Grundlagen Verteilter Systeme

Institut für Verteilte Systeme | Wintersemester 2019/2020
David Mödinger, Prof. Dr.-Ing. Franz Hauck

Übungsblatt 3: Java Executor Framework

Abgabetermin: 12.12.2019, 03:59Uhr

Aufgabe 1: Miningpool: Threaded Computational Race

Ziel der Aufgabe ist es, ein Gerät in einem Miningpool zu programmieren.

Ihre Lösung für diese Aufgabe hat die folgenden Randbedingungen:

1. Unter *gvs.lxd-vs.uni-ulm.d:27341* läuft ein ZeroMQ Publisher, welcher Strings versendet. Subscriben Sie zu diesem Dienst.
2. Unter *gvs.lxd-vs.uni-ulm.d:27349* läuft ein ZeroMQ Reply Socket, senden Sie Lösungen an diesen Dienst.
3. Bestimmen Sie einen String s, wobei der String s und der Hash des Strings $H(s)$ (in Hexadezimaler darstellung) beide mit vom Publisher erhaltenen Wert beginnen.
4. Partitionieren Sie den Suchraum um alle vorhandenen Kerne der Maschine ihres Programms auszunutzen.

Die erhaltenen Strings enthalten nur Zeichen die in der Hexadezimalen Darstellung möglich sind (0-9 und a-f).

Beispiel: Der Publish Dienst sendet ihnen den String *8c2bf*. Sie probieren nun einige Hashes durch, bis Sie den String *8c2bf273824* finden. Es gilt:

$$SHA_{256}(8c2bf273824) = 8c2bf3cda2ff951b33bd19bc55a7acb246925d50cef448045ee2b3ca40622a1c.$$

Sie reichen also *8c2bf273824* als Lösung ein.

Aufgabe 2: Miningpool as Distributed System

Ziel der Aufgabe ist es einen Miningpool zu programmieren. Die basis bildet derselbe Server mit Publish und Reply Socket wie in Aufgabe 1.

Erstellen Sie folgende Programme:

1. Einen Controler, welcher sich zum System Subscribed und die eingehenden Aufträge partitioniert und weitergibt.
2. Einen Submitter, welcher die Ergebnisse an den Server weiterleitet.
3. Worker, welche in beliebiger Zahl gestartet werden können, und sich bei ihrem Controler für Aufgaben melden, die Hash Berechnung durchführen und bei gefundenem Ergebnis weiterreichen. (Worker sollten Multithreaded sein.)

Submitter und Controler können in ihrem Design daselbe Programm sein. Worker müssen als eigenständiges Programm gestartet werden.

Hinweise:

- Verwenden Sie für die generierung der Hash-Strings den Beispielcode aus dem Moodle.
- Der Server publiziert mehrfach daselbe Rätsel, erst wenn er eine Lösung erhalten hat, publiziert er ein weiteres!
- Eine mögliche Lösung bieten ZeroMQ Push/Pull Sockets:
- Im Beispielaufbau mit Push/Pull Sockets wird ein weiterer Publish-Subscribe Socket verwendet um die Worker über zu ignorierende Aufträge zu informieren, da Push/Pull Sockets keine Aufträge zurücknehmen können.