Verteilte Softwaresysteme

Blatt 3

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 28.11.2018 13:35

Abgabe bis 04.12.2018, 11:45 per Pull-Request gegen den `master`-Branch.

Aufgabe

Die fristgerechte Abgabe einer Lösung für diese Aufgabe ist notwendige Voraussetzung für den Leistungsnachweis.

Diese Aufgabe können Sie alleine oder zu zweit bearbeiten und abgeben. Über den GitHub-Classroom-Link https://classroom.github.com/g/JITYUVQ4 müssen Sie als erstes ein neues Team erstellen (als der/die Erste von beiden oder auch wenn Sie alleine arbeiten). Den Namen können Sie sich selbst aussuchen. Wenn Sie als ZweiteR einer Gruppe auf den Link gehen, müssen Sie dann dem bereits erzeugten Team beitreten. Der/die Erste hat bereits ein Repository bekommen, dem Sie dann beitreten werden.

Wenn Sie alleine arbeiten, dann gehen Sie jetzt nach dem Git Flow oder nach dem GitHub-Flow vor, zu zweit nach dem Git-Flow. Die Abgabe erfolgt in beiden Fällen als Pull-Request. Für mehr Informationen dazu siehe: https://ob.cs.hm.edu/exercises.html unter `Arbeiten, Abgabe und Feedback`. Schauen Sie sich das bitte als **Allererstes** an, denn es darf **keine** Commits im `master`-Branch geben.

Entwerfen und Implementieren Sie eine Go-Library für einen Blattsuchbaum, der folgende Eigenschaften hat.

Knoten

- Jeder Knoten läuft in einer eigenen Goroutine und kommuniziert ausschließlich über Channels, d.h. auch Nachfolger und Vorgänger sind nicht bekannt, aber jeweils Channels zu ihnen.
- Ein innerer Knoten hat zwei Nachfolger. Er speichert keine Werte, sondern nur den maximalen Schlüssel im linken Teilbaum als Wegweiser (siehe Blattsuchbaum).
- Ein Blatt speichert eine maximale Anzahl von Schlüssel-Wert-Paaren in einer `map`. Als Schlüssel verwenden Sie Werte vom Typ `int`. Für die Werte sollen alle Typen erlaubt sein (Stichwort: `interface{}`). Die maximale Anzahl soll für alle Blätter gleich sein und beim Nutzen der Bibliothek angegeben werden können.
- Wird ein Blatt zu voll, muss es durch einen inneren Knoten mit zwei Blättern, die jeweils ca. die Hälfte der Schlüssel-Wert-Paare enthalten, ersetzt werden.

API / Operationen

Die Bibliothek stellt folgende Schnittstelle zur Verfügung:

```
func NewTree(leafSize int, logFile string) Tree

type Tree interface {
    Search(key int) interface{}
    Insert(key int, value interface{})
    Delete(key int) (int, interface{})
    Traverse() []KeyValue
}
```

Die Struct `KeyValue` kapselt einfach nur den Schlüssel und den Wert um viele davon in einem Slice speichern zu können.

```
type KeyValue struct {
    Key   int
    Value interface{}
}
```

Logs

Zum Loggen und Debuggen implementieren Sie eine weitere Goroutine, die über einen Channel Messages (`strings` sind ausreichend) annimmt und diese mit einem Timestamp in ein Logfile schreibt, z.B. so:

```
2018-10-28 20:31:08: new master constructed
2018-10-28 20:31:08: master entering the loop
```

```
2018-10-28 20:31:13: master inserts {0 {12 Hallo}}
2018-10-28 20:31:13: creating first leaf
2018-10-28 20:31:13: leaf up and running
2018-10-28 20:31:13: leaf inserted (12, Hallo)
2018-10-28 20:31:22: master inserts {0 {18 Welt}}
2018-10-28 20:31:22: leaf inserted (18, Welt)
2018-10-28 20:31:22: leaf full after inserting (18, Welt)
2018-10-28 20:31:22: sending {0 {12 Hallo}} to left leaf
2018-10-28 20:31:22: leaf up and running
2018-10-28 20:31:22: leaf up and running
2018-10-28 20:31:22: leaf inserted (12, Hallo)
2018-10-28 20:31:22: sending {0 {18 Welt}} to right leaf
2018-10-28 20:31:22: new node up and running
2018-10-28 20:31:22: leaf inserted (18, Welt)
```

In dem Beispiel kann jedes Blatt maximal ein Schlüssel-Wert-Paar speichern.

Vorgehen bei der Softwareentwicklung

Gehen Sie, wie oben bereits erwähnt, nach dem Git Flow oder dem GitHub Flow vor. Committen und pushen Sie häufig mit sinnvollen Commit-Messages. Schreiben Sie Dokumentation und Unittests, die Sie auf dem Travis CI ausführen lassen.

Nutzen Sie eine `README.md` um Ihre Bibliothek und die Nutzung kurz zu beschreiben. Formatieren Sie die Datei in Markdown.

Schreiben Sie eine einfache Anwendung, mit deren Hilfe Sie Ihre Bibliothek vorführen können. Es reicht z.B. eine einfache Kommandozeilen-Anwendung über die Sie mit dem Baum interagieren können und dazu das Logfile in einem weiteren Terminal zeigen können.

Ein Beispiel für eine Lösung

Wenn Sie Docker[1] nutzen, können Sie meinen Lösungsvorschlag mit folgendem Kommando ausprobieren (erzeugen Sie **vorher** im aktuellen Verzeichnis das Unterverzeichnis `log`):

```
docker run --rm -it --mount type=bind,source="$(PWD)/log",target=/app/log
    obraun/vss-solutionb3a1:latest 2 log/logger.log
```

Das gesamte Kommando muss auf einer Zeile eingegeben werden. Vor `rm` und `mount` steht jeweils ein doppeltes Minuszeichen. Die `2` am Ende steht für die maximale Größe

[1] Docker ist übrigens in Go implementiert.

der Blätter und `logger.log` ist die Datei, in die in das Unterverzeichnis `log` die Logs geschrieben werden. Beides können Sie natürlich auch beliebig ersetzen.

Das Kommando steht bereits in dem `Makefile`, so dass, wenn Sie Make auf Ihrem Rechner installiert haben, folgende Eingabe reicht:

```
make docker-run
```

Auf den MWN-PCs ist aktuell leider kein Docker installiert. Sie können es sich in dem virtuellen Linux selbst dazu installieren und ausprobieren.

Mein Vorschlag ist nur **eine** Möglichkeit die Aufgabe zu lösen. Sie können und sollten Ihre eigenen Designentscheidungen treffen.

Abnahme

Im Rahmen des Praktikums erfolgt die Abnahme durch eine kurze Vorführung und Erklärung Ihrer Bibliothek auf dem Beamer.