CS429: Computer Organization and Architecture
Amdahl's Law

Dr. Bill Young
Department of Computer Science
University of Texas at Austin

Last updated: May 1, 2019 at 12:48

Performance Example

Exercise: You have a task X with two component parts A and B, each of which takes 30 minutes. What is the latency of X?

Suppose that you can speedup part B by a factor of 2. What is the latency now? What is the overall speedup?

Performance Example

Exercise: You have a task X with two component parts A and B, each of which takes 30 minutes. What is the latency of X?

Suppose that you can speedup part B by a factor of 2. What is the latency now? What is the overall speedup?

Answer: It really depends on whether A and B are *concurrent* or *sequential*. If sequential, you can answer using **Amdahl's Law**.

Two Notions of Performance

Plane	DC to Paris	Speed	Passengers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

- **Which has higher performance?**
- **What is performance?**
  - Time to completion (latency)?
  - Throughput?
- We're concerned with performance, but there are other important metrics:
  - Cost
  - Power
  - Footprint

Amdahl's Law

How much extra performance can you get if you speed up *part* of your program? There are two factors:

- How much better is it? (S, k)
- How often is it used? (α)

$$T_{new} = (1-\alpha)T_{old} + (\alpha T_{old})/k = T_{old}[(1-\alpha)+\alpha/k]$$

$$S = \frac{T_{old}}{T_{new}} = \frac{1}{(1-\alpha)+\alpha/k}$$

Example 1

Your pipeline has one very slow stage that consumes 60% of the time to process an instruction. You discover that you can speed it up by a factor of 3. What is the improvement in the latency?

$$S = \frac{T_{old}}{T_{new}} = \frac{1}{(1-\alpha)+\alpha/k}$$

Hence,

$$S = \frac{1}{(1-0.6)+0.6/3} = 1.67X$$

Suppose you could make that stage arbitrarily fast. How much would that improve latency?

Example 2

Suppose:

- Floating point instructions could be improved by $2X$.
- But, only 10% of instructions are floating point.

$$T_{new} = T_{old} * [0.9 + 0.1/2] = 0.95 * T_{old}$$

$$S_{total} = \frac{1}{0.95} = 1.053$$

Speedup is bounded by:

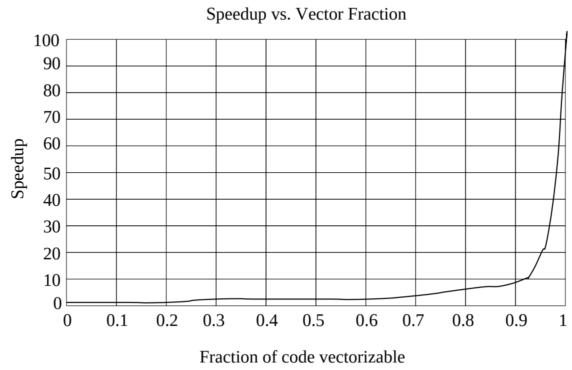
$$\frac{1}{\text{fraction of time not enhanced}}$$

Example 3

Assume you can parallelize some portion of your program *to make it 100X faster.*

How much faster does the whole program get?

$$T_1 = T_0\left[(1-p)+\frac{p}{S}\right]$$



Example 4

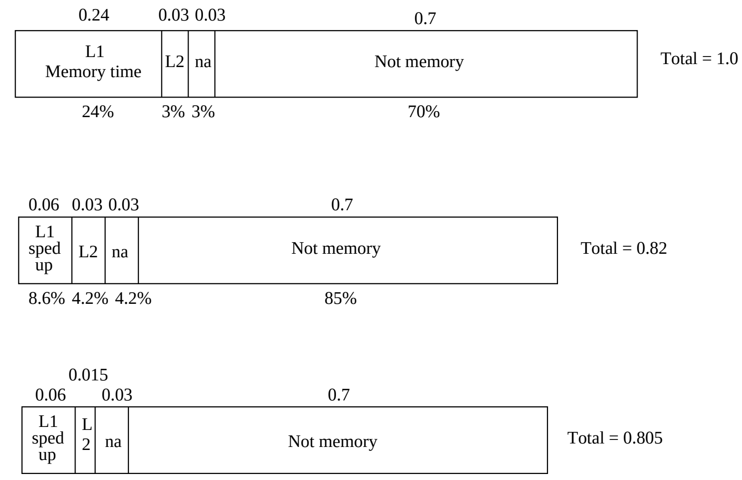
Suppose:

- Memory operations currently take 30% of execution time.
- A new L1 cache speeds up 80% of memory operations by a factor of 4.
- A second new L2 cache speeds up 1/2 of the remaining 20% by a factor of 2.

What is the total speedup?

Example 4 Answer

Applying the two optimizations sequentially:



Speed up = 1.242

Summary Message

Make the common case fast!

Examples

- All instructions require instruction fetch, only some require data memory access. *Improve instruction fetch performance first.*
- Programs exhibit locality (spatial and temporal) and smaller memories are faster than larger memories.
  - Incorporate small, fast caches into processor design.
  - Manage caches to exploit locality.

Amdahl provided a quantitative basis for making these decisions.

Amdahl's Non-Corollary

Amdahl's law does not bound slowdown!

Things can only get so fast, but they can get arbitrarily slow.

Don't do things that hurt the non-common case too much!