Australian National University

COMP2100/6442
Software Design Methodologies / Software Construction

Performance Analysis

Amdahl's Law

Bernardo Pereira Nunes and
Sergio J. Rodríguez Méndez

A Bit of History | Gene Amdahl

Amdahl's Law named after Gene Amdahl

Published the paper "**Validity of the single processor approach to achieving large scale computing capabilities**"[1] in 1967.

Amdahl's Law says that for fixed workloads, the upper bound of **the speedup** expected from moving to a more parallelized environment **gets dominated by the sequential portion of the code as the parallel portion gets further divided into smaller and smaller chunks.**[2]

[1] https://inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf
[2] https://en.wikichip.org/wiki/amdahl%27s_law

Motivating Question

You have a program X with two component parts A and B, each of which takes 10 minutes. What is the *latency* of X?

Latency is the time from the beginning to the end to get a job done!

Suppose that you can speedup part B by a factor of 5. What is the latency now? What is the overall speedup?

If A and B are sequential, then Amdahl's Law provides an answer.

Amdahl's Law

How much extra performance can you get if you speed up part of your program?

Notations

S is the overall performance gain

k is the speed-up factor

α is the portion of speed-up

unimproved part

improved part

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

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

Amdahl's Law

Your program has one very slow procedure that **consumes 70% of the total time**. Next, you **improve it by a factor of 2**. What is the performance gain in the overall latency?

α = 0.7 (70%)
k = 2

$$S = \frac{T_{old}}{T_{new}} = \frac{1}{(1-\alpha)+\frac{\alpha}{k}} = \frac{1}{(1-0.7)+\frac{0.7}{2}} = 1.538$$

Amdahl's Law

Floating point instructions could be improved by 2x. Only 15% of instructions are floating point:

α = 0.15
k = 2

$$S = \frac{T_{old}}{T_{new}} = \frac{1}{(1-\alpha)+\frac{\alpha}{k}} = \frac{1}{(1-0.15)+\frac{0.15}{2}} = 1.081$$

Amdahl's law says to improve the most common task

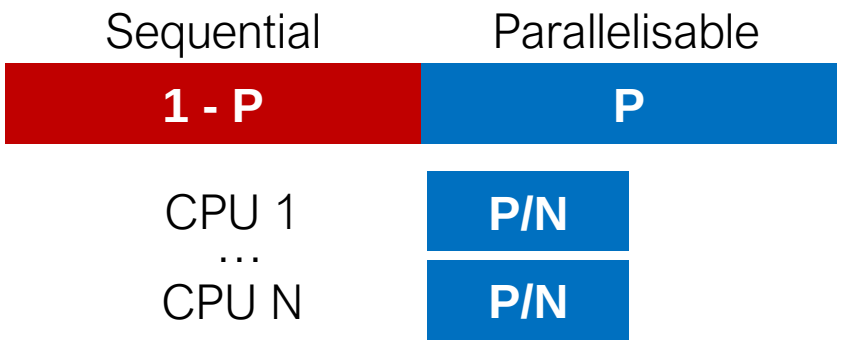
Application to Parallel Processing

Now, let's divide the program into a sequential part, **1-P**, and a parallel part, **P**.

Assume there are **N processors** then the improvement of the parallelisable part is **N**.

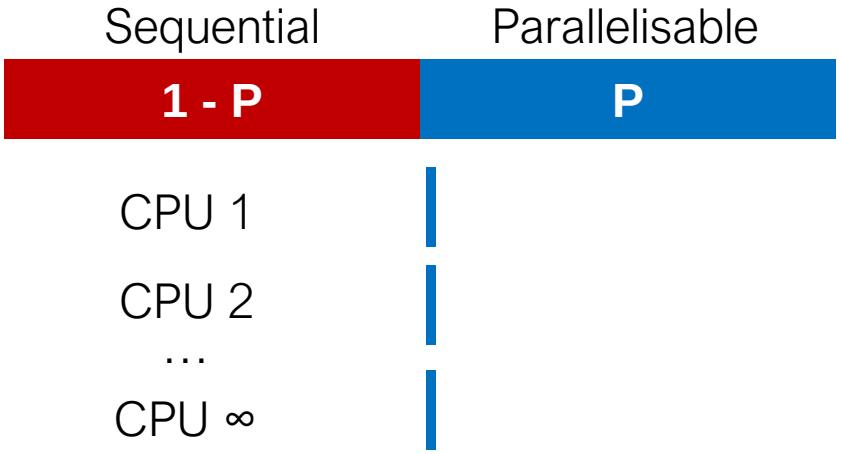
Based on Amdahl's law, the performance gain from N processors is:

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



Limit as N ⟶ ∞

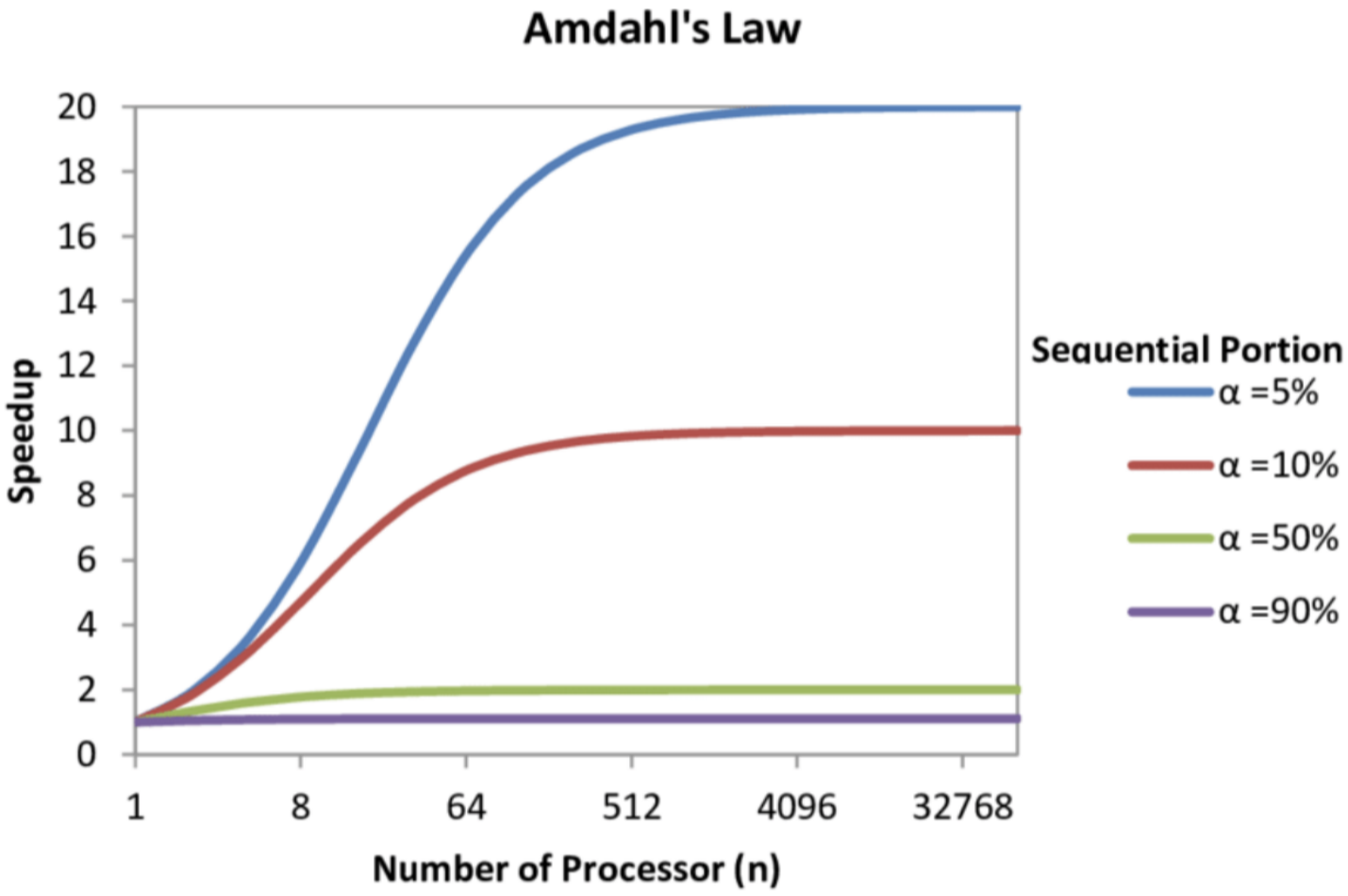
$$\lim_{N\to\infty} \frac{1}{(1-P)+\frac{P}{N}} = \frac{1}{1-P}$$



Fundamental limitation of performance gain from parallelisation (*diminishing returns*); adding more CPUs may not improve performance!

Neglects other potential bottlenecks such as memory bandwidth and I/O bandwidth.

Diminishing Returns

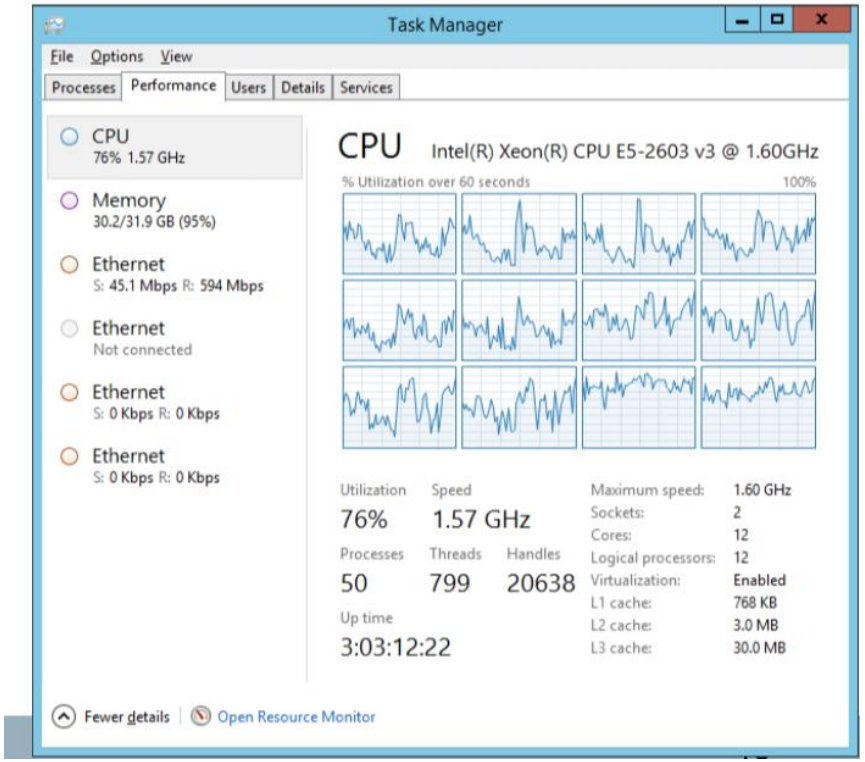


Various aspects of performance analysis

Does your software perform as what you expect?
- Does it complete fast?
- Does it work well with more inputs?
- Does it break?

Metrics of performance:
- Latency
- Throughput
- Memory size
- Network bandwidth
- Concurrency

Performance analysis
- Best case
- Average case
- Worst case



Performance Profiling

Tools that provide a visual interface for detailed information about the runtime operations of a program

Understand how your program utilises different computing resources (e.g. memory, CPU, GPU, hard disk, network)

Identify the bottleneck of your program

Optimise program implementation

Locate potential bugs in your program

Examples:
- Jconsole.
- VisualVM.
- Eclipse Trace Compass.

Trace Compass

Meme for today's lecture! Keep practicing!

Sequential vs Parallel

This course is about…

Design

Algorithms and Complexity

Data Structures

OOP

Design Patterns and UML

Data

JSON, XML, …

Testing

Unit Testing

JML

Performance Amdah's Law

Internals

Tokeniser / Parser

Tools

Git, IDEs

Development (Android/Java)

Intellectual Property/Copyright

COMP3600/6446 - Algorithms

COMP2120 - Software Engineering

COMP2400/6240 – Relational Databases

COMP2120 - Software Engineering

COMP2300 & COMP3320

COMP3610/6361 - Principles of Programming Languages

Recommended Reading

Amdahl's Law paper: Validity of the single processor approach to achieving large scale computing capabilities
https://inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf

Java and Parallel Programming
https://www.oracle.com/technical-resources/articles/java/fork-join.html