Hasso Plattner Institut
Digital Engineering • Universität Potsdam

Parallel Programming and Heterogeneous Computing

A3 - Performance Metrics

Max Plauth, Sven Köhler, Felix Eberhardt, Lukas Wenzel and Andreas Polze
Operating Systems and Middleware Group

Performance

- Which car is faster?
  - … for transporting several large boxes
  - … for winning a race

Performance depends not only on an execution environment but also on the workload it executes!

Recap
Optimization Goals

- Decrease **Latency** – process a single workload faster (= **speedup**)
- Increase **Throughput** – process more workloads in the same time
- ➢ Both are **Performance** metrics

- **Scalability**: make best use of additional resources
  - **Scale Up**: Utilize additional resources on a machine
  - **Scale Out**: Utilize resources on additional machines
- **Cost/Energy Efficiency**:
  - minimize cost/energy requirements for given performance objectives
  - *alternatively: maximize performance for given cost/energy budget*
- **Utilization**: minimize idle time (=waste) of available resources
- **Precision-Tradeoffs**: trade performance for precision of results

Scaling Behavior

Different responses of performance metrics to scaling (additional resources):

- **Speedup**:
  More resources ~ less time executing the **same workload**

  › strong scaling

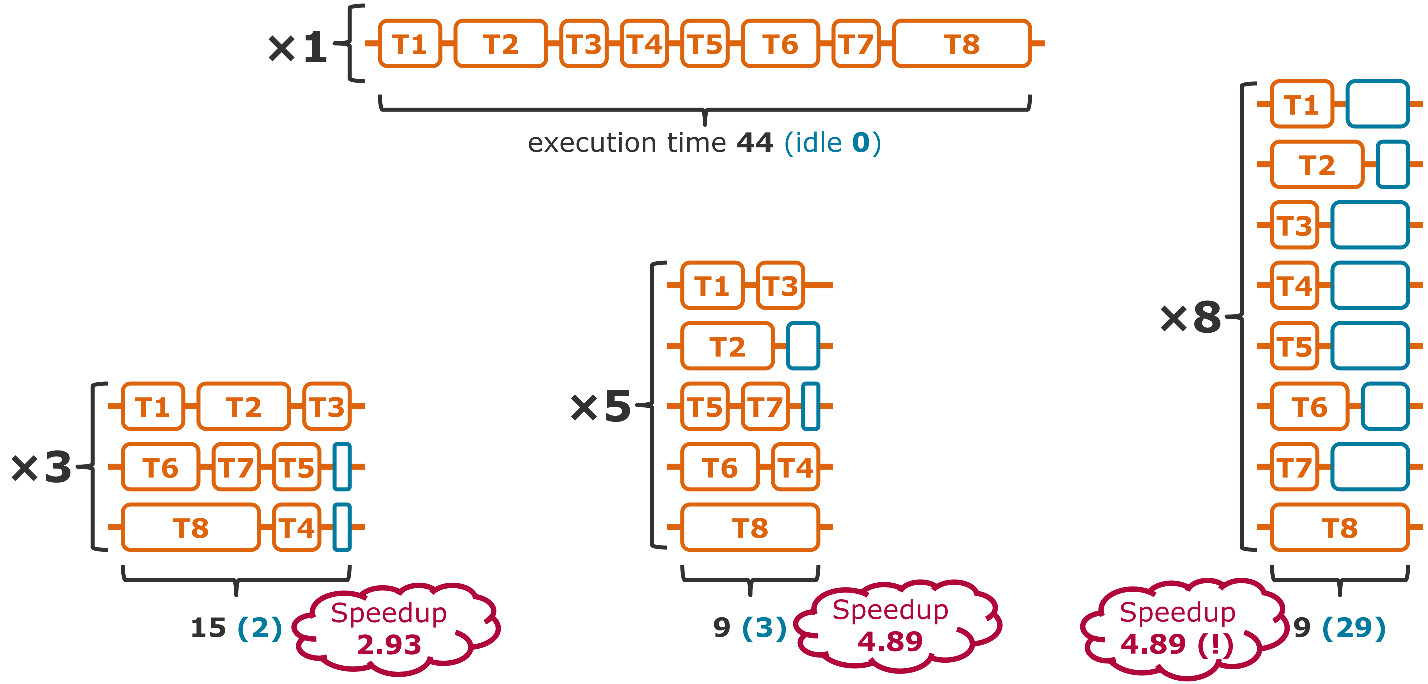
- **Scaled Speedup**:
  More resources ~ same time executing a **larger workload**

  › weak scaling

- Linear speedup = resources and workload execution scale by same factor

Anatomy of a Workload

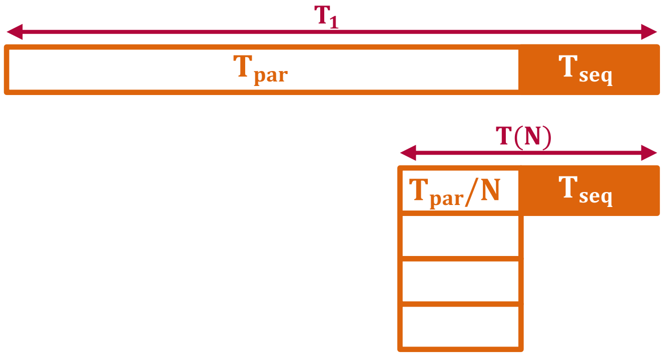
A workload consists of multiple tasks, containing different amounts of operations each.



Anatomy of a Workload

The **longest task** puts a **lower bound on** the shortest **execution time.**

Modeling discrete tasks is impractical → simplified **continuous model.**



$$T(N) = \frac{T_{par}}{N} + T_{seq}$$

Replace absolute times by **parallelizable fraction** P:

$$T_{par} = T_1 \cdot P$$
$$T_{seq} = T_1 \cdot (1 - P)$$

$$T(N) = T_1 \cdot \left(\frac{P}{N} + (1 - P)\right)$$

[Amdahl1967]
Amdahl's Law

Amdahl's Law derives the speedup $\mathbf{s_{Amdahl}(N)}$ for a parallelization degree $\mathbf{N}$

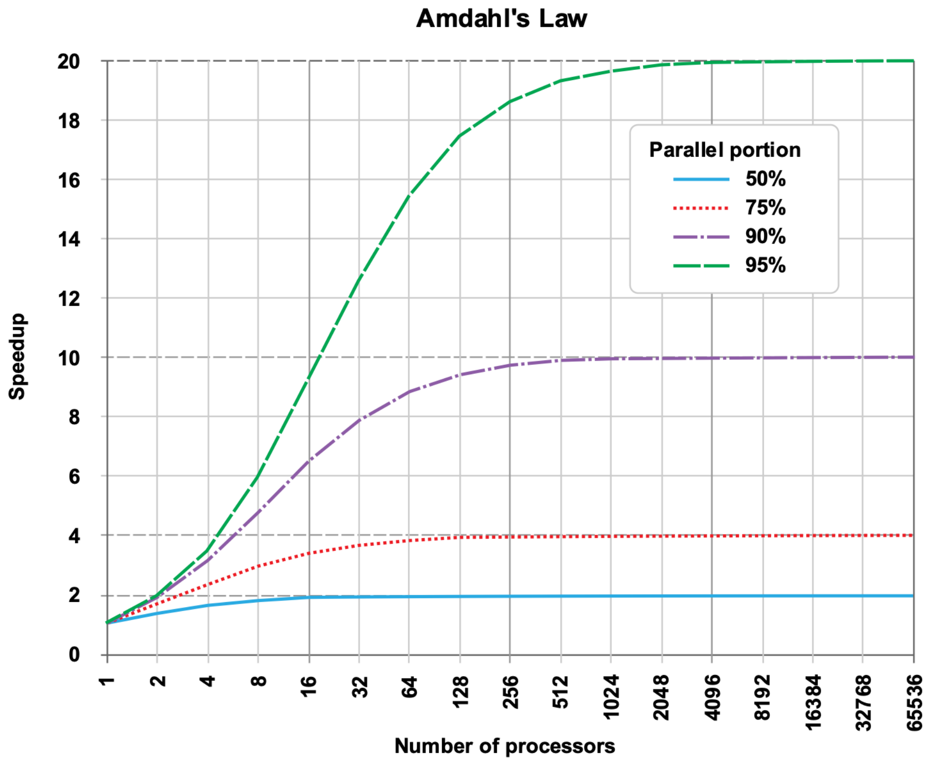
$$\mathbf{s_{Amdahl}(N)} = \frac{T_1}{T(N)} = \frac{T_1}{T_1 \cdot \left(\frac{P}{N} + (1-P)\right)} = \frac{\mathbf{1}}{\frac{\mathbf{P}}{\mathbf{N}} + (\mathbf{1} - \mathbf{P})}$$

Even for **arbitrarily large** $\mathbf{N}$, the speedup converges to a **fixed limit**

$$\lim_{N \to \infty} s_{Amdahl}(N) = \frac{\mathbf{1}}{\mathbf{1} - \mathbf{P}}$$

For getting reasonable speedup out of 1000 processors, the sequential part must be substantially below 0.1%

[Amdahl1967]
Amdahl`s Law



[Amdahl1967]
Amdahl`s Law

*Regardless of processor count, **90% parallelizable** code allows not more than a **speedup by factor 10**.*

- Parallelism requires highly parallelizable workloads to achieve a speedup
- What is the sense in large parallel machines?

Amdahl's law assumes a simple speedup scenario!

- isolated execution of a **single workload**
- **fixed workload size**

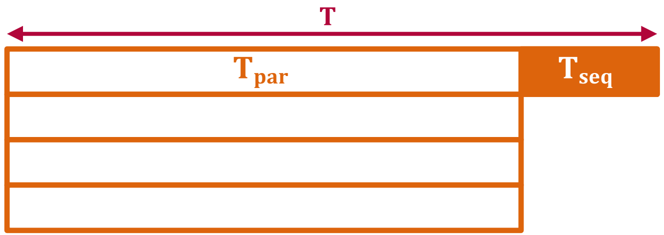
[Gustafson1988]
Gustafson-Barsis’ Law

Consider a **scaled speedup scenario**, allowing a variable workload size $\mathbf{w}$.

Amdahl ~ *What is the shortest execution time for a given workload?*

Gustafson-Barsis ~ *What is the largest workload for a given execution time?*

$$\mathbf{w}_1 \sim T_{par} + T_{seq}$$



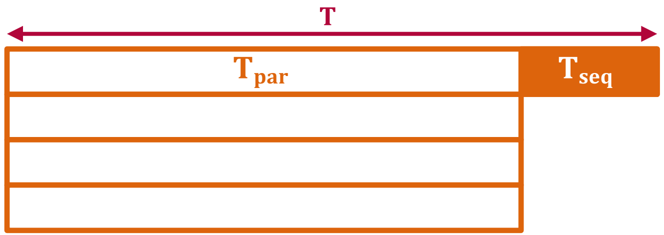
$$\mathbf{w}(N) \sim N \cdot T_{par} + T_{seq}$$

Assumption: The parallelizable part of a workload contributes useful work when replicated.

[Gustafson1988]
Gustafson-Barsis’ Law

Determine the scaled speedup $s_{Gustavson}(N)$ through
the increase in workload size $w(N)$ over the fixed execution time T

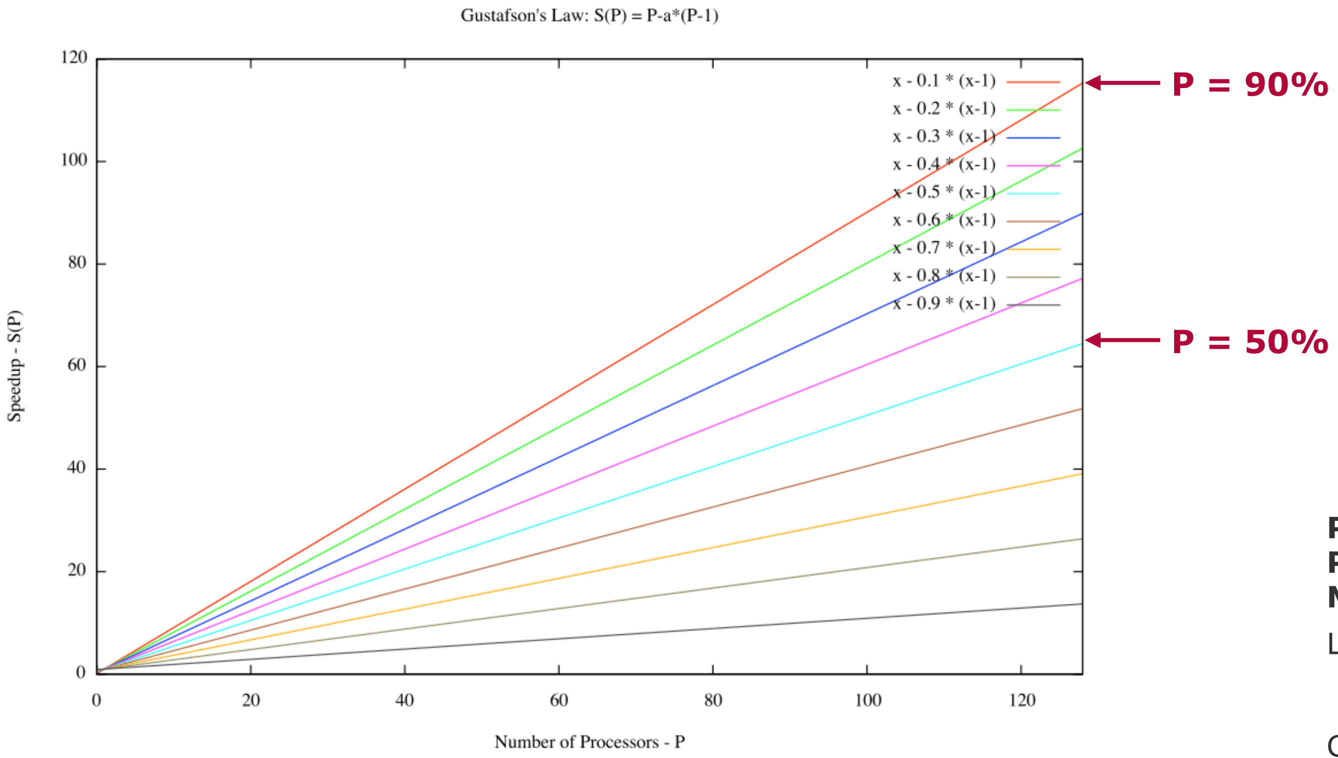
$$s_{Gustafson}(N) = \frac{w(N)}{w_1} = \frac{T \cdot (P \cdot N + (1 - P))}{T \cdot (P + (1 - P))} = \mathbf{P \cdot N + (1 - P)}$$



$$w_1 \sim T_{par} + T_{seq} \qquad w(N) \sim N \cdot T_{par} + T_{seq}$$

Assumption: The parallelizable part of a workload contributes useful work when replicated.

[Gustafson1988] Gustafson-Barsis' Law



[Karp1990]
Karp-Flatt-Metric

Parallel fraction $\mathbf{P}$ is a hypothetical parameter and not easily deduced from a given workload.

- Karp-Flatt-Metric determines sequential fraction $\mathbf{Q} = \mathbf{1} - \mathbf{P}$ empirically
  1. Measure baseline execution time $\mathbf{T_1}$
     by executing workload on a single execution unit
  2. Measure parallelized execution time $\mathbf{T}(\mathbf{N})$
     by executing workload on $\mathbf{N}$ execution units
  3. Determine speedup $\mathbf{s}(\mathbf{N}) = {}^{\mathbf{T_1}}/_{\mathbf{T}(\mathbf{N})}$
  4. Calculate Karp-Flatt-Metric

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

[Karp1990]
Karp-Flatt-Metric

The Karp-Flatt-Metric is derived by rearranging Amdahl's Law.

$$\frac{1}{s(N)} = \frac{T(N)}{T_1} \quad ; \quad T(N) = \left(\frac{1-Q}{N} + Q\right) \cdot T_1$$

$$\frac{1}{s(N)} = \frac{\left(\frac{1-Q}{N} + Q\right) \cdot T_1}{T_1}$$

$$\frac{1}{s(N)} = \frac{1-Q}{N} + Q = \frac{1}{N} + \left(1 - \frac{1}{N}\right) \cdot Q$$

$$\frac{1}{s(N)} - \frac{1}{N} = \left(1 - \frac{1}{N}\right) \cdot Q$$

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

[Karp1990] Karp-Flatt-Metric

Observing $\mathbf{Q}(\mathbf{N})$ for different $\mathbf{N}$ gives an indication,
how the workload reacts to different degrees of parallelism:

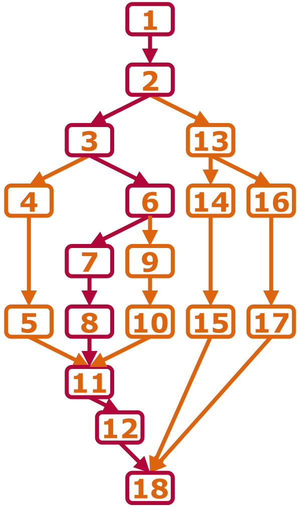
- $\mathbf{Q}(\mathbf{N})$ close to $\mathbf{0}$ ~ *high parallel fraction, workload benefits from parallelization*
- $\mathbf{Q}(\mathbf{N})$ close to $\mathbf{1}$ ~ *low parallel fraction, workload can not use parallel resources*
- $\mathbf{Q}(\mathbf{N})$ increases with $\mathbf{N}$ ~ *workload suffers from parallelization overhead*
- $\mathbf{Q}(\mathbf{N})$ decreases with $\mathbf{N}$ ~ *workload scales well*

Observing $\mathbf{Q}(\mathbf{N})$ for different implementation variants of the workload can reveal bottlenecks.

[Leiserson2008]
A More Detailed View

Directed Acyclic Graph to model a workload:

- Nodes represent operations
- Edges express dependencies between operations



$\mathbf{T_1} = \mathbf{18}$

$\mathbf{T_\infty} = \mathbf{9}$

Work $\mathbf{T}$ - Total workload execution time

$\mathbf{T_1}$ - Execution time with a single processor

~ number of nodes

$\mathbf{T_P}$ - Execution time with P processors

$\mathbf{T_\infty}$ - Execution time with arbitrary number of processors

~ graph diameter

Work Law $T_P \geq {}^{T_1}/_{P}$

(processors can not process multiple operations at once)

Span Law $T_P \geq T_\infty$

(execution order can not break dependencies)

Literature

[Amdahl1967]

Amdahl, Gene M. "Validity of the single processor approach to achieving large scale computing capabilities." *Proceedings of the AFIPS Spring Joint Computer Conference*. 483-485. 1967.

[Gustafson1988]

Gustafson, John L. "Reevaluating Amdahl's law." *Communications of the ACM* 31.5 (1988): 532-533.

[Karp1990]

Karp, Alan H. and Flatt, Horace P. "Measuring parallel processor performance." *Communications of the ACM* 33.5 (1990): 539-543.

[Leiserson2008]

Leiserson, Charles E. and Mirman, Ilya B. "How to survive the multicore software revolution (or at least survive the hype)." *Cilk Arts* 1 (2008): 11.

Hasso Plattner Institut
Digital Engineering • Universität Potsdam

And now for a break and a cup of Oolong.

*or beverage of your choice