

Untersuchung der Skalierbarkeit von parallelem Sortieren auf einem Multicore-Prozessor

Verteidigung der Bachelorarbeit

Leon Zoerner

Informatik

15. Februar 2026

Agenda

- 1 Einleitung
- 2 Theoretische Grundlagen
- 3 Methodik und Versuchsaufbau
- 4 Ergebnisse und Analyse
- 5 Diskussion und Fazit
- 6 Binärbaum
- 7 Formeln
- 8 Herleitung $T(n,p)$

Motivation & Zielsetzung

Implementierungsvarianten Code

Laufzeitmessungen (sequenziell)

Analyse der Threading-Methoden

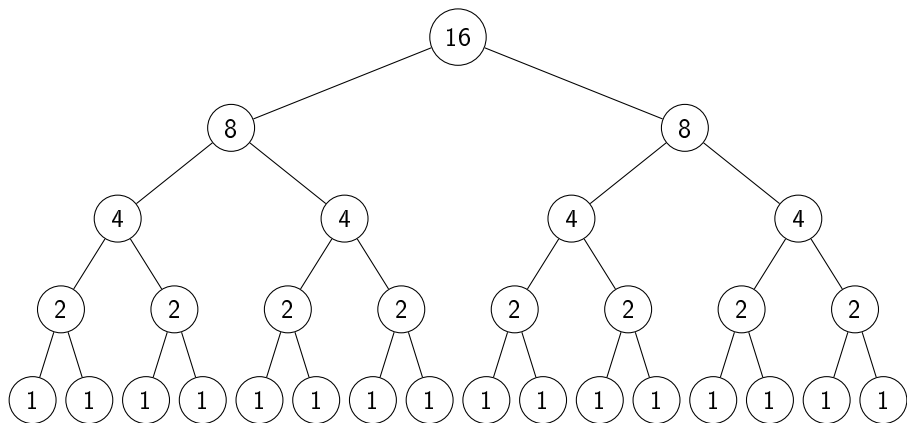
Beantwortung der Forschungsfrage

Zusammenfassung und Ausblick

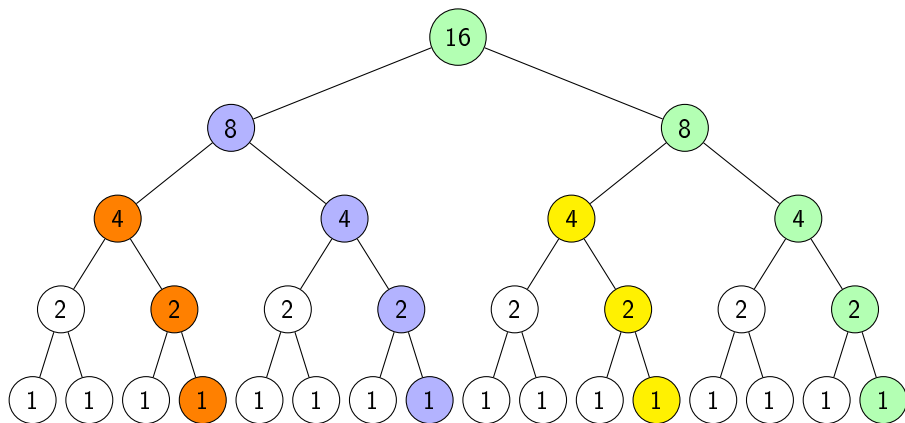
Vielen Dank für Ihre Aufmerksamkeit!

Fragen und Diskussion

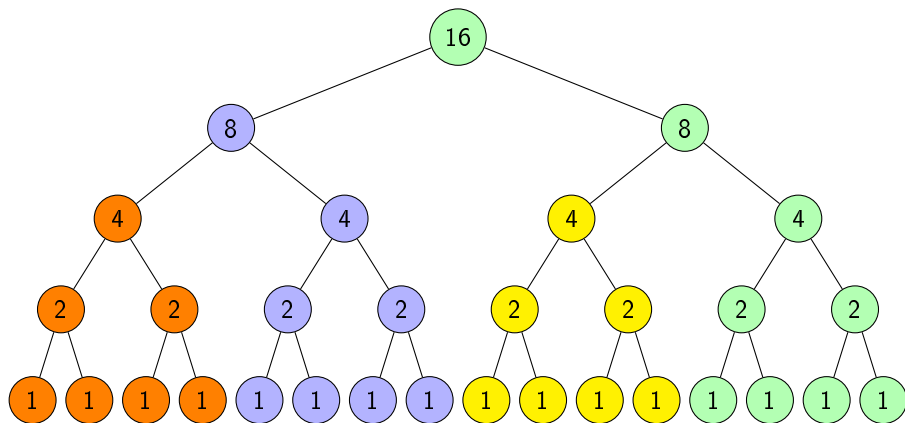
Balancierter Binärbaum für $n = 16$



Balancierter Binärbaum für $n = 16$, $p = 4$, $e = 2$

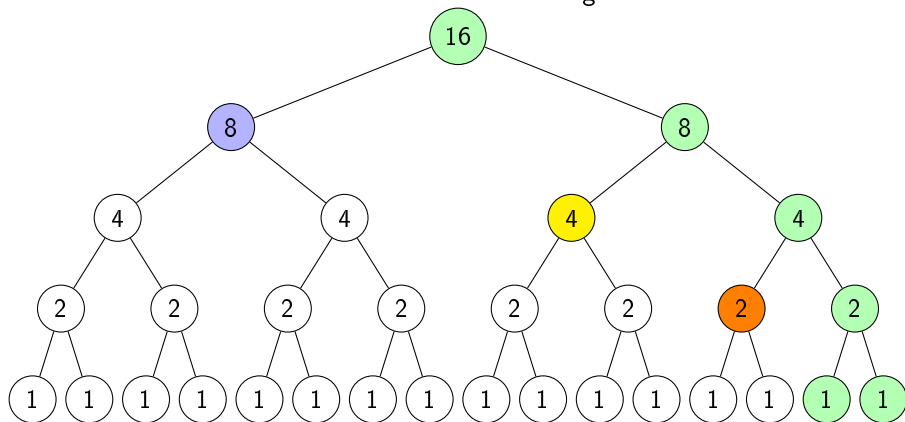


Balancierter Binärbaum für $n = 16$, $p = 4$, $e = 2$



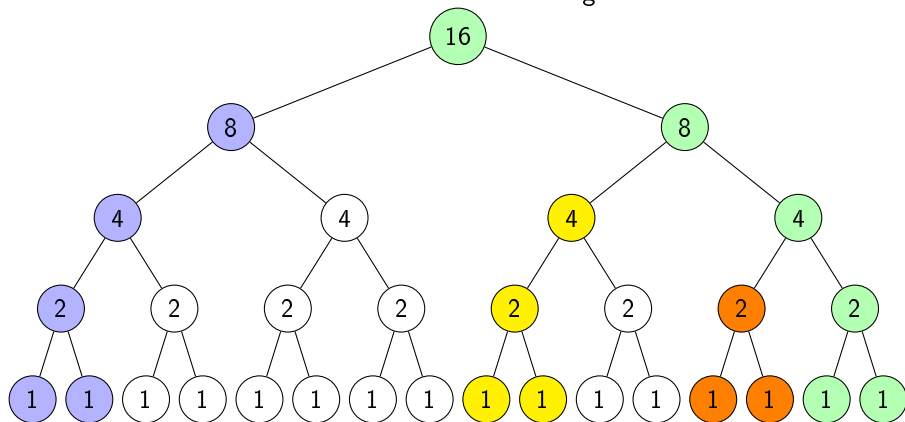
Balancierter Binärbaum für $n = 16$, $p = 4$

Worker-Thread-Variante Mergesort:



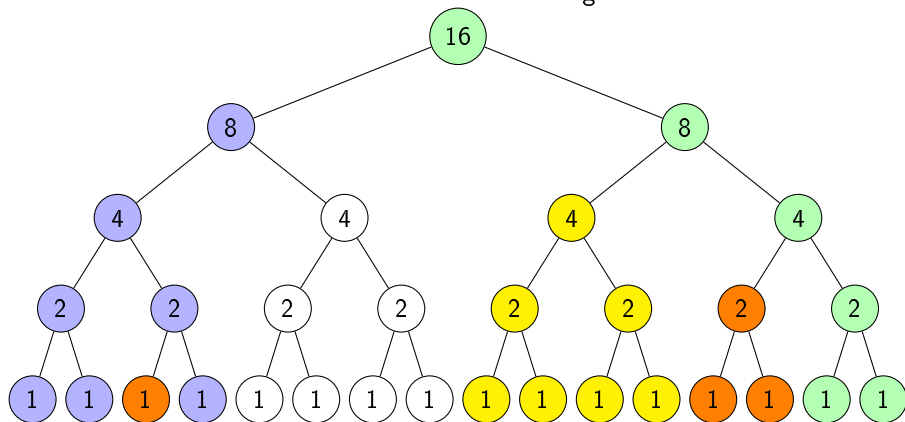
Balancierter Binärbaum für $n = 16$, $p = 4$

Worker-Thread-Variante Mergesort:



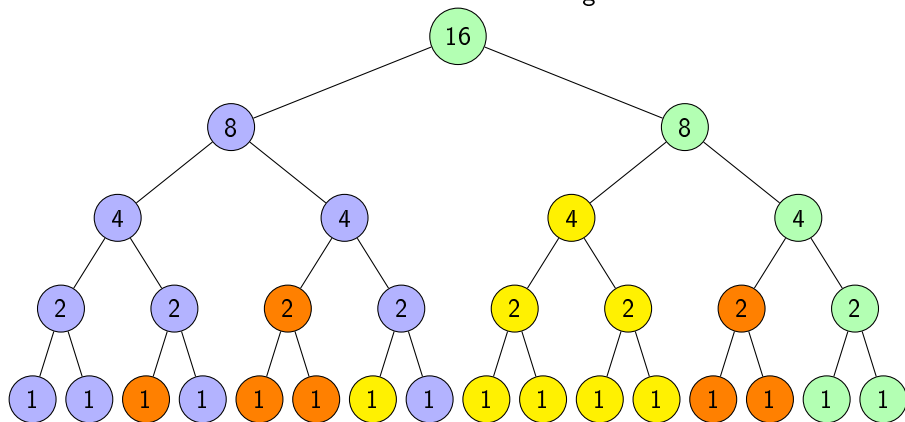
Balancierter Binärbaum für $n = 16$, $p = 4$

Worker-Thread-Variante Mergesort:



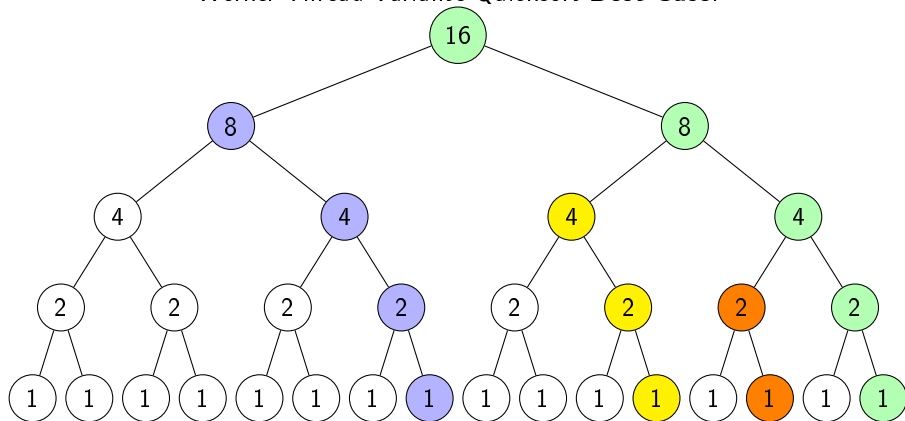
Balancierter Binärbaum für $n = 16$, $p = 4$

Worker-Thread-Variante Mergesort:



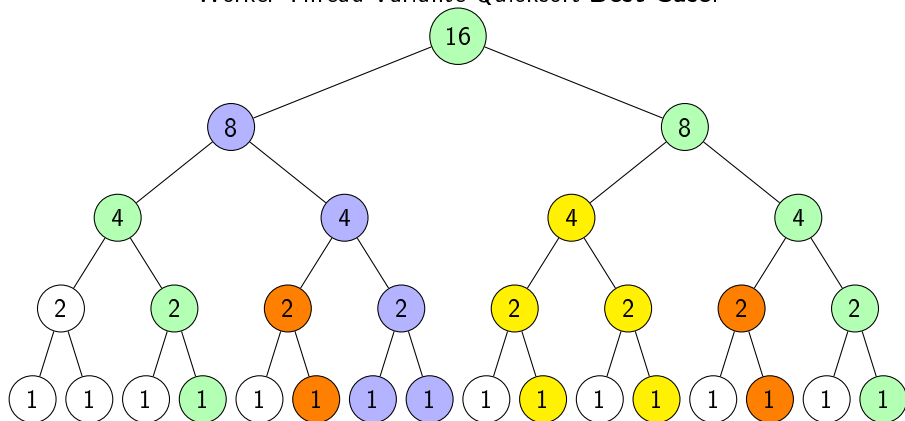
Balancierter Binärbaum für $n = 16$, $p = 4$

Worker-Thread-Variante Quicksort **Best-Case**:



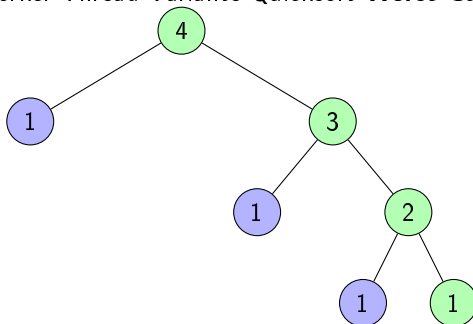
Balancierter Binärbaum für $n = 16$, $p = 4$

Worker-Thread-Variante Quicksort **Best-Case**:



Maximal unbalancierter Binärbaum für $n = 4$, $p = 2$

Worker-Thread-Variante Quicksort **Worst-Case**:



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

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

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

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

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

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:

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

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

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

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

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

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

$$e = \log_2(p)$$

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

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

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

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

$$p_{\max} = n$$

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

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

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

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

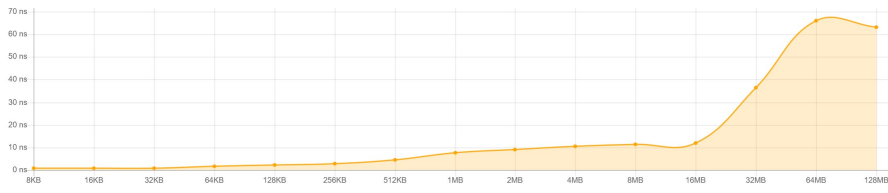
Aufgrund von Overheads wie z. B. erhöhte Speicherlatenzen durch Cache-Misses:

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

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

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

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



Nebeneffekt des Skalierens mit dem Faktor x

Sollte die Standard-Formel

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

falsch sein und stattdessen diese Formel gelten

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

kann man diese auch schreiben als

$$T(n) = 2T\left(\frac{n}{2}\right) + \underbrace{\left(1 - \frac{1}{n}\right)}_x \cdot n$$

Dabei ist $\left(1 - \frac{1}{n}\right)$ implizit auch durch x repräsentiert. Durch diesen Nebeneffekt korrigiert der Faktor x auch gleich implizit die Formel. Dies macht die Formel nach dem Hochskalieren auf die sequentielle Laufzeit so präzise und sorgt damit dafür, dass die $T(n,p)$ -Formel so exakt die untere Grenze der Laufzeitverbesserung vorhersagen kann.

Herleitung $T(n,p)$: Ausgangsdefinition

Gegeben ist die Rekursion:

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

Zusätzlich gilt:

$$p = 2^e$$

Herleitung $T(n,p)$: Entfaltung der Rekursion (1. Schritt)

Für $e > 0$ gilt:

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

Ein Einsetzen ergibt:

$$T(n, e) = T\left(\frac{n}{4}, e - 2\right) + \frac{n}{2} + n$$

Noch ein weiteres Einsetzen:

$$T(n, e) = T\left(\frac{n}{8}, e - 3\right) + \frac{n}{4} + \frac{n}{2} + n$$

Nach e Schritten ergibt sich:

$$T(n, e) = T\left(\frac{n}{2^e}, 0\right) + n \left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{e-1}}\right)$$

Herleitung $T(n,p)$: Auswertung der geometrischen Reihe

Die Summe

$$\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{e-1}}\right) = \sum_{i=0}^{e-1} \frac{1}{2^i}$$

ist eine geometrische Reihe mit Quotient $\frac{1}{2}$.

Sie ergibt:

$$\sum_{i=0}^{e-1} \frac{1}{2^i} = 2 \left(1 - \frac{1}{2^e}\right)$$

Damit folgt:

$$T(n, e) = T\left(\frac{n}{2^e}, 0\right) + 2n \left(1 - \frac{1}{2^e}\right)$$

Herleitung $T(n,p)$: Einsetzen des Basisfalls

Für $e = 0$ gilt:

$$T(n, 0) = 2T\left(\frac{n}{2}, 0\right) + n$$

Dies entspricht der Rekursion von sequenziellem Mergesort.
Bekannt ist:

$$T(n, 0) = n \log_2 n + n$$

Ersetzen von n durch $\frac{n}{2^e}$ ergibt:

$$T\left(\frac{n}{2^e}, 0\right) = \frac{n}{2^e} \log_2 \left(\frac{n}{2^e}\right) + \frac{n}{2^e}$$

Einsetzen in $T(n, e)$ liefert:

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

Herleitung $T(n,p)$: Substitution $p = 2^e$

Da gilt:

$$p = 2^e$$

folgt:

$$\frac{n}{2^e} = \frac{n}{p}$$

Damit ergibt sich:

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

Umsortiert erhält man:

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