Übungsblatt: Parallelisierung von Quicksort

Die Lernziele in diesem Praktikum sind

- Implementierung paralleler Algorithmen mit Threads.
- Abfragen der Linux-Fehlerbehandlung
- Erstellen und verwenden von Makefiles.

Das Sortieren eines großen int-Arrays soll durch eine Parallelisierung auf Threadebene beschleunigt werden. Hierbei soll der aus den Vorlesung OOP bekannt Algorithmus Quicksort zum Einsatz kommen.

In der Literatur wird eine Reihe von Varianten zur Parallelisierung von Quicksort beschrieben. Eine sehr einfache Variante lässt sich wie folgt realisieren (siehe Abb.):

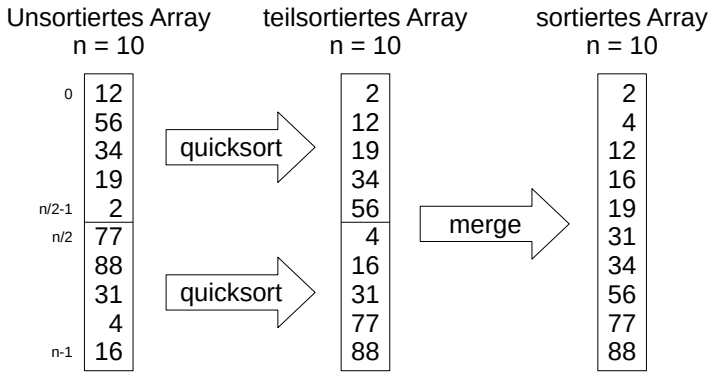

Abbildung 1: Paralleler Ablauf des Quicksort-Programms

1. Ein Prozess erzeugt ein großes int-Array.
2. Zwei Threads sortieren unabhängig voneinander die obere Hälfte (Indexbereich 0 bis n/2-1) bzw. die untere Hälfte (Indexbereich n/2 bis n-1) nach dem Quicksort-Verfahren.
3. Der Hauptprozess mischt anschließend die sortierten Hälften in ein separates Array von identischer Größe.

Aufgabe 1
Schreiben Sie ein C++-Programm für das Betriebssystem Linux welches einen parallelen Sortiervorgang realisiert.

Dazu soll ein Array mit 1.000.000 zufälligen `int`-Werten sortiert werden. Die Existenz von mehreren gleichen Werten ist zulässig.

Im export-Verzeichnis des SWT-Servers finden sie ein Modul `sort_merge` (`sort_merge.h` und `sort_merge.cpp`), welches die Funktionen für den Quicksort-Algorithmus und den Misch-Algorithmus bereit stellt.

Da ihr Programm sich nun aus zwei Quelldateien zusammensetzt, wird das Compilieren von der Kommandozeile komplizierter. Realisieren Sie ihr Programm daher unter Verwendung des

für einen Linux-Programmierer unverzichtbaren Programms `make`. Ein erklärender Abschnitt findet sich in den Praktikumsunterlagen.

Hinweise:

- Verwenden Sie zur Generierung der zufälligen int-Werte die Funktion `rand()` aus der C-Standard-Bibliothek `cstdlib`.
- Für das Mischen wird ein zweites Array zur Aufnahme der sortierten Zahlenfolge benötigt.
- Testen Sie das Programm zunächst mit einem kleineren Array (z.B. 10 `int`-Werte) und lassen Sie sich das Array zu Testzwecken im unsortierten, im teilsortierten und im sortierten Zustand ausgeben. (Die Ausgabe muss in der Endversion entfernt werden.)

Aufgabe 2
Verwenden Sie das von Ihnen erstellte Messprogramm `zeiten` zur Messung der Ausführungszeiten des parallelen Quicksort-Programms.

Verändern Sie das Programm so, dass die beiden Schritte zur Teilsortierung sequentiell hinter einander ausgeführt werden (siehe Abbildung).

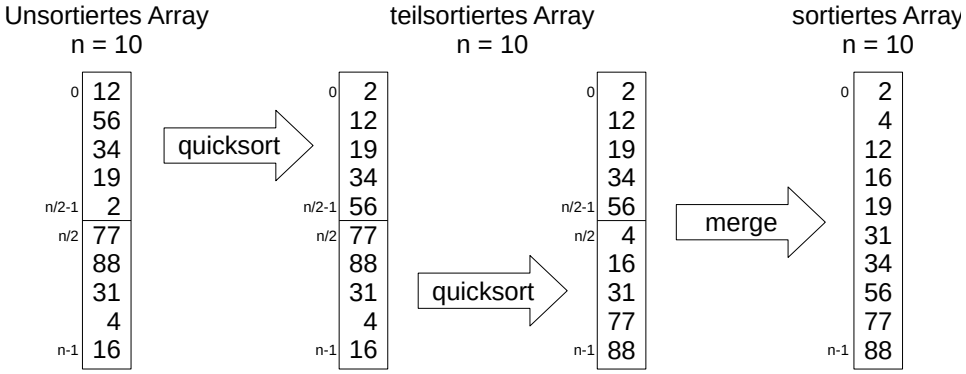

Abbildung 2: Sequentieller Ablauf des Quicksort-Programms

Vergleichen Sie die Ausführungszeiten der sequentiellen mit denen der parallelen Variante. Was ist Ihre Beobachtung?