

# Untersuchung der Skalierbarkeit von parallelem Sortieren auf einem Multicore-Prozessor

## Verteidigung der Bachelorarbeit

Leon Zoerner

Informatik

12. Februar 2026

# Agenda

- 1 Formeln
- 2 Einleitung
- 3 Theoretische Grundlagen
- 4 Methodik und Versuchsaufbau
- 5 Ergebnisse und Analyse
- 6 Diskussion und Fazit

$$T(n) = m_1 + m_2 + n$$

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n$$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n,$$

$$T(n) = n \cdot \log_2(n) + n,$$

$$O(T(n)) = O(n \log n).$$

**Best-Case** von Quicksort:

$$T(n) = q1 + q2 + n$$

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n$$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n,$$

$$T(n) = n \cdot \log_2(n) + n,$$

$$O(T(n)) = O(n \log n).$$

# Formeln: Quicksort

Der **Worst-Case** von Quicksort ist:

$$T(n) = q_1 + q_2 + n$$

$$T(n) = T(n-1) + 1 + n,$$

$$T(n) = \frac{1}{2} \cdot (n^2 + n) + n,$$

$$O(T(n)) = O(n^2).$$

Der **heuristisch betrachtete Average-Case** von Quicksort ist:

$$T(n) = q_1 + q_2 + n$$

$$q_1 = T\left(\frac{1 + \dots + (n-1)}{n-1}\right) = T\left(n \cdot \frac{n-1}{2} \cdot \frac{1}{n-1}\right) = T\left(\frac{n}{2}\right) = q_2$$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$$

$$T(n) = n \cdot \log_2(n) + n$$

$$O(T(n)) = O(n \log n).$$

# Formeln: parallel Mergesort

$$p = \text{Thread-Anzahl}$$

$$e = \log_2(p)$$

$$T(n, e) = \begin{cases} 2 \cdot T\left(\frac{n}{2}, 0\right) + n & , \text{ wenn } e = 0 \\ 1 \cdot T\left(\frac{n}{2}, e - 1\right) + n & , \text{ wenn } e > 0 \end{cases}$$

$$T(n, p) = 2n \left(1 - \frac{1}{p}\right) + \frac{n}{p} \cdot \log_2 \left(\frac{n}{p}\right) + \frac{n}{p}$$

$$O(T(n, p)) = O\left(\frac{n}{p} \cdot \log_2(n) + n\right)$$

$$e_{\max} = \log_2(n)$$

$$p_{\max} = n$$

Der **Best-Case** und **heuristisch betrachtete Average-Case** von Quicksort ist:

$$p = \text{Thread-Anzahl}$$

$$e = \log_2(p)$$

$$T(n, e) = \begin{cases} 2 \cdot T\left(\frac{n}{2}, 0\right) + n & , \text{ wenn } e = 0 \\ 1 \cdot T\left(\frac{n}{2}, e - 1\right) + n & , \text{ wenn } e > 0 \end{cases}$$

$$T(n, p) = 2n \left(1 - \frac{1}{p}\right) + \frac{n}{p} \cdot \log_2 \left(\frac{n}{p}\right) + \frac{n}{p}$$

$$O(T(n, p)) = O\left(\frac{n}{p} \cdot \log_2(n) + n\right)$$

$$e_{\max} = \log_2(n)$$

$$p_{\max} = n$$

Der **Worst-Case** von Quicksort (Worker-Thread-Variante) bei  $p > 1$  ist:

$$T(n) = q_2 + n,$$

$$T(n) = T(n-1) + n,$$

$$T(n) = \frac{1}{2} \cdot (n^2 + n),$$

$$O(T(n)) = O(n^2).$$



# Formeln: Einheiten und Skalierung

$$x \cdot T(n) = 2 \cdot T\left(\frac{n}{2}\right) + x \cdot n$$

$$x \cdot T(n) = x \cdot n \cdot \log_2(n) + x \cdot n$$

$$x \cdot T(n) = x \cdot (n \cdot \log_2(n) + n)$$

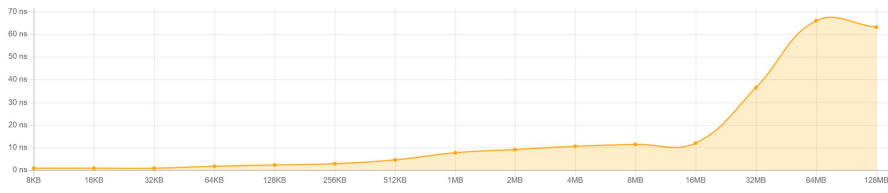
Aufgrund von Overheads wie z. B. erhöhte Speicherlatenzen durch Cache-Misses:

$$x = f(n, p)$$

$$f(n, p) \leq f(n+1, p)$$

$$f(n, p) \leq f(n, p+1)$$

L1/L2/L3 CPU cache and main memory (DIMM) access latencies in nano seconds



# Motivation & Zielsetzung





# Implementierungsvarianten Code

# Laufzeitmessungen (sequenziell)

# Analyse der Threading-Methoden

# Beantwortung der Forschungsfrage



# Zusammenfassung und Ausblick

# Vielen Dank für Ihre Aufmerksamkeit!

Fragen und Diskussion