Verteilte Softwaresysteme

Blatt 2

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 16.10.2018 12:37

Für die beiden Aufgaben auf diesem Blatt bekommen Sie ein GitHub-Repository über den Link https://classroom.github.com/a/aw6il7Sv.

Die beiden im Folgenden zu implementierenden Algorithmen sind in https://link.springer.com/book/10.1007/978-3-8274-2804-2 beschrieben. Sie können das Buch als PDF im VPN der Hochschule herunterladen.

Aufgabe 1 — paralleles Mergesort

Implementieren Sie den rekursive 2-Wege-Mergesort wie im o.a. Buch auf Seite 113ff beschrieben.

Die zu implementierende Funktion im Package `sorting` hat folgende Signatur:

```
func Mergesort(s []int) []int
```

Das übergebene Slice soll unverändert bleiben, ein sortiertes Slice als Ergebnis zurück gegeben werden.

Die eigentliche Sortierung soll in der Funktion

```
func mergesortParallel(s []int, c chan<- int)
```

durchgeführt werden. Der Parameter `s` ist das zu sortierende Teilslice, der Channel `c` soll für die sortierte Rückgabe genutzt werden. Jeder Rekursionsschritt kommuniziert mit seinem Aufrufer über einen eigenen Channel. Die `Mergesort`-Funktion befüllt dann ein neues Slice mit den Werten die sie über den Channel bekommt. Dazu muss natürlich jedes `mergesortParallel` als eigene Go-Routine ausgeführt werden.

Im Repository sind bereits einige Testfälle enthalten, die Sie natürlich gerne erweitern können.

Aufgabe 2 — paralleles Quicksort

Implementieren Sie das rekursive Quicksort-Verfahren, so wie im o.a. Buch auf Seite 93ff beschrieben. In diesem Fall soll das Slice *in-situ* sortiert, also tatsächlich verändert werden.

Die Quicksort-Funktion hat die Signatur

```
func Quicksort(s []int)
```

Die eigentliche Sortierung soll dann in der Funktion

```
func quicksortParallel(a []int, wg *sync.WaitGroup)
```

durchgeführt werden. Nachdem in dieser Aufgabe keine Kommunikation durch Channels erfolgt, sondern die einzelnen Instanzen von `quicksortParallel` einfach ihren Teil des Slices bearbeiten, nutzen Sie die `WaitGroup`, damit die jeweilige Funktion auf die Beendigung der von ihr gestarteten Go-Routinen warten kann.

Da in Go ein Slice nur ein Ausschnitt eines darunter liegenden Arrays ist, können Sie einfach das zu bearbeitende Slice übergeben und müssen nicht, wie im Buch beschrieben, den Bereich durch explizite Angabe der rechten und linken Grenze einschränken.

Im Repository sind bereits einige Testfälle enthalten, die Sie natürlich gerne erweitern können.