Amdahl's Law

The necessity to parallelize

Balázs Pál

Eötvös Loránd University

Data Models and Databases in Science,
December 3, 2020

The necessity to parallelize

- Algorithms can easily exceed any reasonable amount of runtime.
  - Example from the slides: *"Algorithms more complex than $\mathcal{O}(n \log(n))$ are hopeless to trace."*

The necessity to parallelize

- Algorithms can easily exceed any reasonable amount of runtime.
  - Example from the slides: *"Algorithms more complex than $\mathcal{O}(n \log(n))$ are hopeless to trace."*
  - Solution: parallelization of subtasks!

The necessity to parallelize

- Algorithms can easily exceed any reasonable amount of runtime.
  - Example from the slides: *"Algorithms more complex than $\mathcal{O}(n \log(n))$ are hopeless to trace."*
  - Solution: parallelization of subtasks!
- Problems
  - There are task that can be solved only sequentially (eg. reading from disk).
  - There are task which requires a lot of effort and work to parallelize.

The necessity to parallelize

- Algorithms can easily exceed any reasonable amount of runtime.
    - Example from the slides: *"Algorithms more complex than $\mathcal{O}\left(n\log\left(n\right)\right)$ are hopeless to trace."*
    - Solution: parallelization of subtasks!
- Problems
    - There are task that can be solved only sequentially (eg. reading from disk).
    - There are task which requires a lot of effort and work to parallelize.
- We expect - or at least hope - that parallelization shortens runtime significantly.

Amdahl's law

- Quantifies the theoretical magnitude of speedup due to parallelization.

Amdahl's law

- Quantifies the theoretical magnitude of speedup due to parallelization.
- We can split a whole arbitrary algorithm (denoting with 1) into to parts:

$$T = S \cdot T + P \cdot T \quad \rightarrow \quad 1 = S + P, \tag{1}$$

where T is the total execution time of the algorithm, $S <= 1$ denotes the fraction of runtime of the sequentially and $P <= 1$ the parallel solved part.

Amdahl's law

- Quantifies the theoretical magnitude of speedup due to parallelization.
- We can split a whole arbitrary algorithm (denoting with 1) into to parts:

$$T = S \cdot T + P \cdot T \quad \rightarrow \quad 1 = S + P, \tag{1}$$

  where T is the total execution time of the algorithm, $S <= 1$ denotes the fraction of runtime of the sequentially and $P <= 1$ the parallel solved part.
- Sequential part (S)
  - Takes a lot of time in every case, because this part is not parallelizable.
- Parallel part (P)
  - Doing the same operations in parallel on a lot of batch of data.
  - Should be always a lot faster than S.

Amdahl's law

- Amdahl's law now can be formulated as the following for tasks with fixed workload:

$$Q_{\text{speedup}}(N) = \frac{1}{S + \frac{P}{N}} = \frac{1}{S + \frac{1-S}{N}}, \tag{2}$$

Where N is the number of parallel threads, and the runtime P was expressed in the denominator using equation (1).

Amdahl's law

- Amdahl's law now can be formulated as the following for tasks with fixed workload:

$$Q_{\text{speedup}}(N) = \frac{1}{S + \frac{P}{N}} = \frac{1}{S + \frac{1-S}{N}}, \tag{2}$$

Where N is the number of parallel threads, and the runtime P was expressed in the denominator using equation (1).

- The behaviour of the Q speedup's value can be expressed as

$$\begin{cases} Q_{\text{speedup}} = \frac{1}{S} & \text{if } N \to \infty, \\ Q_{\text{speedup}} < \frac{1}{S} & \text{else.} \end{cases} \tag{3}$$

Amdahl's law

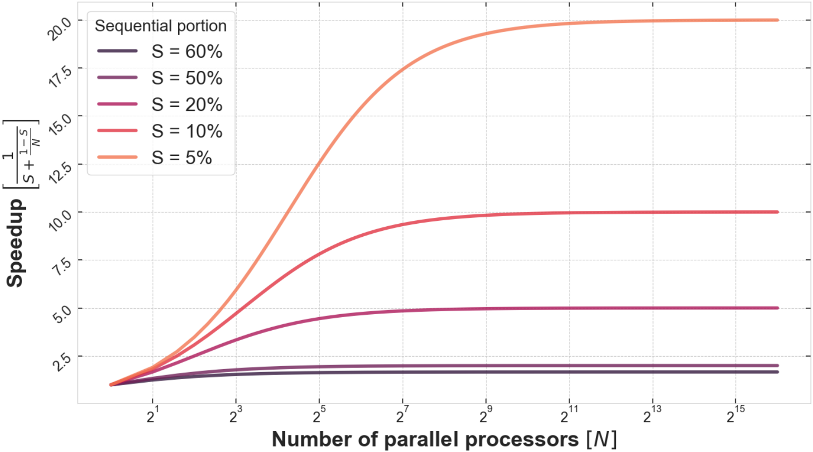

Figure 1: Visualization of the speedup with increasing number of parallel threads for different values of S. The figure shows that when the sequential part of the task is just the small portion of the whole algorithm, then the speedup is the highest. It can also seen, that the limit for the speedup is $\lim_{N\to\infty} = \frac{1}{S}$

Other forms of Amdahl's laws

(Embarassingly just from Wikipedia)

- Optimizing the sequential part of parallel programs

$$Q_{\text{speedup}}(O, N) = \frac{\frac{S}{O} + (1 - S)}{\frac{S}{O} + \frac{1-S}{N}}, \tag{4}$$

where O is the speedup of the sequential runtime, so $T_{S_{\text{new}}} = T_{S_{\text{old}}}/O$.

- Transforming sequential parts of parallel programs into parallelizable

$$Q_{\text{speedup}}(O', N) = \frac{1}{\frac{S}{O'} + \left(1 - \frac{S}{O'}\right)\frac{1}{N}}, \tag{5}$$

where S is reduced by a factor of O', so $S_{\text{new}} = S_{\text{old}}/O'$.

Amdahl's number

- Objectively quantifies how fast the procession of a task is on an arbitrary machine:

$$A = \frac{1\,\text{bit}\,\frac{\text{IO}}{\text{sec}}}{1\,\frac{\text{instruction}}{\text{sec}}} \tag{6}$$

- *"How many CPU instructions needed to process* 1 *bit of data?"*

Amdahl's number

- Objectively quantifies how fast the procession of a task is on an arbitrary machine:

$$A = \frac{1\,\mathrm{bit}\,\frac{\mathrm{IO}}{\mathrm{sec}}}{1\,\frac{\mathrm{instruction}}{\mathrm{sec}}} \qquad (6)$$

- *"How many CPU instructions needed to process* 1 *bit of data?"*
- Of course the bigger the value of A is the better.

Typical values for A

- Example from the lecture slides: Server with four SSDs
  - 4 SSDs with 150 MB/s read/write speed: $4 \cdot 150$ MB/s $= 4.8$ Gb/s (Megabyte to Gigabit!)
  - CPU with 8 cores with a clock speed of 2.5 GHz: $8 \cdot 2.5$ GHz $= 20$ G.inst/s
  - Value of A is $4.8/20 = 0.24$.

Typical values for A

- Example from the lecture slides: Server with four SSDs
  - 4 SSDs with 150 MB/s read/write speed: $4 \cdot 150$ MB/s $= 4.8$ Gb/s (Megabyte to Gigabit!)
  - CPU with 8 cores with a clock speed of 2.5 GHz: $8 \cdot 2.5$ GHz $= 20$ G.inst/s
  - Value of A is $4.8/20 = 0.24$.
- Some other values
  - Compute cluster (CC.) IBM Blue Gene : $A = 0.013$
  - CC. Beowulf : $A = 0.08$
  - CC. Cloud VM : $A = 0.08$

    $\vdots$
  - Average desktop PC : $A = 0.2$
  - Atom+Ion+SSD : $A = 1.25$