Aufgaben zur verteilten Berechnung mit Hilfe von Aktoren

TECHNISCHE UNIVERSITÄT DARMSTADT

Hier wird ein Beispielprogramm entwickelt, welches Aktoren nutzt um den Wert e zu berechnen. Die Grundidee ist es mit einem Masteraktor die Berechnung zu initialisieren und Workeraktoren zu erstellen. Der Master teilt die Arbeit in Chunks auf und verteilt diese an die Worker. Anschließend wartet er darauf, dass die Worker ihre Teilergebnisse zurück senden und rechnet diese zusammen. Wenn das Ergebnis komplett ist, sendet der Master das Ergebnis an einen Printer, welcher dieses Ausgibt und das ActorSystem beendet.

Aufgabe 1 Berechnung von e

Aufgabe 1.1 Package und Messages

Legen Sie einen Ordner entsprechend Ihrem Namen in excercises/tutorial an. Kopieren Sie den Akka-Tutorial Ordner nach dort. Erweitern Sie das package de.tkip.akkatutorial. Erstellen Sie case-Klassen für die benötigten Messages.

Diese sind:

- Calculate - wird an den Master gesendet, um die Berechnung zu starten
- Work - wird vom Master an die Worker gesendet und enthält den zu berechnenden Chunk
- Result - wird von den Workern an den Master geschickt mit dem Ergebnis der Berechnung
- eApproximation - wird vom Master nach erfolgreicher Berechnung an einen Printer geschickt, der das Ergebnis ausgibt

Aufgabe 1.2 Der WorkerActor

Legen Sie die Klasse Worker an, die den Aktor Trait erweitert und definieren Sie die receive Methode. Diese soll bei Empfang einer Work-Message einen Teil von e berechnen und das Ergebnis an den Master zurück schicken. Zur Erinnerung: $e = \sum_{n=0}^{\infty} \frac{1}{n!}$

Aufgabe 1.3 Der MasterActor

Schreiben Sie nun die Klasse Master. In seinem Konstruktor sollen bereits eine Anzahl Worker angelegt und gestartet werden. Außerdem sollen diese zur Vereinfachung der Verteilung der Arbeit in einen load-balancing Router zusammengefasst werden. Der Master soll die Berechnung von e auf die Worker verteilen und am Ende die Approximation von e an einen Printer schicken.

Die folgenden imports könnten hilfreich sein:

```
import akka.actor.{Actor, ActorRef, Props}
import akka.routing.RoundRobinRouter
```

Der Konstruktor kann die folgenden Parameter enthalten:

- nrOfWorkers – Wie viele Worker sollen verwendet werden
- nrOfMessages – Wie viele Chunks sollen an die Worker verteilt werden
- nrOfElements – welchen Umfang sollen die Chunks haben

Aufgabe 1.4 Der PrinterActor

Schreiben Sie einen PrinterActor, der bei Empfang einer eApproximation das Ergebnis ausgibt und das ActorSystem beendet.

Aufgabe 1.5 Bootstrap Applikation

Erweitern Sie das Applikations Object um die Erzeugung der zum Start erforderlichen Elemente und dem Start der Kalkulation.

Aufgabe 2 Erweiterung um Futures

Nun sollen Sie das Akka Ask-Pattern benutzen um die Resultate zu empfangen. Schauen Sie sich zuerst noch mal das Dokument ”Futures und Routing” an.

Aufgabe 2.1 Ask-Pattern

Ihre bisherige Lösung versendet durch eine Schleife die Chunks. Passen Sie die Schleife so an, dass diese mit dem Ask-Pattern Futures mit dem Typ Result zurückgibt.

Reduzieren Sie die erhaltene Sequenz von Futures auf ein einiges Future[Result].

Aufgabe 2.2 PipeTo-Pattern

Verwenden Sie nun das pipeTo-Pattern um das reduzierte Result an den PrinterActor zu senden. Passen Sie den PrinterActor so an, dass dieser Result Nachrichten anstelle der eApproximation empfängt.