



# Reader Guide

## A Note on Format

Material laid out in a visual presentation slide format with active learning parts and room to add notes and work out exercises.

## Margin Text

-  The 12-point text on the side is for reading along from a handout in class or later study.
-  Intended to reduce the talk parts of class presentation for a more engaging experience.

## Quiz







Pause and ask.

## Exercise

A little work to find out or try something along the way (or later if marked homework).

## Iconography

Less wordy, draw attention.

-  Notice (sometimes "Note that...")
-  Key idea
-  See reference/textbook
-  Homework
-  Think (hmm)
-  Lookup

# Enhancing A Part

Parts of a program are enhanced (affected times marked gray).

Before

Enhanced times



How much improvement in execution time should be expected?

After

Only gray times become shorter

Conventionally, put bigger number in numerator to get non-fractional (>1) valued speedup ratio.

$$Speedup_{overall} \stackrel{\text{def}}{=} \frac{(exec\ time)_{old}}{(exec\ time)_{new}} = ?$$

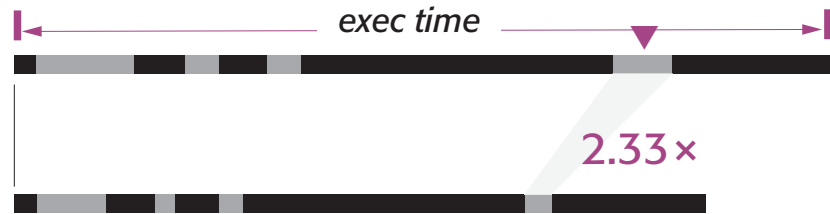
# Enhancing a Part Elaboration

The timings could come from software or other systems in general. Amdahl was concerned originally with the effects of adding hardware parallelism.

Enhanced times could be due to improved code sections or corresponding circuitry.





The after enhancement time bar is  $\approx 0.85$  of the before one in length (assume relatively to scale).

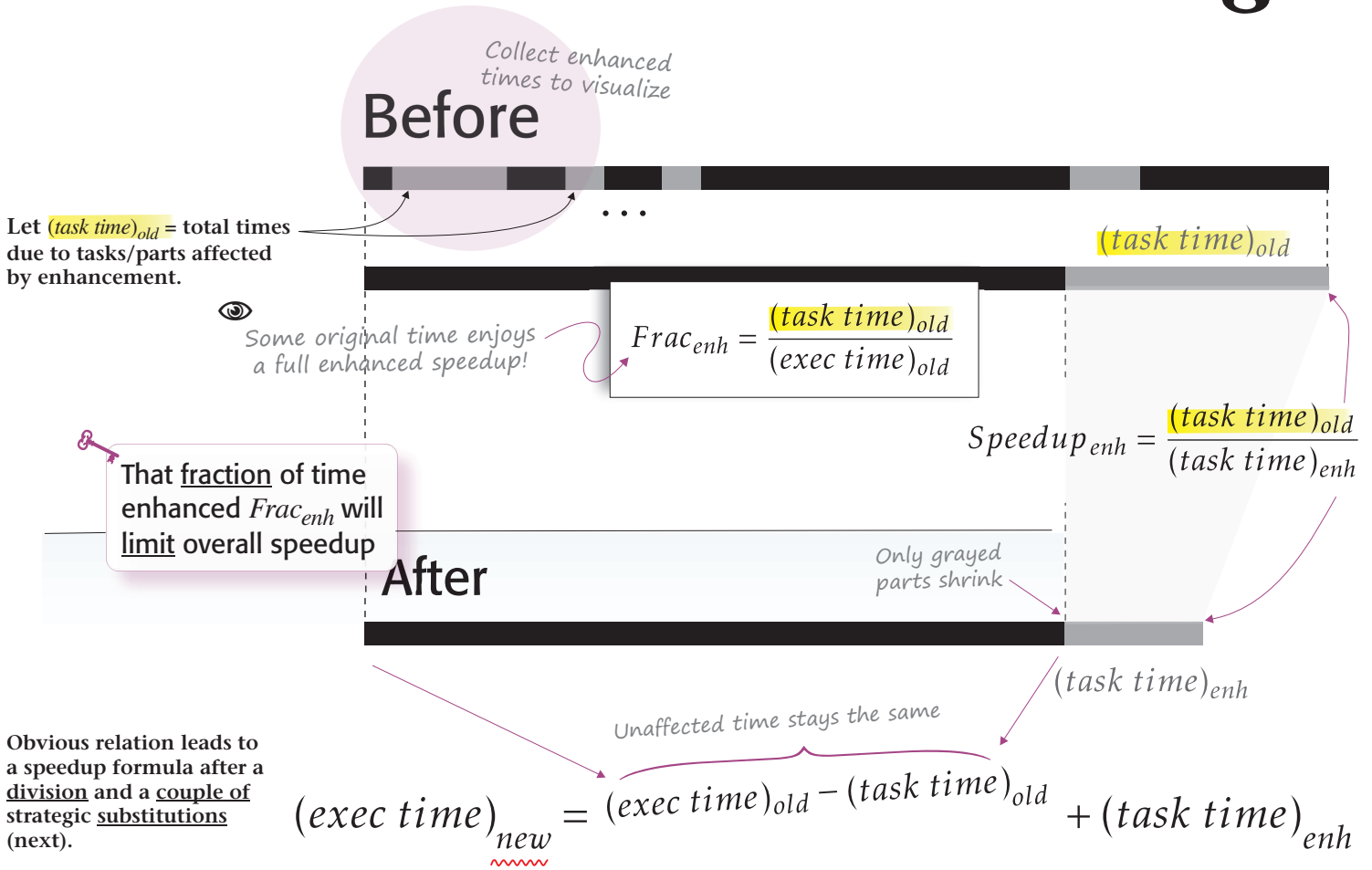


## Mixed improvement

Lengths of time bars in the figure suggest an overall speedup of  $1.18\times$  ( $.85^{-1} \approx 1.18$ ), even though the last enhanced time (marked) seems to enjoy more than a twofold speedup (233%).

-  More than  $2\times$  faster sometimes
-  An overall 18% speedup (only!)

# Amdahl's Insight



# Amdahl's Insight A Formula



Two substitutions unravel the formula.

$$(exec\ time)_{new} = (exec\ time)_{old} - (task\ time)_{old} + (task\ time)_{enh}$$

$$\div (exec\ time)_{old}$$

To obtain an overall speedup term in left-hand side  
(in inverted form, flip back at the end)

and note

$$\frac{(task\ time)_{old}}{(exec\ time)_{old}} = Frac_{enh}$$

also observe (from previous slide)

$$\frac{Frac_{enh}}{Speedup_{enh}} = \frac{(task\ time)_{old}}{(exec\ time)_{old}} \times \frac{(task\ time)_{enh}}{(task\ time)_{old}}$$

another handy substitution for 3rd right-hand term after the division

**Exercise**  
Complete the derivation leading to the formula (next). Ans. last slide.

# Amdahl's Law

$$\frac{(exec\ time)_{old}}{(exec\ time)_{new}}$$



## Exercise

Plot the speedup, overall vs. enhanced (e.g., 2 to 200 times) for different values of  $Frac_{enh}$  (e.g., 0.9, 0.8, ..., 0.4). Comment on the shapes of the curves.

$$Speedup_{overall} = \frac{1}{(1 - Frac_{enh}) + \frac{Frac_{enh}}{Speedup_{enh}}}$$

## CA:AQA Example 1

Suppose an **enhancement runs 10 times faster** than the original machine but is only **usable 40% of the time**. What is the overall speedup gained by incorporating the enhancement?

# Amdahl's Law

## CA:AQA Example 2

**Hint.** Figure out formula variables from problem wording (sometimes you need a small calculation to get those vars).

Suppose a cache is 10 times faster than main memory, and suppose that the cache can be used 90% of the time. How much speedup do we gain by using the cache?

# Amdahl's Law Example 3

 P&H (p. 396)

Multiple-issue code scheduling example (ignore grayed parts here): 1.25x vs. scalar, based on comparing CPI/IPC.



Instructions reordered to eliminate dependencies detrimental to the assumed pipelined execution mode.

```

Loop:
lw   $t0,0($s1)
addu $t0,$t0,$s2
sw   $t0,0($s1)
addi $s1,$s1,-4
bne  $s1,$0,Loop
    
```

Issue Slot 1	Slot 2	
	lw \$t0,0(\$s1)	CC1
addi \$s1,\$s1,-4		CC2
addu \$t0,\$t0,\$s2		CC3
bne \$s1,\$0,Loop	sw \$t0,4(\$s1)	CC4

? Speedup

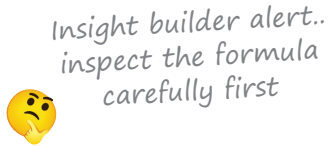
not a typo

Given essential info about the enhanced schedule, a simple application of Amdahl's law yields the answer directly.

**Quiz**  
Identify the formula variables. **Hint:** only 2 instrs enjoy a full speedup.

A scalar processor can run a 5-instruction loop iteration in 5 clock cycles. Adding hardware to issue an ALU instruction and a load-store in the same cycle could double its peak throughput. What's the speedup if we could only schedule two of those in one cycle?

# Amdahl's Law Example 4



**Suppose a program runs in 100 seconds, with division operations responsible for 80 seconds of this time. How much would those ops need to be improved to make the program run five times faster?**

# Amdahl's Insight Elaboration



## Quiz

Find the limit (fig). Interpret result.

Amdahl demonstrated in a seminal work that the amount of exploitable parallelism in workloads is a fundamental limiting factor on performance.

For example, can't add a 2nd processor and expect a 2-fold improvement in runtime unless it is usable 100% of the time (i.e.,  $Frac_{enh} = 1$ , which leads to an overall speedup of 2).

Limit reveals that the fraction of time not enhanced places an upper limit on overall speedup.

Why expend resources on a big improvement if that time (hidden behind  $Frac_{enh}$ ) were small? Unless it comes really cheap?

$$Speedup_{overall} = \frac{1}{(1 - Frac_{enh}) + \frac{Frac_{enh}}{Speedup_{enh}}}$$

$Speedup_{enh} \rightarrow \infty$

⇒ **An overall ceiling**

⇒ **(task time)<sub>old</sub> cost tradeoff**

Main info/factor to weigh against resources to commit to enhancement

# References

## Exercise

(Graduate) Check the derivation of Amdahl's law in the original 1967 paper (!)



- ➔ Gene M. Amdahl, *Validity of the single processor approach to achieving large scale computing capabilities*, Proceeding AFIPS '67 (Spring) April 18–20, 1967, Spring Joint Computer Conference, pages 483–485.

<https://dl.acm.org/doi/10.1145/1465482.1465560>

<https://ieeexplore.ieee.org/document/4785615>

- ➔ Reprint 2000 in Hill, Jouppi, Sohi, *Readings in Computer Architecture*, (The Morgan Kaufmann Series in Computer Architecture and Design).

- ➔ Also in IEEE Solid State Circuits Society Newsletter (SSCS NEWS Summer 2007), vol. 12, issue 3. <https://www.scribd.com/document/620606754/Amdahl-1967>

- ➔ [CA:AQA] Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, Morgan-Kaufmann (ed. 2).

- ➔ [P&H] Patterson and Hennessy, *Computer Organization & Design: the Hardware/Software Interface*, MK/Elsevier (revised 4 or later).

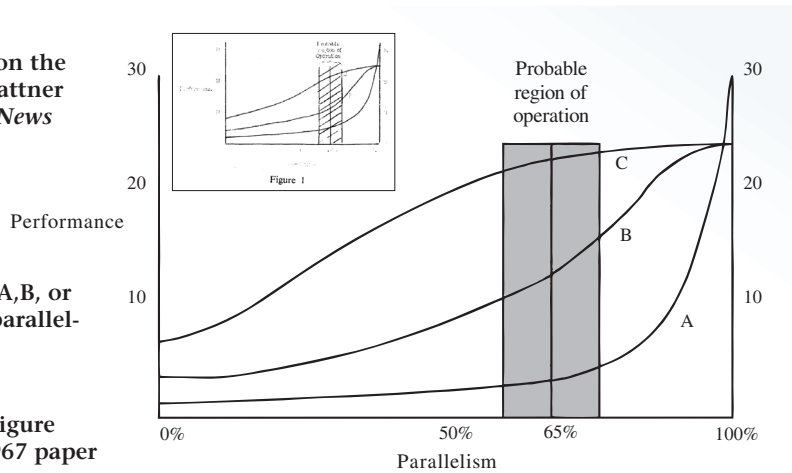


See commentary on the figure by Justin Rattner in the *IEEE SCS News* reprint.

## Quiz

Which machine (A,B, or C) had the most parallelism?



Reproduction of figure from Amdahl's 1967 paper (see SVG link).



[https://hashimi.ws/ca/svg/amdahl\\_1967\\_fig.svg](https://hashimi.ws/ca/svg/amdahl_1967_fig.svg)

# A Modern Application

**A main argument** refs below

-  Small per case payoff, seemingly
-  Significant part of a workload

## References

Salem Alsari & Muhammad Al-Hashimi, *“Investigation of Energy and Power Characteristics of Various Matrix Multiplication Algorithms”*, Energies 2024. <https://doi.org/10.3390/en17092225>

Othman Alamoudi & Muhammad Al-Hashimi, *“On the Energy Behaviors of the Bellman–Ford and Dijkstra Algorithms: A Detailed Empirical Study”*, J. Sens. Actuator Netw. 2024. <https://doi.org/10.3390/jsan13050067>

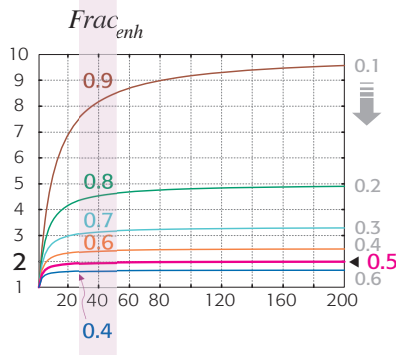
# Some Answers

$$\left(\frac{\text{exec time}_{new}}{\text{exec time}_{old}}\right) = 1 - \frac{\text{Frac}_{enh}}{\text{Speedup}_{enh}} + \frac{\text{Frac}_{enh}}{\text{Speedup}_{enh}}$$

$$\text{Speedup}_{overall} = \frac{\text{exec time}_{old}}{\text{exec time}_{new}} = \frac{1}{(1 - \text{Frac}_{enh}) + \frac{\text{Frac}_{enh}}{\text{Speedup}_{enh}}}$$

Curves show diminishing returns as part enhancement increases.  
Overall speedup ceiling goes down as portion of time unaffected (in gray) increases.

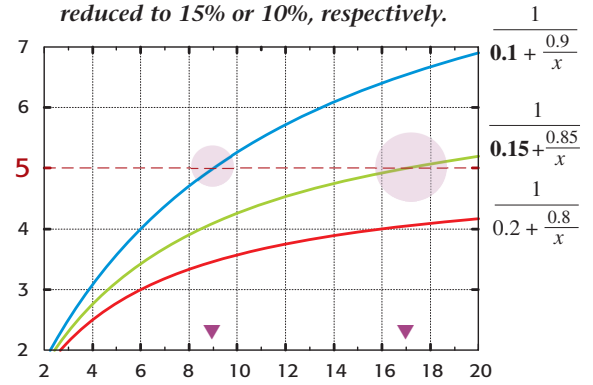
2nd scenario for twofold overall speedup.



[Example 3] Enhanced part: 2 instrs out of 5, part speedup: 2 cycles down to 1, i.e., doubled speed for 2/5 instrs. Therefore,  $\text{Frac}_{enh}=0.4$ ,  $\text{Speedup}_{enh}=2 \Rightarrow \text{Speedup}_{overall} = (0.6 + 0.4/2)^{-1} = 1.25$ .

[Example 4] No amount, technically. Check formula for  $\text{Speedup}_{enh} = \infty$ . No partial speedup can result in 5x overall when 1/5 of the time is not enhanced (since  $\lim_{\text{Speedup}_{enh} \rightarrow \infty} 5$ ). Even if improved a million times?  $1/(0.2+0.8/10^6) = 4.999980\dots$

A 5x overall may be attainable if the unaffected time were decreased (enhancement made more usable), e.g., with 17x or 9x part improvement if reduced to 15% or 10%, respectively.



Whenever there are fractional terms, a smart reader inspects them for division by 0 or  $\infty$ , particularly if a physical meaning is involved. Here,  $\infty$  means what if we apply a crazy big enhancement to the part. In general, limit on  $\text{Speedup}_{overall}$  is  $(1/\text{Frac}_{not\_enh})$  as  $\text{Speedup}_{enh} \rightarrow \infty$ .