HOCHSCHULE KONSTANZ
TECHNIK, WIRTSCHAFT UND GESTALTUNG

Verteilte Systeme
Übung

30. Mai 2018

Aufgabenblatt 3:
Wartezeiten, Load Balancing und RMI

Abgabe bis: 27.06.2016

Prof. Dr. Rainer Mueller

SS 2018

Aufgabe 1: Zeitmessung

1

Diese Aufgabe baut auf der Aufgabe 3 des Aufgabenblatts 2 auf. Sie sollen hierbei ein Gefühl dafür bekommen, wie sich die Antwortzeiten des Servers zusammensetzen und wie diese Zusammensetzung bei unterschiedlichen Nebenläufigkeitsgraden ausfällt.

a. Verändern Sie den Client und den Server so, dass pro Server-Anfrage folgende Zeiten in *ms* gemessen werden:

- Processing Time [p]: Zeit, die für die Berechnung der Primzahleigenschaft benötigt wird
- Waiting Time [w]: Zeit von Eingang der Anfrage bis zum Beginn der Berechnung der Primzahleigenschaft
- Communication Time [c]: Restzeit=Gesamtzeit-Processing Time-Waiting Time, wobei die Gesamtzeit die Zeitspanne vom Zeitpunkt der Serveranfrage beim Client zum Empfang der Antwort beim Client ist.

Geben Sie dabei nach jeder Anfrage die Zeiten für diese Anfrage und auch die Durchschnittszeiten über alle bisherigen Anfragen an, wie etwa in folgendem Beispiel:

```
100000000000000012:  not prime | p: 0 (1012) ms | w: 2720 (234) ms | c: 0 (5) ms
100000000000000013:  prime | p: 12719 (1239) ms | w: 0 (219) ms | c: 10 (6) ms
```

b. Experimentieren Sie mit den verschiedenen Nebenläufigkeitsgraden des Servers und auch mit den verschiedenen Anfragemodi des Clients. Verwenden mehrere nebenläufige Clients und Threadpools mit einer Größe, die geringer als die Client-Anzahl ist. Nennen Sie Gründe für die Unterschiede bei den jeweiligen Zeiten p, c und vor allem w.

Aufgabe 2: Load Balancer

2

a. Erweitern Sie den Server aus Aufgabe 1 so um Anwendereingabemöglichkeiten, dass der (Empfänger-)Port, ggf. auch der Nebenläufigkeitsgrad konfigurierbar sind, zum Beispiel:

```
Port [1234] > 2000
Concurrency mode [FIX_POOLED] > synchronized
```

Sie können die Konfigurierbarkeit auch über Kommandozeilenparameter oder eine Konfigurationsdatei realisieren, aber für die nachfolgenden Tests innerhalb einer IDE (Eclipse) eignet sich die Anwendereingabe sicher besser.

b. Verwenden Sie mehrere Instanzen des Servers aus Aufgabenteil a., um einen Server-Cluster zu realisieren. Entwickeln Sie einen Load Balancer, der anstelle der Server die Client-Anfragen nebenläufig entgegen nimmt und gleichmäßig auf die Server des Clusters verteilt.
Nutzen Sie im Load Balancer zur Realisierung der Server-Verwaltung und der Cluster-Konfiguration das bereitgestellte Java-Archiv `rm.serverAdmin.jar`.
Die Klasse `serverAdmin` liest dabei die Konfigurationsdaten für den Cluster aus einer Konfigurationsdatei, zum Beispiel:

```
localhost 2000 2100
localhost 2001 2101
localhost 2002 2102
```

Dabei steht jede Zeile für einen Server des Clusters, der hier beispielsweise bei der ersten Zeile unter dem DNS-/Host-Namen `localhost` erreichbar ist und auf Port `2000` empfängt bzw. auf Port `2100` antwortet.

Aufgabe 2: Load Balancer

2

...b.

Die genaue Beschreibung von `rm.serverAdmin.jar` finden Sie im zugehörigen Javadoc `rm.serverAdmin.javadoc.jar`.
Hinweis: Beachten Sie unbedingt die Verwendung der Methode `cleanUp` der Klasse `Component`, um blockierte Ports rechtzeitig wieder freizugeben.

Demonstrieren Sie die Funktionsweise des Load Balancers mit folgendem Setup:

- Cluster mit 3 Primzahlservern
- Load Balancer
- 4 Clients

c. Implementieren Sie die Klassen des Java-Archivs `rm.serverAdmin.jar`. Orientieren Sie sich zur Spezifikation der einzelnen Methoden ggf. am Javadoc zum Archiv.
Hinweis: Beachten Sie bei der Implementierung die nebenläufige Verwendung der Klassen.

Demonstrieren Sie die korrekte Funktionsweise Ihrer Implementierung erneut am in Aufgabenteil b. beschriebenen Setup.

d. OPTIONAL (Dieser Aufgabenteil muss nicht abgegeben werden, um den Schein zur Veranstaltung zu bekommen)
Erweitern Sie die Zeitmessung aus Aufgabe 1 so, dass sie auch bei Verwendung des Load Balancers korrekte Werte liefert. Testen Sie ein real verteiltes Setup, in dem Sie die Server des Clusters und auch den Load Balancer auf unterschiedliche Maschinen verteilen.

...

Aufgabe 2: Load Balancer

2

…d.

Vergleichen Sie die Werte der Zeitmessung in diesem verteilten Setup mit den Werten aus dem vergleichbaren zentralisierten Setup ohne Load Balancer in Aufgabe 1. Interpretieren Sie die ggf. auftretenden Wertunterschiede.

Aufgabe 3: Kommunikationsgrade und RMI

Bauen Sie die Lösung der Aufgabe 3 auf Aufgabenblatt 2 so um, dass die gesamte über das Package `rm.requestResponse` realisierte Kommunikation zwischen Client und Server durch RMI-basierte Kommunikation ersetzt wird.

Hinweis: Beachten Sie die Nebenläufigkeitsgrade des Servers und die verschiedenen Anfrage-Modi des Clients.

Wenn Sie es etwas anspruchsvoller mögen, können Sie an Stelle der Lösung von Aufgabe3/Aufgabenblatt 2 auch gerne die Lösung von Aufgabe 2 auf diesem Aufgabenblatt nehmen.