

# Übung 10: Parallelisierung

## Einführung in die Rechnerarchitektur

**Niklas Ladurner**

School of Computation, Information and Technology  
Technische Universität München

3. Januar 2025



*TUM Uhrenturm*

Keine Garantie für die Richtigkeit der Tutorfolien.  
Bei Unklarheiten/Unstimmigkeiten haben VL/ZÜ-Folien recht!

- Single-Threaded Rechenleistung immer weiter durch physikalische Limits eingeschränkt
- Optimierungen: Pipelining, Out-of-Order-Processing, Ausnutzen von Parallelität
- SIMD: Eine Instruktion, die gleichzeitig auf mehrere Daten ausgeführt wird (mehr dazu in GRA)

		Instruction stream	
		Single	Multiple
Data stream	Single	SISD	MISD
	Multiple	SIMD	MIMD

Quelle: A Taxonomy of Reconfigurable Single-/Multiprocessor Systems-on-Chip

- Mehrkernsysteme: Was wenn CPU1 und CPU2 beide ein Datum gecached haben und es modifizieren?  
→ Cache-Inkonsistenzen
- Einführung von Zuständen für Cachezeilen
- **M**odified, (**E**xclusive), **S**hared, **I**nvalid



lokal



entfernt

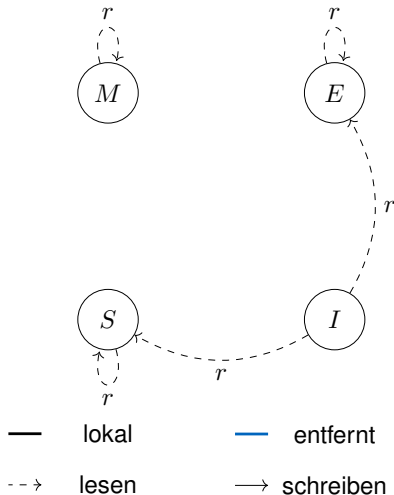


lesen

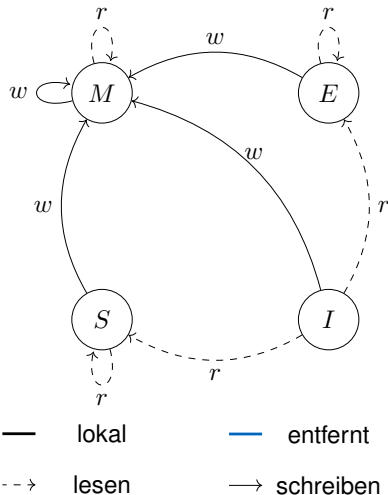


schreiben

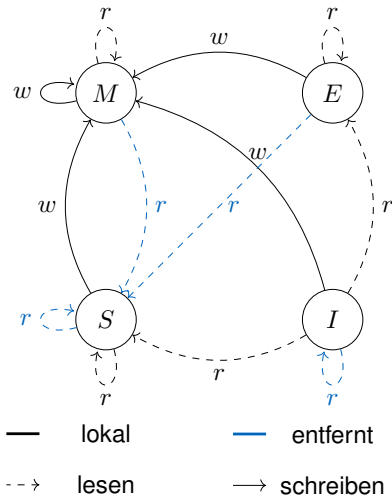
- Mehrkernsysteme: Was wenn CPU1 und CPU2 beide ein Datum gecached haben und es modifizieren?  
→ Cache-Inkonsistenzen
- Einführung von Zuständen für Cachezeilen
- **M**odified, (**E**xclusive), **S**hared, **I**nvalid



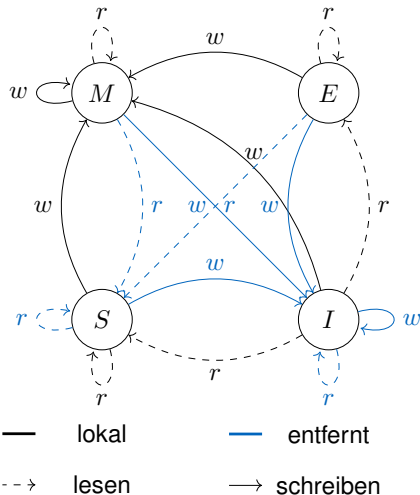
- Mehrkernsysteme: Was wenn CPU1 und CPU2 beide ein Datum gecached haben und es modifizieren?  
→ Cache-Inkonsistenzen
- Einführung von Zuständen für Cachezeilen
- **M**odified, (**E**xclusive), **S**hared, **I**nvalid



- Mehrkernsysteme: Was wenn CPU1 und CPU2 beide ein Datum gecached haben und es modifizieren?  
→ Cache-Inkonsistenzen
- Einführung von Zuständen für Cachezeilen
- **M**odified, (**E**xclusive), **S**hared, **I**nvalid



- Mehrkernsysteme: Was wenn CPU1 und CPU2 beide ein Datum gecached haben und es modifizieren?  
→ Cache-Inkonsistenzen
- Einführung von Zuständen für Cachezeilen
- **M**odified, (**E**xclusive), **S**hared, **I**nvalid





Mit  $t_s$  sequentieller Programmteil,  $t_p$  paralleler Programmteil,  $n$  Anzahl CPU-Kerne,  $T$  Ausführungszeit mit  $n = 1$ :

- Amdahl's Law: Gleiche Problemgröße, aufgeteilt auf mehrere Kerne  $\rightarrow$  begrenzt durch sequentiellen Anteil

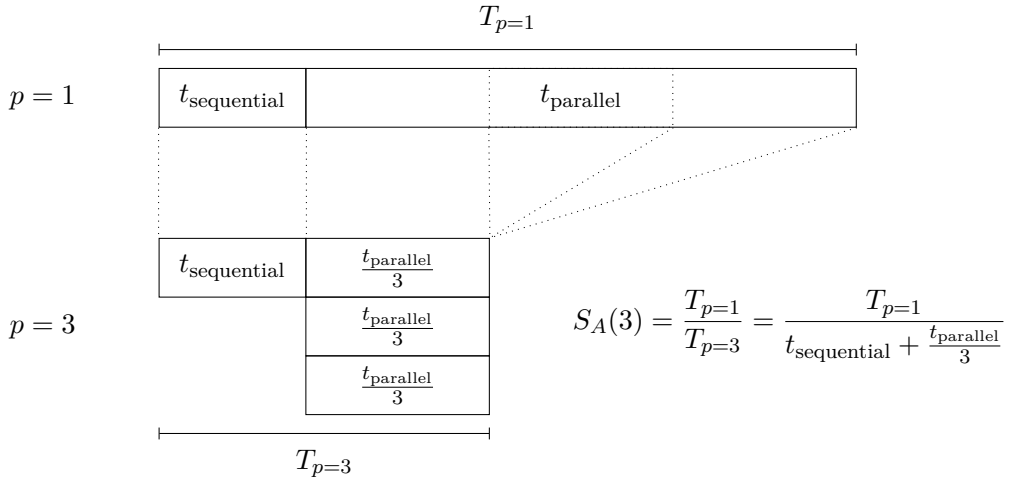
$$S_{\text{Amdahl}}(n) = \frac{T}{t_s + \frac{t_p}{n}}$$

- Gustafson's Law: Mehr Kerne können mehr berechnen: Größeres Problem  $\rightarrow$  paralleler Anteil wächst mit Problemgröße,  $t_s$  proportional kleiner

$$S_{\text{Gustafson}}(n) = \frac{t_s + n \cdot t_p}{T}$$

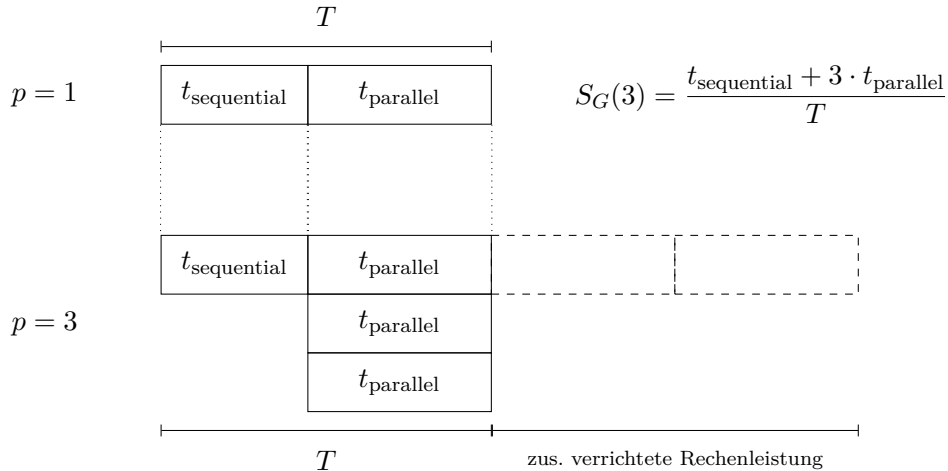
- Zwei verschiedene Perspektiven, abhängig von Problemszenario verschieden geeignet

# Amdahlsches Gesetz



Quelle: Vorlesungsmaterial ERA

# Gustafsons Gesetz



- „H10 — MESI“ bis 12.01.2025 23:59 Uhr
- Durchlaufen der MESI-Zustände für 4 parallel arbeitende CPUs
  
- „B01 — RFC 9402“ bis 26.01.2025 23:59 Uhr
- Wiederholungsaufgabe RISC-V Assembly: Strings zusammenkopieren
- Erste (und vsl. letzte) Bonusaufgabe: 10 Punkte Bonuspunkte!

- Zulip: „ERA Tutorium - Do-1600-1“ bzw. „ERA Tutorium - Fr-1500-2“
- ERA-Moodle-Kurs
- ERA-Artemis-Kurs
- Wikipedia zu MESI
- Amdahlsches und Gustafsons Gesetz

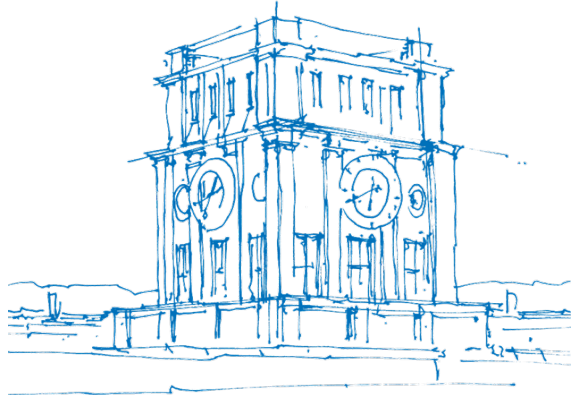
# Übung 10: Parallelisierung

## Einführung in die Rechnerarchitektur

**Niklas Ladurner**

School of Computation, Information and Technology  
Technische Universität München

3. Januar 2025



*TUM Uhrenturm*