

Untersuchung der Skalierbarkeit von parallelem Sortieren auf einem Multicore-Prozessor

Bachelorarbeit

Studiengang: Informatik

Bearbeiter: Leon Zoerner

7. Dezember 2025

Inhaltsverzeichnis

1	Einleitung	3
1.1	Motivation	3
1.2	Zielsetzung und Forschungsfrage	3
2	Theoretische Grundlagen	4
2.1	Sortieralgorithmen: Quicksort und Mergesort	4
2.2	Grundlagen der Parallelisierung	4
2.3	Thread-Modelle, Overheads und Skalierungsgrenzen	4
3	Methodik und Versuchsaufbau	5
3.1	Messumgebung und Hardware	5
3.2	Implementierungsvarianten	5
3.3	Messmethodik	5
4	Ergebnisse und Analyse	6
4.1	Grundlegende Laufzeiten abhängig von der Arraygröße	6
4.1.1	Messziel	6
4.1.2	Erwartung	6
4.1.3	Diagramm	6
4.1.4	Analyse und Interpretation	6
4.2	Vergleich der Threading-Methoden	7
4.2.1	Messziel	7
4.2.2	Erwartung	7
4.2.3	Diagramm	7
4.2.4	Analyse und Interpretation	7
4.3	Einfluss der Arraygröße im Detail	8
4.3.1	Messziel	8
4.3.2	Erwartung	8
4.3.3	Diagramm	8
4.3.4	Analyse und Interpretation	8
4.4	Workerthreads vs. neu erzeugte Threads	9
4.4.1	Messziel	9
4.4.2	Erwartung	9
4.4.3	Diagramm	9
4.4.4	Analyse und Interpretation	9
4.4.5	Overheads und Skalierungsgrenzen	9
4.5	Exakter Einfluss des Listentyps	10
4.5.1	Messziel	10
4.5.2	Erwartung	10
4.5.3	Diagramm	10
4.5.4	Analyse und Interpretation	10
5	Diskussion und Fazit	11
5.1	Interpretation aller Ergebnisse	11
5.2	Beantwortung der Forschungsfrage	11
5.3	Zusammenfassung	11

6	Anhang	12
6.1	Code	12
6.2	Exakte Hardware-Spezifikationen	12

1 Einleitung

1.1 Motivation

Ziel dieser Arbeit ist es, die Grenzen von Threads und Parallelisierung aufzuzeigen. Dabei soll insbesondere untersucht werden, wie groß der Overhead durch Threads ist und welchen Performanceunterschied es macht, bereits initialisierte Workerthreads zu verwenden, im Vergleich zur Erstellung neuer Threads. Da sich für diese Untersuchungen ein geeigneter, leicht verständlicher und programmierbarer Anwendungsfall anbietet, habe ich mich für Sortieralgorithmen entschieden, die sich zudem sehr gut parallelisieren lassen.

1.2 Zielsetzung und Forschungsfrage

Ziel dieser Bachelorarbeit ist die systematische Analyse der Laufzeitentwicklung paralleler Sortierverfahren. Dabei soll untersucht werden, wie sich parallele Implementierungen von Quicksort und Mergesort im Vergleich zu ihren sequentiellen Varianten verhalten. Im Fokus stehen insbesondere folgende Punkte:

- der Einfluss verschiedener Threadingstrategien auf die Laufzeit,
- die Frage, ab welcher Eingangsgröße und bei welcher Anzahl von Threads ein messbarer Geschwindigkeitsvorteil entsteht,
- sowie die Identifikation von Thread-Management-Techniken, die für Sortieralgorithmen die besten Laufzeiten erzielen.

Aus diesen Aspekten ergibt sich die zentrale Forschungsfrage dieser Arbeit:

Unter welchen Bedingungen liefern parallele Sortieralgorithmen anhand von Quicksort und Mergesort einen signifikanten Laufzeitvorteil gegenüber der sequentiellen Ausführung, und welche Threadingstrategien führen dabei zur besten Laufzeit?

2 Theoretische Grundlagen

2.1 Sortieralgorithmen: Quicksort und Mergesort

2.2 Grundlagen der Parallelisierung

2.3 Thread-Modelle, Overheads und Skalierungsgrenzen

3 Methodik und Versuchsaufbau

3.1 Messumgebung und Hardware

3.2 Implementierungsvarianten

3.3 Messmethodik

4 Ergebnisse und Analyse

4.1 Grundlegende Laufzeiten abhängig von der Arraygröße

4.1.1 Messziel

4.1.2 Erwartung

4.1.3 Diagramm

4.1.4 Analyse und Interpretation

4.2 Vergleich der Threading-Methoden

4.2.1 Messziel

4.2.2 Erwartung

4.2.3 Diagramm

4.2.4 Analyse und Interpretation

4.3 Einfluss der Arraygröße im Detail

4.3.1 Messziel

4.3.2 Erwartung

4.3.3 Diagramm

4.3.4 Analyse und Interpretation

4.4 Workerthreads vs. neueThreadsBisEbene

4.4.1 Messziel

4.4.2 Erwartung

4.4.3 Diagramm

4.4.4 Analyse und Interpretation

4.4.5 Overheads und Skalierungsgrenzen

4.5 Exakter Einfluss des Listentyps

4.5.1 Messziel

4.5.2 Erwartung

4.5.3 Diagramm

4.5.4 Analyse und Interpretation

5 Diskussion und Fazit

5.1 Interpretation aller Ergebnisse

5.2 Beantwortung der Forschungsfrage

5.3 Zusammenfassung

6 Anhang

6.1 Code

6.2 Exakte Hardware-Spezifikationen