DezSys

PiCalculator

RMI

FOCK & WORTHA

Inhalt

Aufgabenstellung

Distributed PI Calculator

Als Dienst soll hier die beliebig genaue Bestimmung von pi betrachtet werden. Der Dienst stellt folgendes Interface bereit:

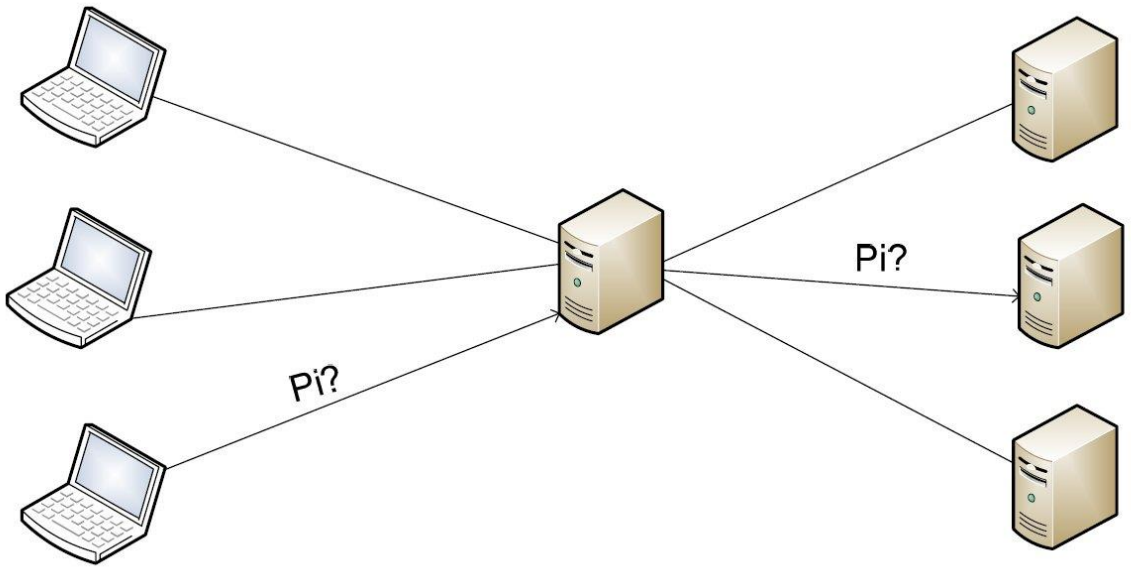

Abbildung 1 Client-Balancer-Server

```
1  // Calculator.java
2  public interface Calculator {
3          public BigDecimal pi (int anzahl_nachkommastellen);
4  }
```

Ihre Aufgabe ist es nun, zunächst mittels Java-RMI die direkte Kommunikation zwischen Klient und Dienst zu ermöglichen und in einem zweiten Schritt den Balancierer zu implementieren und zwischen Klient(en) und Dienst(e) zu schalten. Gehen Sie dazu folgendermassen vor:

1. Ändern Sie Calculator und CalculatorImpl so, dass sie über Java-RMI von aussen zugreifbar sind. Entwicklen Sie ein Serverprogramm, das eine CalculatorImpl-Instanz erzeugt und beim RMI-Namensdienst registriert. Entwicklen Sie ein Klientenprogramm, das eine Referenz auf das Calculator-Objekt beim Namensdienst erfragt und damit pi bestimmt. Testen Sie die neu entwickelten Komponenten.
2. Implementieren Sie nun den Balancierer, indem Sie eine Klasse CalculatorBalancer von Calculator ableiten und die Methode pi() entsprechend implementieren. Dadurch verhält sich der Balancierer aus Sicht der Klienten genauso wie der Server, d.h. das Klientenprogramm muss nicht verändert werden. Entwickeln Sie ein Balanciererprogramm, das eine

CalculatorBalancer-Instanz erzeugt und unter dem vom Klienten erwarteten Namen beim Namensdienst registriert. Hier ein paar Details und Hinweise:

- Da mehrere Serverprogramme gleichzeitig gestartet werden, sollten Sie das Serverprogramm so erweitern, dass man beim Start auf der Kommandozeile den Namen angeben kann, unter dem das CalculatorImpl-Objekt beim Namensdienst registriert wird. dieses nun seine exportierte Instanz an den Balancierer übergibt, ohne es in die Registry zu schreiben. Verwenden Sie dabei ein eigenes Interface des Balancers, welches in die Registry gebinded wird, um den Servern das Anmelden zu ermöglichen.
- Das Balancierer-Programm sollte nun den Namensdienst in festgelegten Abständen abfragen um herauszufinden, ob neue Server Implementierungen zur Verfügung stehen.
- Java-RMI verwendet intern mehrere Threads, um gleichzeitig eintreffende Methodenaufrufe parallel abarbeiten zu können. Das ist einerseits von Vorteil, da der Balancierer dadurch mehrere eintreffende Aufrufe parallel bearbeiten kann, andererseits müssen dadurch im Balancierer änderbare Objekte durch Verwendung von synchronized vor dem gleichzeitigen Zugriff in mehreren Threads geschützt werden.
- Beachten Sie, dass nach dem Starten eines Servers eine gewisse Zeit vergeht, bis der Server das CalculatorImpl-Objekt erzeugt und beim Namensdienst registriert hat sich beim Balancer meldet. D.h. Sie müssen im Balancierer zwischen Start eines Servers und Abfragen des Namensdienstes einige Sekunden warten.

Testen Sie das entwickelte System, indem Sie den Balancierer mit verschiedenen Serverpoolgrössen starten und mehrere Klienten gleichzeitig Anfragen stellen lassen. Wählen Sie die Anzahl der Iterationen bei der Berechung von pi entsprechend gross, sodass eine Anfrage lang genug dauert um feststellen zu können, dass der Balancierer tatsächlich mehrere Anfragen parallel bearbeitet.

Gruppenarbeit

Die Arbeit ist als 2er-Gruppe zu lösen und über das Netzwerk zu testen! Nur localhost bzw. lokale Testzyklen sind unzulässig und werden mit 6 Minuspunkten benotet!

Benotungskriterien

o 12 Punkte: Java RMI Implementierung (siehe Punkt 1)
o 12 Punkte: Implementierung des Balancers (siehe Punkt 2)

o davon 6 Punkte: Balancer

o davon 2 Punkte: Parameter - Name des Objekts
o davon 2 Punkte: Listing der Server (dyn. Hinzufügen und Entfernen)
o davon 2 Punkte: Testprotokoll mit sinnvollen Werten für Serverpoolgröße und Iterationen

Quellen

An Overview of RMI Applications, Oracle Online Resource,http://docs.oracle.com/javase/tutorial/rmi/overview.html (last viewed 28.11.2014)

Requirements-Analyse

RMI-Implementierung

- UML
- Server
- Client
- Pi-Berechnung
- Beim Namensdienst registrieren
- Testen

Balancer

- Balancerklasse (Server zuweisen)
- Auf neue Server abfragen
- Testen

Zeitaufzeichnung

Beschreibung	Geschätzte Zeit	Aktuelle Zeit
UML	1h	Fock ➔ 1h Wortha ➔ 1h
Java RMI Implementierung • Server • Client	3h	Fock ➔ 1h Wortha ➔ 1h
Balancer Implementierung	4h	Fock ➔ 1 ½ h Wortha ➔ 1 ½ h
UML nachgebessert	½ h	Fock ➔ ½ h Wortha ➔ ½ h
Testen	1h	Fock ➔ ½ h Wortha ➔ 1h
Dokumentation	2h	Fock ➔ 1h Wortha ➔ ¾ h
Gesamt	11 ½ h	Fock ➔ 5 ½ h Wortha ➔ 5 ¾ h

UML

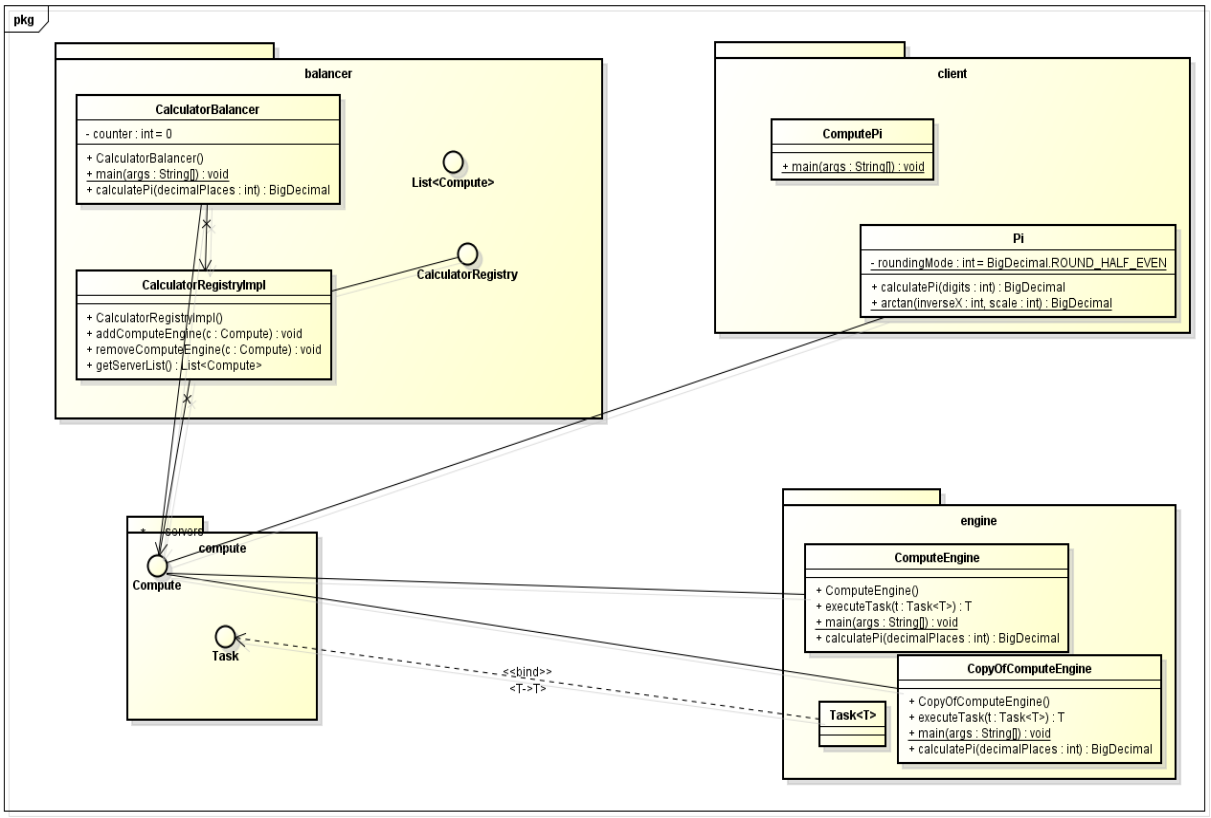

Abbildung 2 UML

Umsetzung

Allgemein

Beim RMI ist der Server/ BerechnungsMaschine ein relativ simples Programm. Dieser rechnet in unserem Fall nur PI aus und liefert dieses als BigDezimal zurück.
Der Client ist ein wenig komplexer. Dieser muss erstmal schaffen den Server/BerechnungsMaschine zu erreichen und muss auch noch die Aufgabe definieren, welche gemacht werden muss.

RMI-Server

- Der Server akzeptiert Aufgaben von Clients, welche er vom Balancer weitergeleitet bekommt bei dem er sich angemeldet hat. Diese Aufgaben führt er aus und übermittelt dann ein Resultat.
  Der Servercode besteht aus
  - Interface
    - Das Interface definiert die Methoden welche die Clients ausführen müssen um mit dem Server zu interagieren.

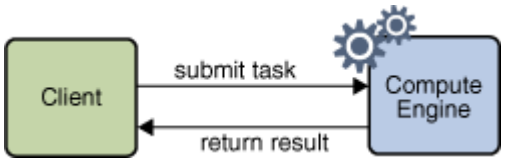

Abbildung 3 RMI-Server-Interface [1]

  - Klasse
    - Die Klasse enthält die Implementation
    - Die Pi Implementation wurde von dem Oracle Tutorial [1] übernommen.

RMI-Client

- Der Client schickt eine Anfrage/Aufgabe an den Balancer, welcher ihn an einen Server vermittelt, und danach wartet er nur mehr auf das Ergebnis.

Balancer

- Der Balancer erstellt ein Register in dem er alle Server, welche sich bei ihm anmelden abspeichert. Danach leitet er die Anfragen an die angemeldeten Server weiter und versucht die Auslastung einzelner Server zu verringern.

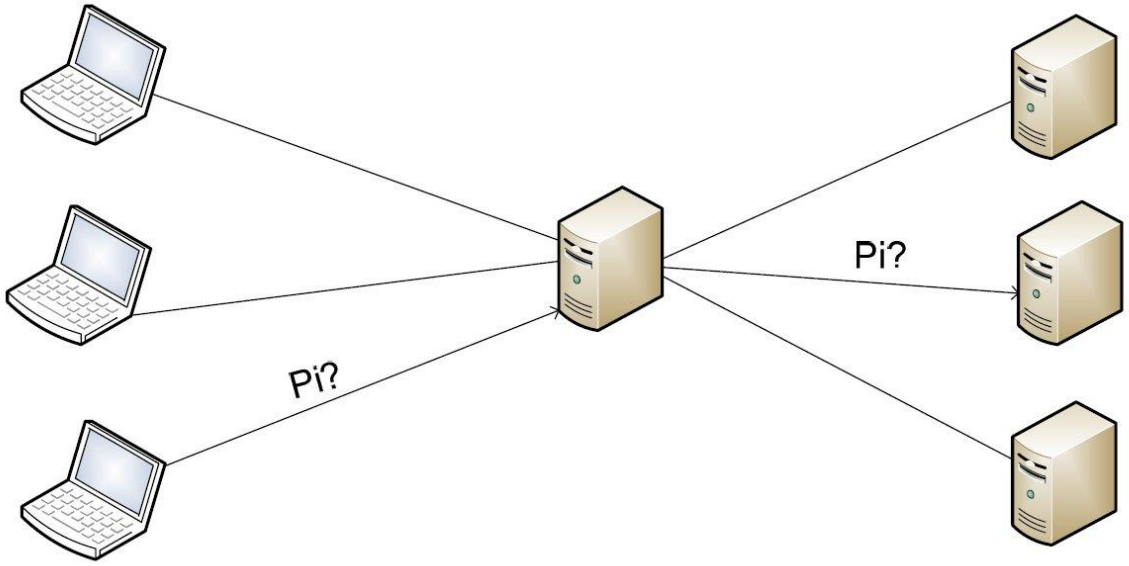

Abbildung 4 Client-Balancer-Server [2]

Testing

CalculatorBalancer wurde nicht getestet, da der größte Teil des Codes aus dem Tutorial genommen wurde und hier die meisten Schritte in der Main-Methode ablaufen (und wir arbeitscheu waren diesen Teil zu ändern).

Quellen

- [1] Tutorial für die RMI Applikation
  - http://docs.oracle.com/javase/tutorial/rmi/overview.html
    - Abgerufen am 12.12.2014
- [2] Aufgabe PiCalculator
  - https://elearning.tgm.ac.at/mod/assign/view.php?id=30895
    - Abgerufen am 08.01.2015

Abbildungsverzeichnis