1/9/2015

Christian Janeczek, Wolfgang Mair

5AHITT

DEZSYS05

Load Balancing

Inhalt

[Task Description 2](#_Toc408558428)

[Aufgabenstellung 2](#_Toc408558429)

[Auslastung 2](#_Toc408558430)

[Tests 2](#_Toc408558431)

[Modalitäten 2](#_Toc408558432)

[Design consideration 3](#_Toc408558433)

[RMI 3](#_Toc408558434)

[Strategy Pattern 3](#_Toc408558435)

[Description of the application 4](#_Toc408558436)

[UML-Diagrams 5](#_Toc408558437)

[Test Cases 6](#_Toc408558438)

[Technology Description 7](#_Toc408558439)

[Load Balancing 7](#_Toc408558440)

[Weighted Round-Robin 7](#_Toc408558441)

[Least Connection 7](#_Toc408558442)

[Least Connected Slow Start-Time 7](#_Toc408558443)

[Weighted Least Connection 7](#_Toc408558444)

[Agent Based Adaptive Balancing / Server Probes 7](#_Toc408558445)

[Apportionment of work with effort estimation 8](#_Toc408558446)

[Final Time Apportionment 9](#_Toc408558447)

[Sources 10](#_Toc408558448)

# Task Description

## Aufgabenstellung

Es soll ein Load Balancer mit mindestens 2 unterschiedlichen Load-Balancing Methoden (jeweils 7 Punkte) implementiert werden (ähnlich dem PI Beispiel [1]; Lösung zum Teil veraltet [2]). Eine Kombination von mehreren Methoden ist möglich. Die Berechnung bzw. das Service ist frei wählbar!  
  
Folgende Load Balancing Methoden stehen zur Auswahl:

* Weighted Round-Round
* Least Connection
* Least Connected Slow- Start Time
* Weighted Least Connection
* Agent Based Adaptive Balancing / Server Probes

Um die Komplexität zu steigern, soll zusätzlich eine "Session Persistence" (2 Punkte) implementiert werden.

## Auslastung

Es sollen die einzelnen Server-Instanzen in folgenden Punkten belastet werden können:

* Memory (RAM)
* CPU Cycles
* I/O Zugriff (Harddisk)

Bedenken Sie dabei, dass die einzelnen Load Balancing Methoden unterschiedlich auf diese Auslastung reagieren werden. Dokumentieren Sie dabei aufkommenden Probleme ausführlich.

## Tests

Die Tests sollen so aufgebaut sein, dass in der Gruppe jedes Mitglied mehrere Server fahren und ein Gruppenmitglied mehrere Anfragen an den Load Balancer stellen. Für die Abnahme wird empfohlen, dass jeder Server eine Ausgabe mit entsprechenden Informationen ausgibt, damit die Verteilung der Anfragen demonstriert werden kann.

## Modalitäten

Gruppenarbeit: 2 Personen  
Abgabe: Protokoll mit Designüberlegungen / Umsetzung / Testszenarien, Sourcecode (mit allen notwendigen Bibliotheken), Java-Doc, Jar  
  
  
Viel Erfolg!

# Description of the application

Die Aufgabe dieses Programmes ist es die Funktion eines LoadBalancer darzustellen. Dabei sollen sich Clients auf Server verbinden, welche wiederum Methoden aufrufen die Ressourcen auf dem Server verbrauchen. Diese Verbindungen funktioniert nur dann wenn ein LoadBalancer erstellt wird. Auf diesem LoadBalancer werden Server angemeldet. Die Server melden sich nach der Erzeugung automatisch an. Die Clients melden sich nachdem sie erzeugt wurden ebenfalls automatisch bei dem LoadBalancer und verlangen eine RMI-Adresse eines Servers. Die verschiedenen Ressourcen die bei den einzelnen Servern durch die Clients verbraucht werden sind: CPU, RAM und IO. Die Methoden des Servers verbrauchen bestimmte Ressourcen die nach dem Vorgang wieder freigelegt werden und dann wieder verbraucht werden. Dieser Vorgang wird solange durchgeführt bis der Client, der die Methode ausführt, die Verbindung beendet. Der LoadBalancer benutzt in diesem Programm zwei mögliche Verteil-Algorithmen (Least Connection, Weighted Round Robin) die bei der Erstellung des LoadBalancer definiert werden.

# Design consideration

## RMI

Wir werden RMI benützen, um die Clients mit den Servern und LoadBalancer kommunizieren zu lassen. Die Informationen, wie den Namen zum Aufrufen der Methoden vom LoadBalancer, bekommen die Clients und Server bei ihrer Erstellung. Der LoadBalancer besitzt 2 Methoden die mittels RMI aufrufbar sind. Eine Methode wird vom Server aufgerufen und speichert dessen RMI Adresse im LoadBalancer ab. Die zweite Methode wird vom Client aufgerufen und gibt diesen die RMI-Adresse eines Servers. Der Server bietet ebenfalls eine RMI-Methode an die es dem Client erlaubt Ressourcen auf dem Server zu verbrauchen.

[Bild RMI]

## Strategy Pattern

Um das Programm dynamisch erweiterbar zu machen benutzen wir 2-mal das Strategy Pattern. Einmal bei den Servern und einmal bei dem LoadBalancer. Mittels Strategy Pattern bestimmen wir welche Ressourcen der Server bei dem Methodenaufruf verbraucht. Außerdem bestimmen wir durch das Pattern welches Verteilungssystem der LoadBalancer benutzt.

[Bild Strategy-Server] [Bild Strategy-LoadBalancer]

# Technology Description

## Load Balancing

### Weighted Round-Robin

### Least Connection

### Least Connected Slow Start-Time

### Weighted Least Connection

### Agent Based Adaptive Balancing / Server Probes

# UML-Diagrams

# Test Cases

# Apportionment of work with effort estimation

Excel lässt Grüßen

# Final Time Apportionment

Excel lässt Grüßen

# Sources

[1] "Praktische Arbeit 2 zur Vorlesung 'Verteilte Systeme' ETH Zürich, SS 2002", Prof.Dr.B.Plattner, übernommen von Prof.Dr.F.Mattern (<http://www.tik.ee.ethz.ch/tik/education/lectures/VS/SS02/Praktikum/aufgabe2.pdf>)  
[2] <http://www.tik.ee.ethz.ch/education/lectures/VS/SS02/Praktikum/loesung2.zip>