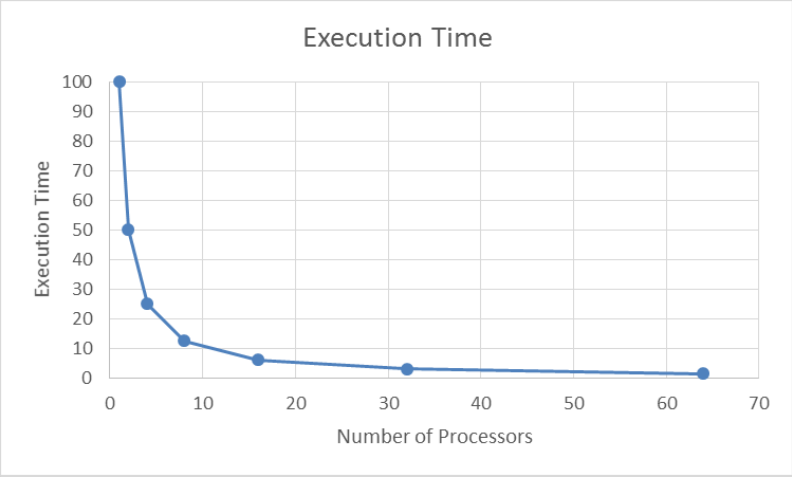
premature optimization is the root of all evil (or at least most of it) in programming" การพยายาม optimize เร็วเกินไป เป็นสาเหตุของความชั่วร้ายในการเขียนโปรแกรม

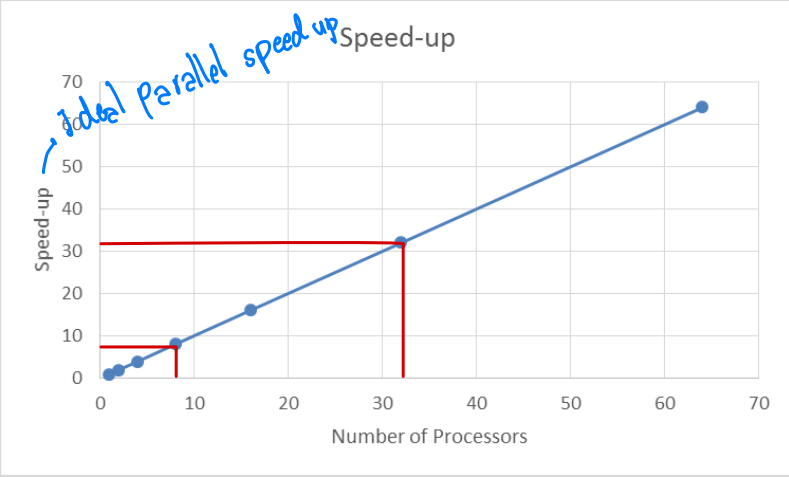
--- Donald Knuth, 1974 Knuth บอกว่าเขียนโปรแกรมให้ถูกต้อง เข้าใจง่ายก่อน ค่อย optimized ทีหลัง

Speed-up

- Speed-up determines how much faster parallel execution is versus serial execution. ค.เร็วที่มากขึ้น เมื่อเทียบกับตอนยังไม่ปรับปรุงโปรแกรม
- Speed-up = original run time / run time after change เวลาของ code ที่ปรับปรุง
  เวลา code เดิม
- Parallel Speed-up = serial run time / parallel run time
- Ideal parallel speed-up = number of processors

maximum speed up ที่เป็นไปได้ของ Parallel computing = จ.น. processor เช่น มี 10 processors ก็เร็วขึ้น 10 เท่า





Note: When computing speed-up, the best and fastest serial algorithm & code must be used for measuring serial runtime. Do not use the parallel version with 1 thread.

Parallel Efficiency

- Parallel efficiency indicates how well software utilizes the computational resources of the system.
- Parallel efficiency = speed-up / number of processors
- Maximum efficiency = 1 or 100%
- Ex. 12x speedup on 16 cores → 75% efficiency

This means that, on average, each core is idle about 25% of the time.

Parallel Performance Limitation

สาเหตุที่ efficiency ไม่ได้เท่ากับ ideal

- Unparallelizable computation (inherently serial computation)
- Parallel computing overhead
  - Thread creation & scheduling
  - Communication & synchronization
- Load balancing

การคำนวณที่ทำ parallel ไม่ได้ เช่น ติด dependencies

การสร้างและสลับการทำงานของ thread มี overhead

overhead การสื่อสารส่งข้อมูล

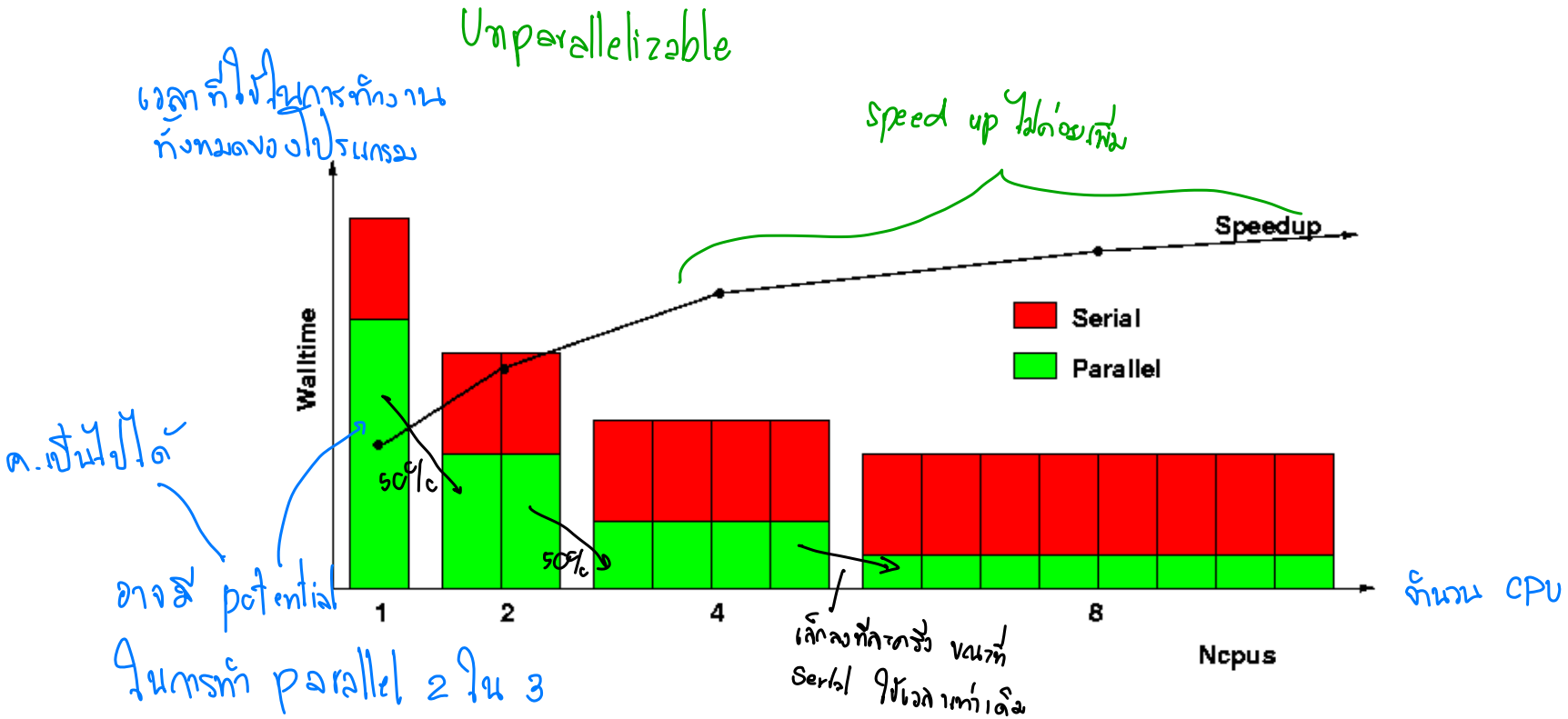
การกระจายงานที่ไม่เท่ากัน งานย่อยๆ มีขนาดเล็กใหญ่ไม่เท่ากัน

CPU ที่รับงานเล็กๆ ทำเสร็จเร็วก็ไม่มีอะไรทำ (idle)

เวลาที่ใช้จริงๆ ว่าทำงานเสร็จเมื่อไหร่คือ processor ตัวสุดท้ายทำงานเสร็จ (งานใหญ่สุด)

เวลาแทนที่ควรใช้กับเวลาจริง มันต่างไปที่ไม่เกี่ยวข้องกับการคำนวณ = overhead

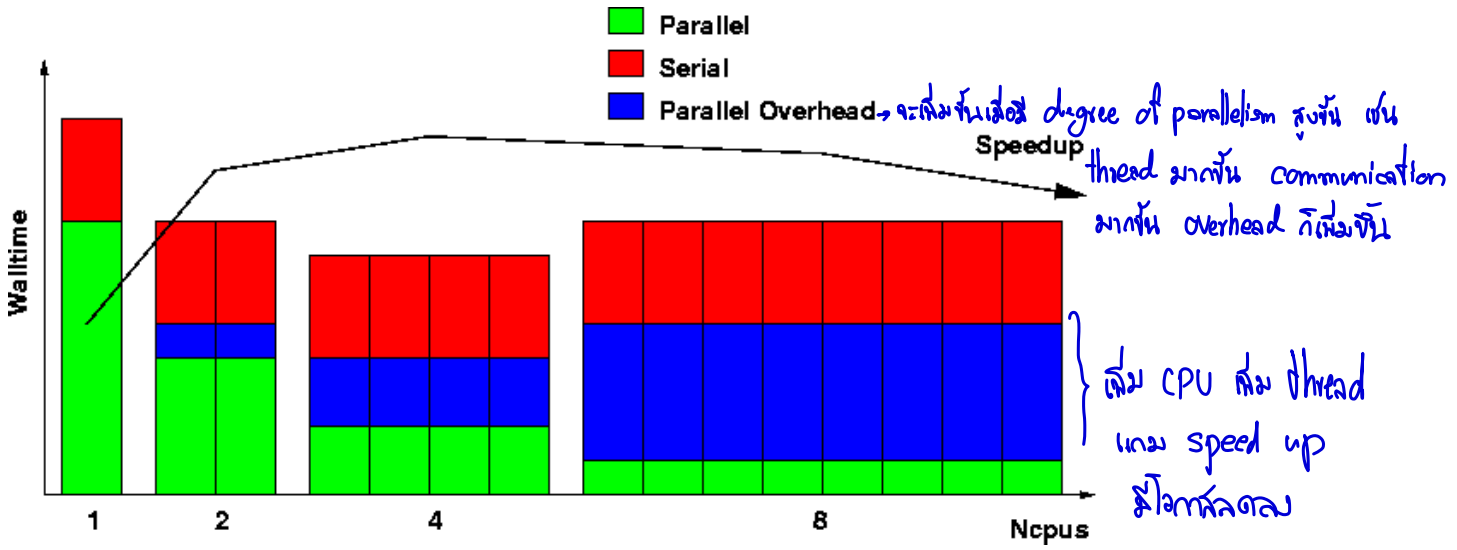
Realistic Speedup



- Inherently serial computation limits speedup gain.

Realistic Speedup

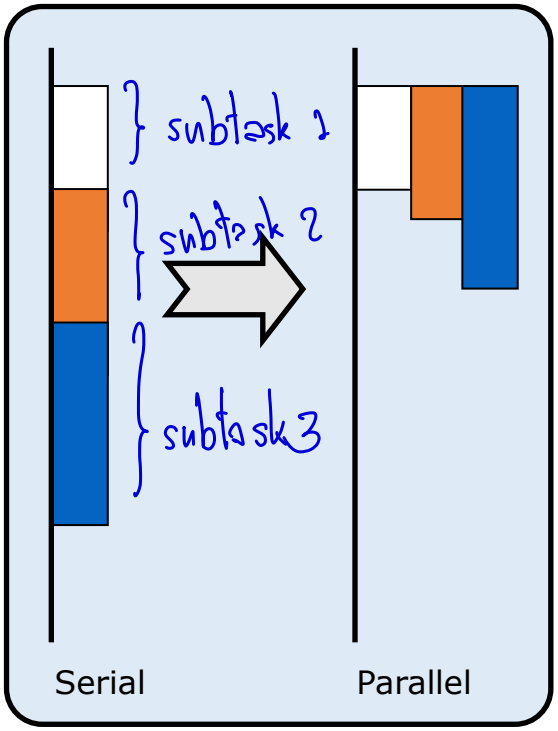
Overhead

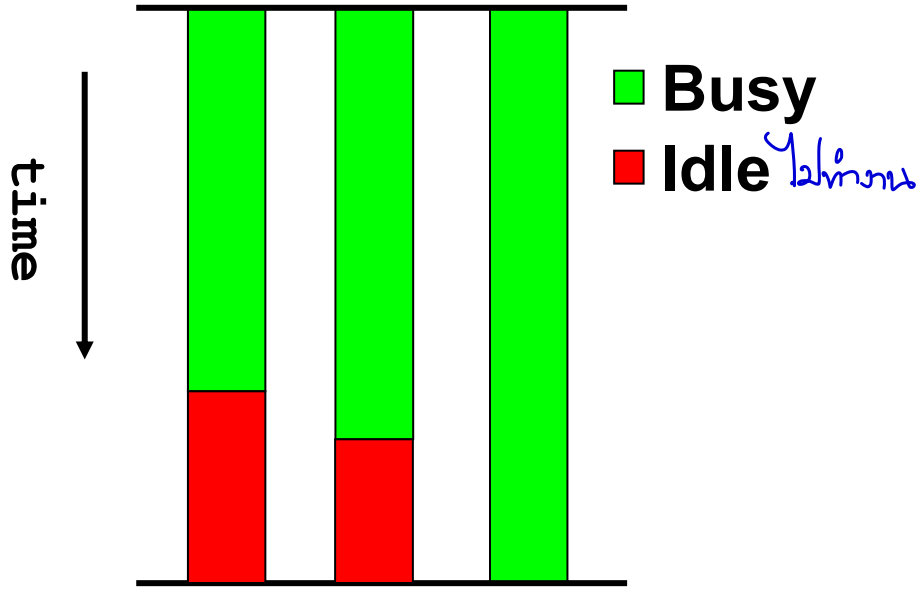


- Overhead can even reduce speedup when adding more processors.

Load Imbalance

- Unequal work loads lead to idle processors and wasted time.





Fix Load Model

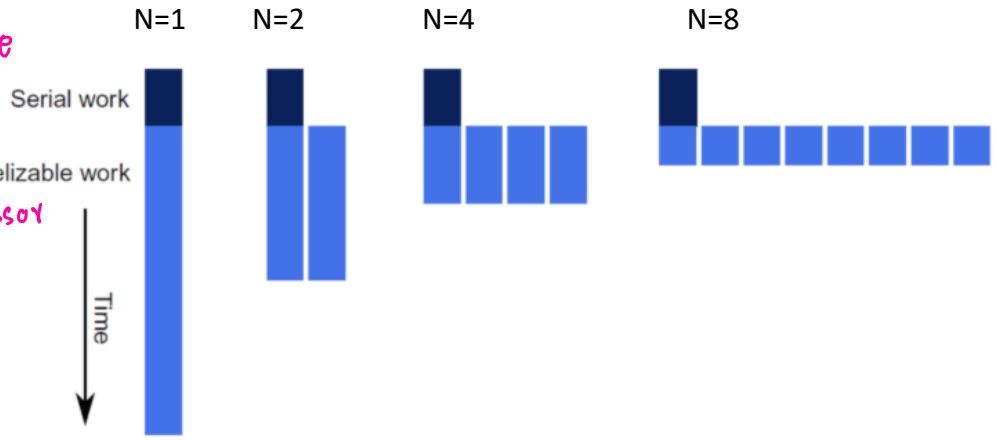
Amdahl's Law

พิจารณาเฉพาะของ Unparallelizable code

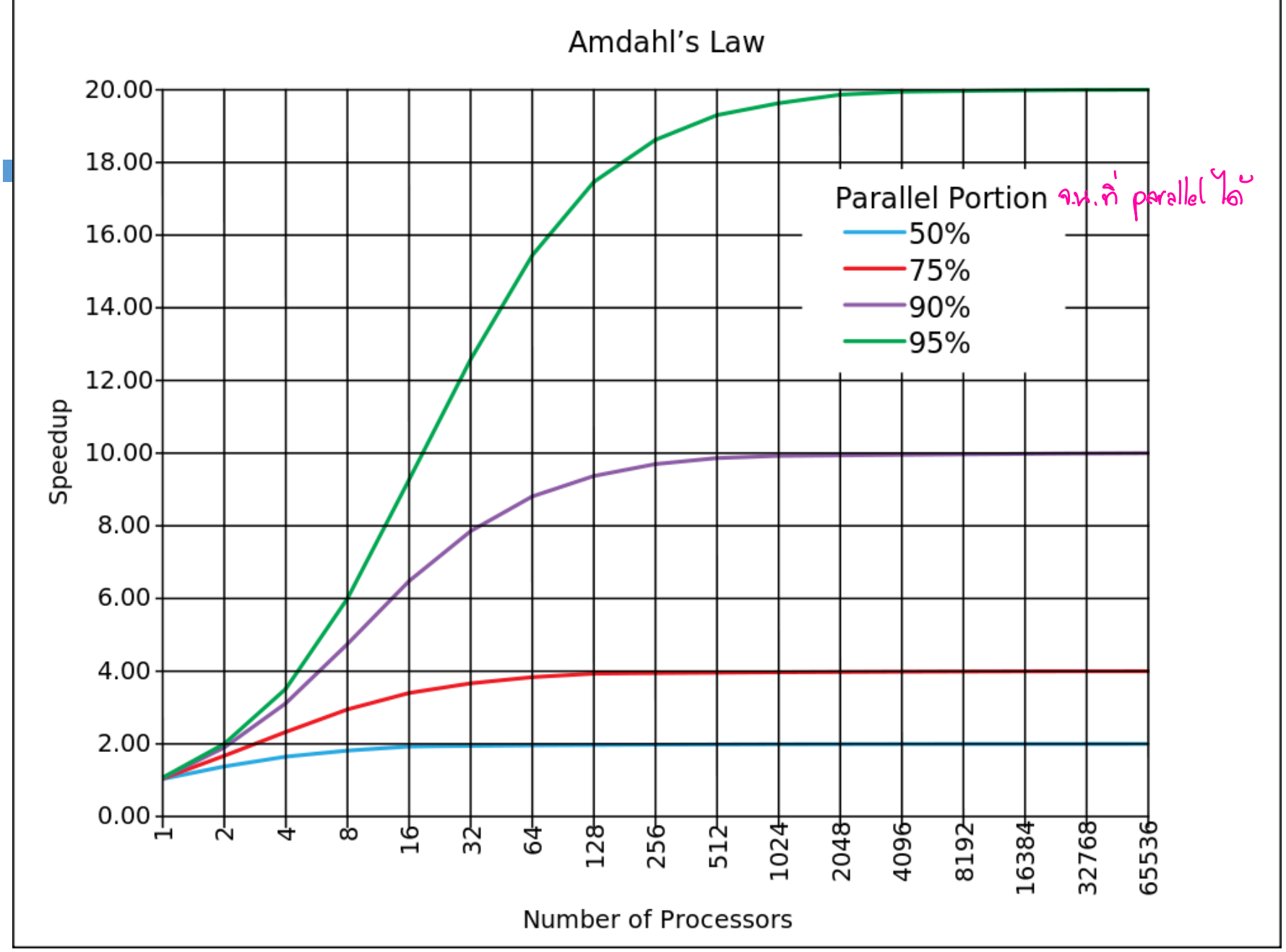
Let s be fraction of sequential code, p be fraction of parallelizable code $(s + p = 1)$ พยายามหาคำนวณ speed up จากสัดส่วนว่าโปรแกรมสามารถทำ parallel ได้มากน้อยแค่ไหน

- Speedup $= \frac{s+p}{s+\frac{p}{N}}$ sequential time / parallel time (จ.น. processor)

$$= \frac{1}{s+\frac{(1-s)}{N}}$$



- If N approaches infinite then speed-up approaches $1/s$ and efficiency approaches 0. จ.น. processor เยอะขึ้น ทำให้ speed up เข้าใกล้ $\frac{1}{s}$ และประสิทธิภาพเข้าใกล้ 0
- Amdahl's Law implies that parallel computing is only useful with small number of processors or the problem can be perfectly parallelized.



จำนวนที่ parallel ได้

Source: http://en.wikipedia.org/wiki/Amdahl's_law

Gustafson's Law

Usually, we add more processors to solve larger problems.

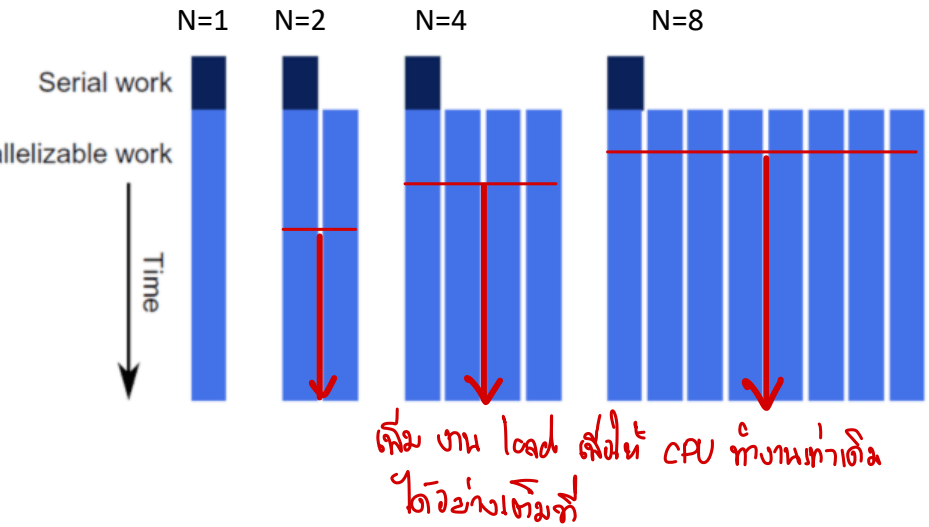
So, let the problem size scale with the number of processors and fix the time.

Gustafson's speed-up measures how much more work can be done on a parallel machine compared to a sequential machine.

$$\text{Speedup} = \frac{s+pN}{s+p}$$

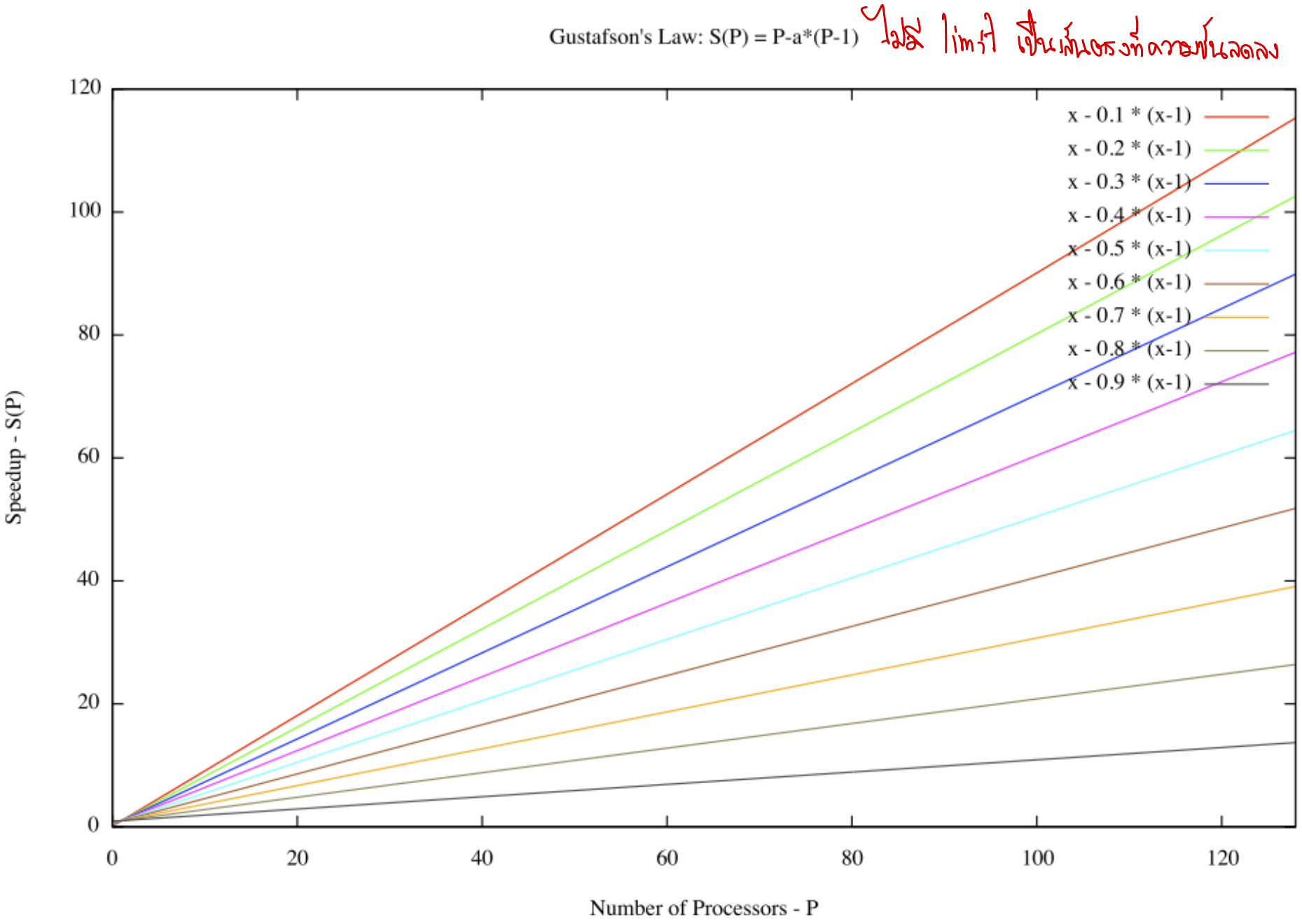
$$= s + (1-s)N$$

$$\text{Efficiency} = \frac{s+(1-s)N}{N}$$



Scaled speed-up has no upper-bound.

If N is very large then efficiency approaches $(1-s)$ or fraction of parallel code.



Source: http://en.wikipedia.org/wiki/Gustafson's_Law

Super-linear Speedup

speed up > processor

- When speedup is greater than number of processors.
- Possible cases: สาเหตุ
  - Smaller subtasks fit better in cache lines/memory pages and therefore reduce misses. ขนาดของงานที่เล็กลงบางครั้งใส่ลงใน cache ได้พอดี (ลด cache miss)
  - Parallel execution has more cache memory to use and share. มี cache มากขึ้น
  - Parallel algorithm reduce amount of computation, e.g. parallel search finds solution faster. algorithm บางครั้งทำให้โปรแกรมจบลงเร็วขึ้น

- Caution: Data sets that are too small – smaller than typical data set size – can give a false sense of performance improvement. ชุดข้อมูลเล็กเกินไป บางทีจะได้ผลลัพธ์ที่ mislead (เข้าใจผิด)

parallel search แบ่งหนังสือ 10 ส่วนให้ 10 คนไป search แล้วปรากฏว่าหน้าที่ต้องการหาอยู่ในหน้าแรกของท่อนที่ 10 process ที่ทำงานท่อนที่ 10 ก็จะจบการทำงานเร็วกว่า 10 เท่า (speed up > 10)

Performance Testing

- Load test
  - Test how system performs under expected load.
- Stress test
  - Test how system reacts under extreme load. At what point the system break and where it breaks.
- Endurance test
  - Test how system behaves when it is used at expected load for a long period of time.
  - Some problems appear only after long running time, e.g. memory leak, unclosed resources/connections.