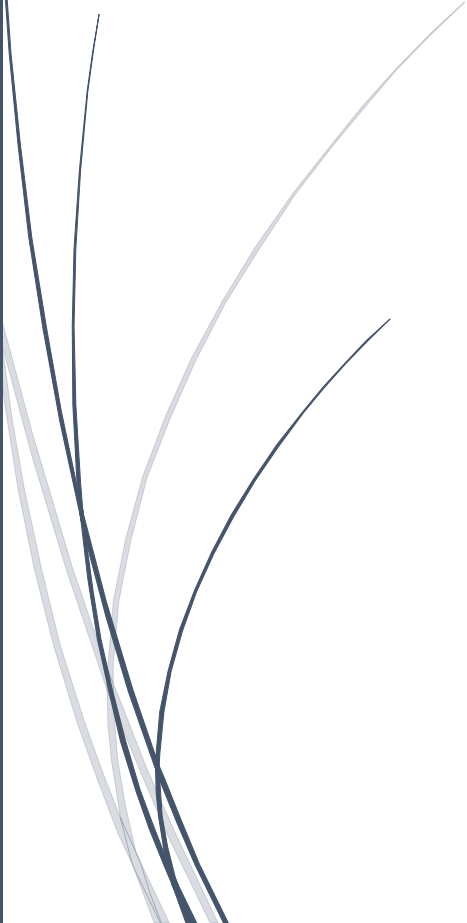


A dark blue vertical bar runs down the left side of the slide. A blue arrow points to the right from this bar, containing the date.

1/15/2015

DEZSYS05

Load Balancing

Several thin, curved lines in dark blue and light grey originate from the bottom left and sweep upwards and to the right.

Christian Janeczek, Wolfgang Mair
5AHITT

Inhalt

Task Description	2
Aufgabenstellung.....	2
Auslastung	2
Tests	2
Modalitäten.....	2
Description of the application	3
Design consideration	4
RMI	4
Strategy Pattern	4
Technology Description.....	6
Load Balancing.....	6
Weighted Round-Robin	6
Least Connection	6
Least Connected Slow Start-Time	6
Weighted Least Connection	6
Agent Based Adaptive Balancing / Server Probes	6
UML-Diagram	7
Test Cases	8
Apportionment of work with effort estimation	9
Christian Janeczek	9
Wolfgang Mair.....	9
Summe.....	9
Sources	10

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!

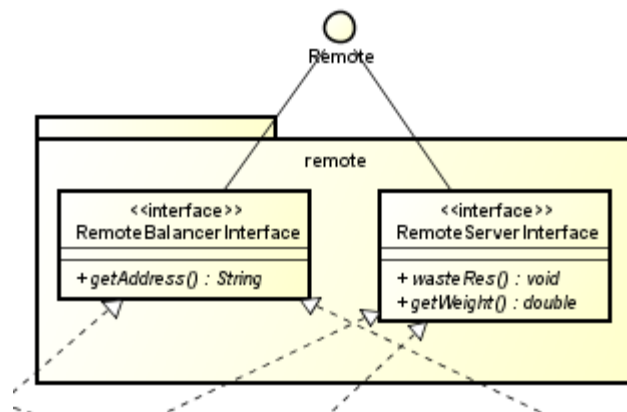
Description of the application

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

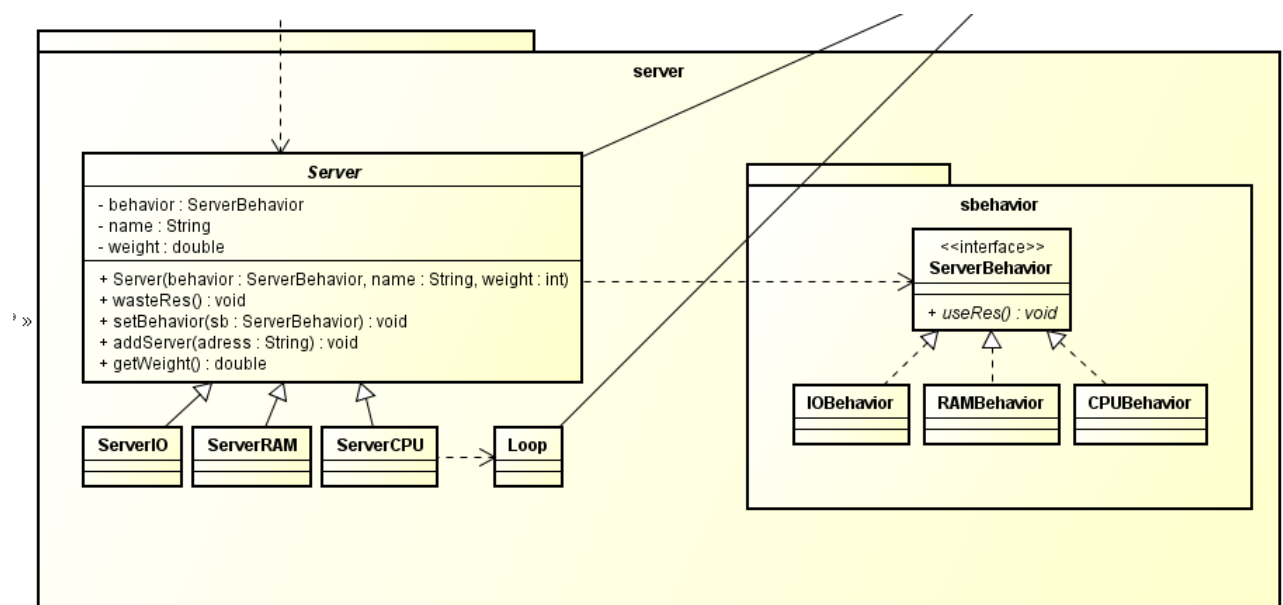
Wir werden RMI benutzen, um die Clients mit den Servern und LoadBalancer kommunizieren zu lassen. Die Informationen, wie den Namen zum Aufrufen der Methoden vom LoadBalancer, bekommen die Clients und Server bei ihrer Erstellung. Der LoadBalancer besitzt 2 Methoden die mittels RMI aufrufbar sind. Eine 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.



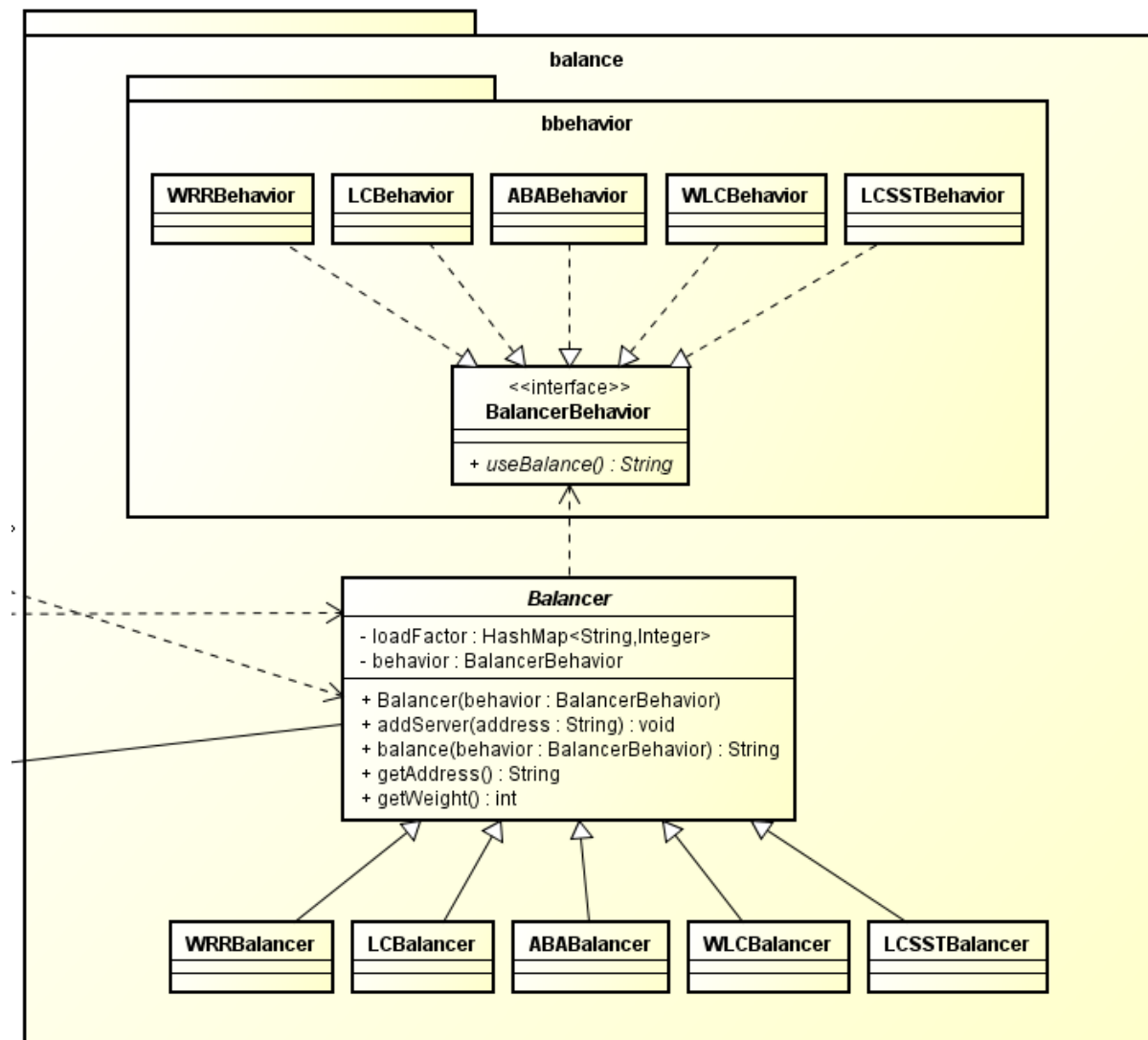
Strategy Pattern

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.

Server - StrategyPattern



LoadBalancer - StrategyPattern



Technology Description

Load Balancing

LoadBalancing ist eine Methode, mithilfe der man die Client verursachte Last auf mehreren Servern besser verteilen kann. Das funktioniert indem der Client bei dem Versuch sich mit einem Server zu verbinden, zuerst zu dem LoadBalancer weitergeleitet wird. Der LoadBalancer besitzt verschiedene Algorithmen und Informationen welche er benutzt, um einen geeigneten Server auszuwählen.

Weighted Round-Robin

Weighted Round-Robin ist ein Verteilungs-Algorithmus bei dem der LoadBalancer die Clients gleichmäßig an die Server verteilt, wobei der LoadBalancer für die einzelnen Server eine Gewichtung haben kann. Diese Gewichtung kann Aussagen das bevor der zweite Server einen Client bekommt die ersten 2 Clients zugewiesen bekommt. Das macht dann Sinn wenn gewisse Server mehr Ressourcen zur Verfügung haben, als andere und somit eine größere Last verarbeiten können.

Least Connection

Die Methode Least Connection reagiert nicht auf Gewichtung, sondern schaut auf welchen Servern am wenigsten Clients verbunden sind. Der Server auf dem die wenigsten Connections aktiv sind, bekommen dadurch die Clients. Das funktioniert besonders dann gut, wenn alle Server gleich gut sind und man davon ausgeht das jeder Client eine gleich große Last erzeugt.

Least Connected Slow Start-Time

Die Least Connected Slow Start-Time Methode funktioniert genauso wie Least-Connection. Allerdings gibt es eine extra Regel welche den Servern die gerade erst gestartet haben eine geringere Anzahl an Clients bekommt. Diese Methode wird besonders bei dem testen eines Servers benutzt, um bei möglichen Fehler eine kleinere Reaktion zu erhalten.

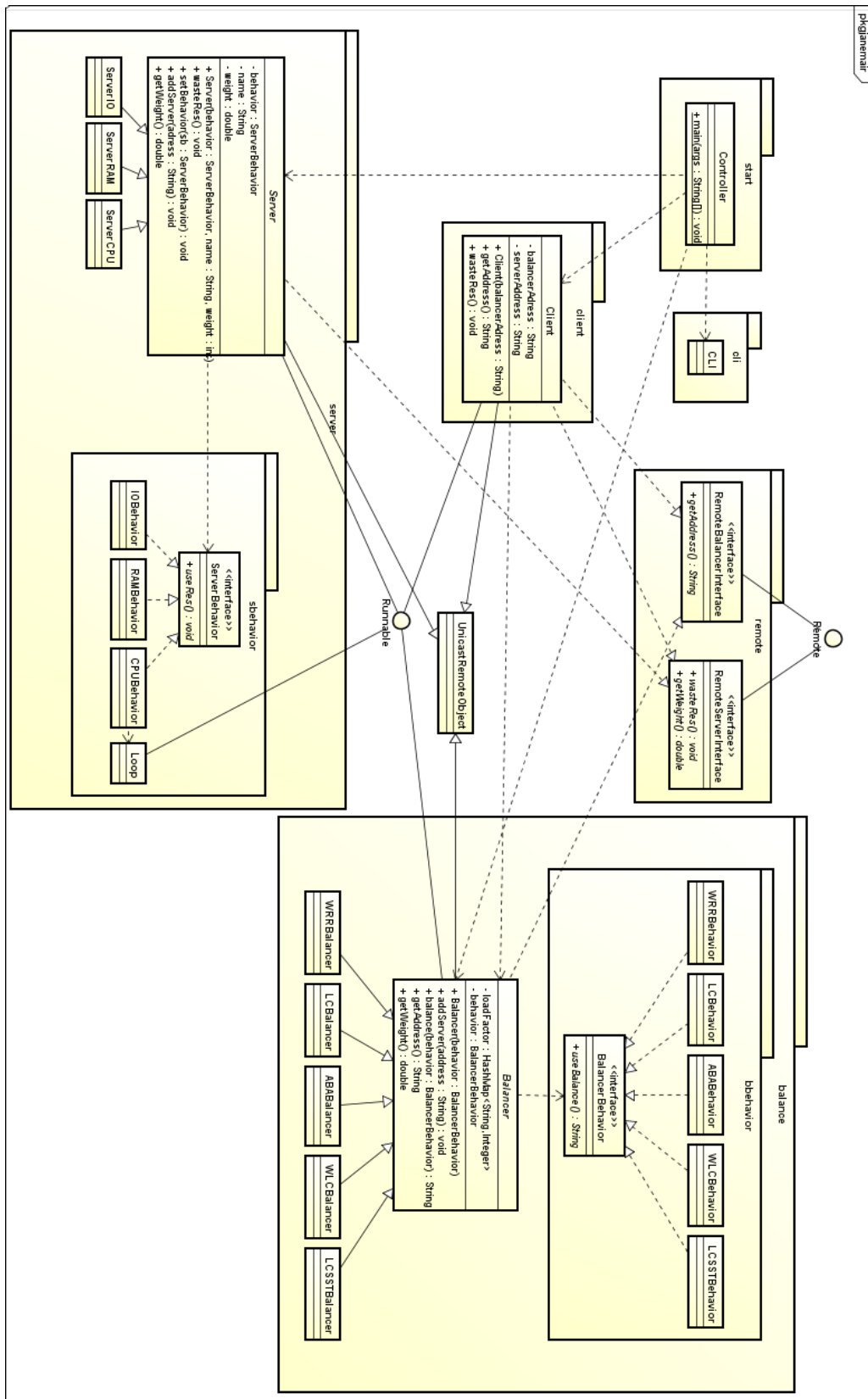
Weighted Least Connection

Weighted beschreibt die Bevorzugung mancher Server vor anderen, wie im Weighted Round-Robin bereits beschrieben wurde. Der Algorithmus ist eine Mischung aus Least Connection und Weighted. Das heißt der LoadBalancer weist den Server mit dem wenigsten Clients und deren Priorität hoch ist den nächsten Client zu.

Agent Based Adaptive Balancing / Server Probes

Dieser Algorithmus benutzt einen Agent der die Server befragt, wie deren Zustand in der letzten Zeit ist. Durch diese Information kann der LoadBalancer, je nach Auslastung, die Clients auf die Server zuteilen dessen Auslastung am geringsten ist.

UML-Diagram



Test Cases

Noch keine vorhanden ...

Apportionment of work with effort estimation

Christian Janeczek

Working Hours

DATE	PHASE	TASK	ESTIMATION	ACTUAL	COMMENT
2015-01-08	Design	Designüberlegung -UML	1:30:00	2:30:00	
2015-01-14	Coding	Vorbereiten des Codes	08:00:00	2:30:00	
2015-01-15	Coding	Least Connection	00:00:00	2:00:00	
SUM			9:30:00	7:00:00	

Wolfgang Mair

Working Hours

DATE	PHASE	TASK	ESTIMATION	ACTUAL	COMMENT
2015-01-08	Design	Designüberlegung - UML	2:00:00	2:30:00	
2015-01-14	Coding	Schreiben der Server	07:00:00	2:30:00	
2015-01-15	Coding	Anfänge RMI	00:00:00	3:00:00	
SUM			9:00:00	8:00:00	

Summe

Working Hours

NAME	ESTIMATION	ACTUAL
Janeczek	9:30:00	7:00:00
Mair	9:00:00	8:00:00
SUM	18:30:00	15:00:00

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>