JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll
Load Balancing

Systemtechnik Labor
5BHITT 2015/16, Gruppe X

Andreas Ernhofer & Jakub Kopec

Version 1.0

Note:

Betreuer: Th.Micheler

Begonnen am 12. Februar 2016

Beendet am 3. März 2016

Inhaltsverzeichnis

1 Einführung 3
1.1 Ziele 3
1.2 Voraussetzungen 3
1.3 Aufgabenstellung 3
2 Ergebnisse 5
2.1 Designüberlegungen 5
2.2 Umsetzung 6
2.3 Algorithmen 7
2.4 Session Persistence 8
2.5 Loadbalancing Software 9
2.6 Testszenarien 13
3 Probleme 15
4 Zeitaufzeichnung 16
5 Quellen 17

Abbildungsverzeichnis

Abbildung 1 Designüberlegung - Erster Entwurf 5
Abbildung 2 UML - Tatsächliche Umsetzung 6
Abbildung 3 nginx - kein eindeutiger Zugriff 10
Abbildung 4 nginx - eindeutiger Zugriff 10
Abbildung 5 nginx Test - Webserver 1 12
Abbildung 6 nginx Test - Webserver 2 12
Abbildung 7 Ausgabe des Load Balancers nach Registrierung der Server 13
Abbildung 8 Verteilung vom Load Balancer 14
Abbildung 9 Ausgabe auf Server2 14
Abbildung 10 Ausgabe auf Server3 14

1 Einführung

Dieses Protokoll beschreibt die Durchführung der DEZSYS-10-Load Balancing-Übung.

1.1 Ziele

Ziel dieser Übung ist:

- Die Auseinandersetzung mit Load Balancing Methoden
- Die Implementierung dieser
- Die Verwendung bestehender Load Balancer

1.2 Voraussetzungen

Die Voraussetzungen für diese Übung beschränken sich auf die Kenntnis verschiedener Kommunikationsmethoden wie beispielsweise Java RMI, Sockets, etc.

1.3 Aufgabenstellung

Es soll ein Load Balancer mit mindestens 2 unterschiedlichen Load-Balancing Methoden (jeweils 6 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 Distribution
- Least Connection
- Response Time
- Server Probes
- Um die Komplexität zu steigern, soll zusätzlich eine "Session Persistence" (3 Punkte) implementiert werden.
-

Vertiefend soll eine Open-Source Applikation aus folgender Liste ausgewählt und installiert werden. (3 Punkte)

https://www.inlab.de/articles/free-and-open-source-load-balancing-software-and-projects.html

Auslastung

Es sollen die einzelnen Server-Instanzen in folgenden Punkten belastet (Memory, CPU Cycles) werden können. 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, Build-Management-Tool (ant oder maven), Gepackt als ausführbares JAR

Viel Erfolg!

Quellen

[1]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

2 Ergebnisse

2.1 Designüberlegungen

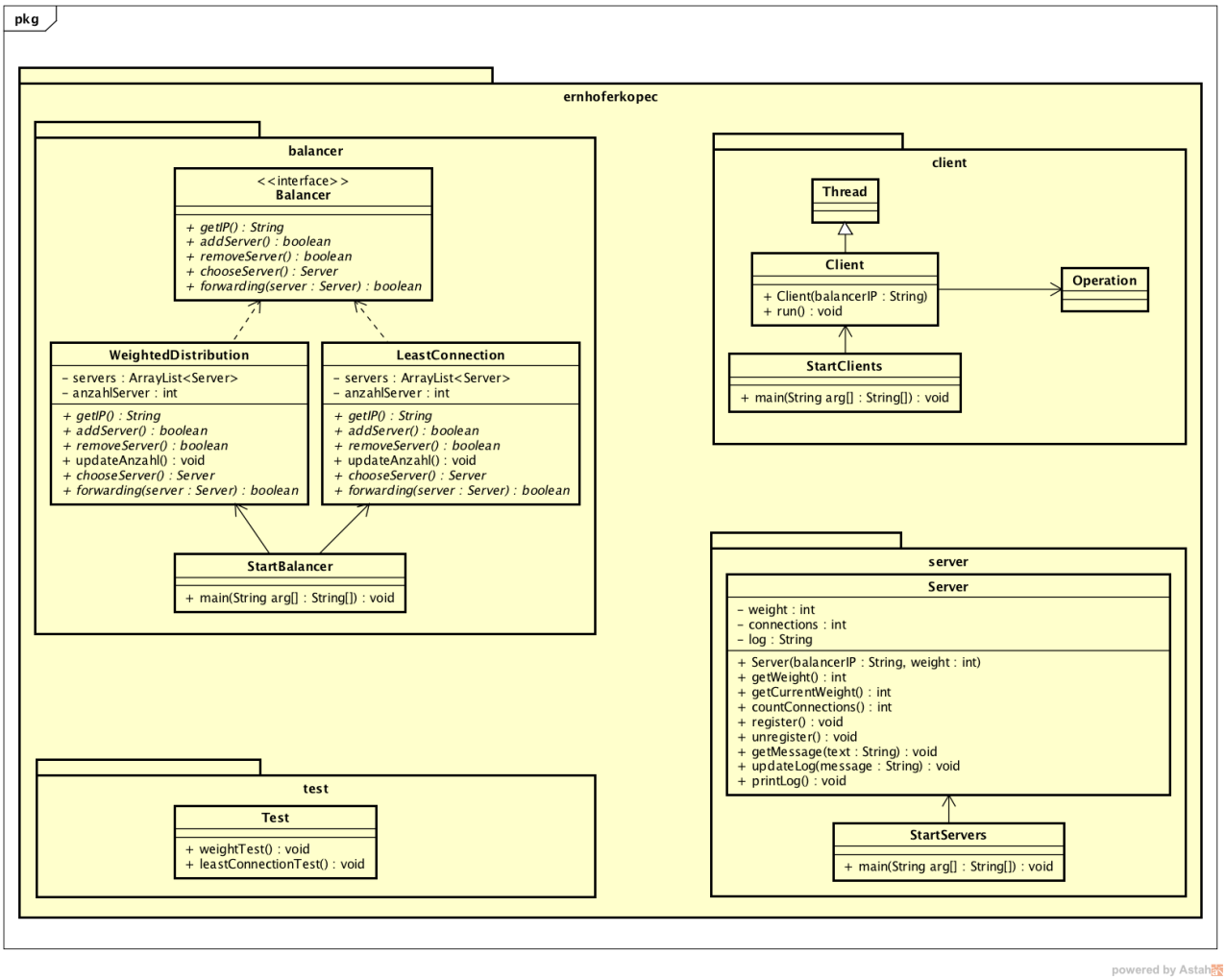


Abbildung 1 Designüberlegung - Erster Entwurf

2.2 Umsetzung

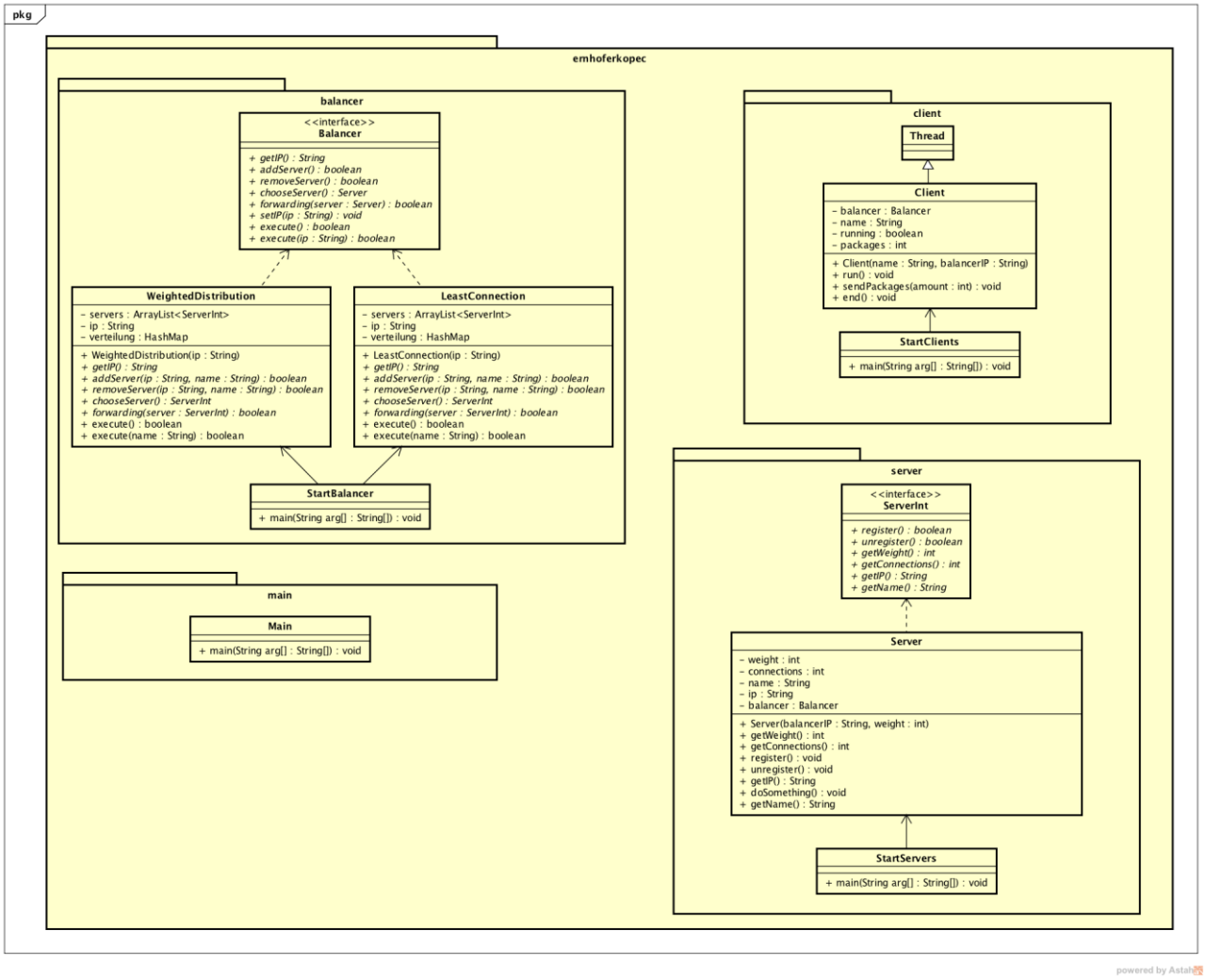


Abbildung 2 UML - Tatsächliche Umsetzung

2.3 Algorithmen

Für diese Übung wurden die Algorithmen Weighted Distribution und Least Connection gewählt.

Weighted Distribution

Dieser Algorithmus funktioniert, indem man die Server gewichtet. Es werden den Webservern beispielsweise Zahlen von 1-10 vergeben. Nun werden die Server in eine Liste am Loadbalancer gespeichert und nach ihrer Gewichtung sortiert. Eine solche Liste könnte beispielsweise wie folgt aussehen:

#	Webserver	Gewichtung
1	Server1	5
2	Server2	4
3	Server3	1

Wenn nun Anfragen an den Loadbalancer kommen, werden diese immer an den ersten Server in der Liste weitergeleitet. Sobald eine Anfrage an einen Server weitergeleitet wird und er diese abarbeitet, sinkt seine Gewichtung um die Anzahl der Anfragen für die Zeit die der Server benötigt um diese Anfragen abzuarbeiten. Sobald der Server damit fertig ist, steigt seine Gewichtung wieder.

Wenn beispielsweise der erste Server eine Anfrage bekommt, sinkt seine Gewichtung auf vier. Nun bleibt er aber immer noch an der ersten Stelle in der Liste, da er nicht weniger wiegt als der nächste Server in der Liste. Noch bevor er mit dem Abarbeiten der ersten Anfrage fertig ist, kommt beim Loadbalancer noch eine Anfrage herein. Diese wird (wie immer) an den ersten Server in der Liste weitergeleitet. Sobald das geschehen ist, sinkt die Gewichtung des ersten Servers auf drei. Da nun die Gewichtung des ersten Servers kleiner ist als die Gewichtung des zweiten Servers in der Liste, nimmt nun der zweite Server den ersten Platz in der Liste ein. Weitere Anfragen werden also an den zweiten Server weitergeleitet. Sobald der erste Server aber mit seinen Anfragen fertig ist, steigt natürlich seine Gewichtung und somit der Platz auf der Liste.

Least Connection

Dieser Algorithmus funktioniert, indem die Anfragen, an den Loadbalancer an den Webserver mit den wenigsten aktiven Verbindungen, weitergeleitet werden.

Anfangs haben alle Server keine aktiven Verbindungen, da die Anfragen der Clients erst an den Loadbalancer gestellt werden können, wenn der Loadbalancer und mindestens ein Server bereits rennen. Somit sind am Anfang, anders als beim Weight Distribution Algorithmus, alle Server

gleichwertig.

Sobald Clients Anfragen an den Loadbalancer stellen, wird überprüft welcher Server die wenigsten aktiven Verbindungen hat. Dies kann durch einen Zähler, welcher die Anzahl der aktiven Verbindungen zählt, auf jedem Server sichergestellt werden. Anfangs wird die erste Anfrage im Endeffekt, an den ersten Server weitergeleitet, da jeder Server gleichwertig ist und es somit keinen Server gibt der eine geringere Anzahl an Verbindungen aufweisen kann als garkeine.

Wird eine Anfrage an einen Server weitergeleitet, erhöht sich sein Zähler der aktiven Verbindungen für die Abarbeitungsdauer der Anfrage. Sobald die Anfrage abgearbeitet ist, sinkt der Zähler wieder um die Anzahl der Abgearbeiteten Anfragen.

Angenommen ein Server ist zum Eintreffzeitpunkt neuer Anfragen an den Loadbalancer bereits mit mehreren Anfragen beschäftigt, überprüft der Loadbalancer wieder alle Server, indem er die Zähler der einzelnen Server abfragt und die Anfrage immer an den Server mit dem kleinsten Zähler weiterleitet.

2.4 Session Persistence

Das Programm soll so geändert werden, dass ein Client, sobald er einmal zu einem Server weitergeleitet wurde, immer zu diesem geleitet wird. Dies soll noch vor der loadbalancing-Methode umgesetzt werden. Das Programm läuft also wie folgt ab. Ein Client sendet eine Anfrage an den Loadbalancer. Anstatt den Benutzer, wie bisher an den, laut loadbalancing-Methode besten Server weiterzuleiten, wird überprüft ob der Benutzer schon mal eine Anfrage gestellt hat. Ist dies nicht der Fall, so wird wie bisher der beste Server ermittelt und die Anfrage des Cients an diesen weitergeleitet. Anschließend wird die Adresse des Clients, sowie die Adresse an welchen der Client weitergeleitet wurde gespeichert. Somit ist es möglich zukünftige Anfragen des Benutzers wieder an den gleichen Server zu senden und somit eine Session Beständigkeit zu gewährleisten.

2.5 Loadbalancing Software

Als Open-Source Applikation für das Loadbalancing wurde nginx gewählt.

Voraussetzung

Um die Software Ordnungsgemäß testen zu können wurden zu diesem Zweck drei virtuelle Maschinen erstellt:

- Loadbalancing Server
- Webserver 1
- Webserver 2

Der Hintergedanke dabei, war es den Loadbalancer anzusprechen, welcher daraufhin die Anfrage auf einen der beiden Webserver weiterleitet.

Installation

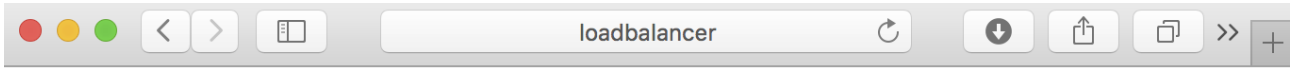
Da alle virtuellen Maschinen auf einer Debian Linux-Distribution basieren, kann man nginx mit folgendem Befehl installieren:

```
$ sudo apt-get install nginx
```

Dieser Befehl muss logischerweise auf allen virtuellen Maschinen ausgeführt werden.

Zugriff

Nachdem nginx installiert wurde, läuft der Service auch bereits und es kann darauf zugegriffen werden. Wenn man nun im Browser die IP-Adressen der einzelnen virtuellen Maschinen eingibt, erhält man eine nginx Willkommens-Maske die derzeit bei jeder Maschine gleich aussehen sollte.



Welcome nginx on Debian!

If you see this page, the nginx web server is successfully installed and working on Debian. Further configuration is required.

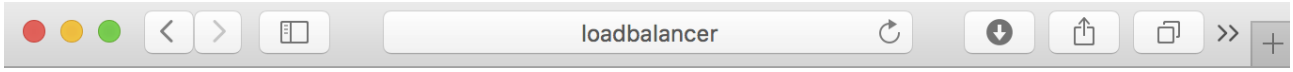
For online documentation and support please refer to nginx.org

Please use the `reportbug` tool to report bugs in the nginx package with Debian. However, check existing bug reports before reporting a new bug.

Thank you for using debian and nginx.

Abbildung 3 nginx - kein eindeutiger Zugriff

Um anhand der Willkommensmaske unterscheiden zu können, welche virtuelle Maschine gerade angesprochen wird, sollte man diese eindeutig verändern. Das index.html – File, welches die Maske darstellt, befindet sich in /var/www/html/. So könnte beispielsweise eine eindeutige Änderung aussehen:



Welcome nginx on Loadbalancer!

If you see this page, the nginx web server is successfully installed and working on Debian. Further configuration is required.

For online documentation and support please refer to nginx.org

Please use the `reportbug` tool to report bugs in the nginx package with Debian. However, check existing bug reports before reporting a new bug.

Thank you for using debian and nginx.

Abbildung 4 nginx - eindeutiger Zugriff

Konfiguration

Sobald jede virtuelle Maschine eine eindeutige Antwort liefert, kann der nächste Schritt vorgenommen werden und der Loadbalancer konfiguriert werden.

Dazu wird in der default-site-config, welche sich unter `/etc/nginx/sites-available/default` befindet folgender Eintrag getätigt:

```
upstream backend {
     server 192.168.99.12;
     server 192.168.99.13;
}
server{
     location / {
          proxy_pass http://backend;
     }
}
```

Sobald der Eintrag getätigt wurde, muss nginx mittels service nginx restart neu gestartet werden. Nun funktioniert das Loadbalancing automatisch mit dem Round Robin Algorithmus. [1]

Test

Um nun zu Überprüfen ob das auch wirklich funktioniert, kann man die IP-Adresse des Loadbalancers im Browser aufrufen. Nun sollte ersichtlich sein, dass die Anfragen an die beiden Webserver weitergeleitet werden, indem die Willkommensmaske des jeweiligen Webservers aufscheint. Durch mehrmaliges Aktualisieren im Browser, sollte die Maske nun immer abwechselnd von einem Webserver zum anderen wechseln.

Abbildung 5 nginx Test - Webserver 1

Abbildung 6 nginx Test - Webserver 2

In diesen Abbildungen ist nun zu sehen, dass die Anfragen des Browsers an den Loadbalancer an die beiden Webserver weitergeleitet werden.

Nützliche Befehle

```
$ sudo service nginx restart

$ sudo service nginx status

$ nginx -t
```

2.6 Testszenarien

Da sich die Gruppe aus zwei Personen zusammenstellt, wurde auch das Programm mit zwei verschiedenen Rechnern getestet. Die beiden Rechner haben folgende Adressen:

Rechner1: 10.0.105.234
Rechner2: 10.0.104.130

Auf Rechner1 wurde der Load Balancer gestartet welcher durch least Connection die Anfragen etwaiger Clients an Server weiterleitet. Zusätzlich dazu wird, wie oben beschrieben eine Session Persistence verwendet. Nachdem der Load Balancer gestartet ist, wurden auf Rechner1 zwei Server, mit den Namen Server1 und Server2 gestartet. Anschließend wurde auch Rechner2 ins Geschehen eingebracht. Auf diesem wurden ebenfalls zwei Server, mit den Namen Server3 und Server4, gestartet. Somit wurden auf beiden Rechner jeweils zwei Server zum Laufen gebracht. Auf dem Load Balancer war nun folgende Ausgabe zu sehen:

```
StartBalancer [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (03.03.2016 16:45:40)
Balancer started
Server server1 von 10.0.105.234 hinzugefuegt
Server server2 von 10.0.105.234 hinzugefuegt
Server server3 von 10.0.104.130 hinzugefuegt
Server server4 von 10.0.104.130 hinzugefuegt
```

Abbildung 7 Ausgabe des Load Balancers nach Registrierung der Server

Durch diese Ausgabe ist nun zu sehen dass erfolgreich die Beiden Server von Rechner1, so wie die Beiden von Rechner2, beim Load Balancer registriert wurden. Der nächste Schritt bestand darin, dass Clients gestartet werden mussten. Dies wurde vom Rechner2 aus durchgeführt. Dabei wurden 50 Clients erzeugt, welche jeweils 10 Pakete ausführen wollen. Diese Anfragen werden an den Load Balancer gesendet, welcher anschließend entscheidet, auf welchem Server die Anfragen ausgeführt werden sollen. Zur besseren Übersicht gibt der Load Balancer für jedes einzelne Paket an, an welcher Client auf welchen Server weitergeleitet wird. Folgendes ist ein Ausschnitt der ausgegebenen Meldung:

```
StartBalancer [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (03.03.2016 16:45:40)
Weiterleiten von Client8 an server1 auf 10.0.105.234
Weiterleiten von Client9 an server1 auf 10.0.105.234
Weiterleiten von Client8 an server1 auf 10.0.105.234
Weiterleiten von Client9 an server1 auf 10.0.105.234
Weiterleiten von Client10 an server2 auf 10.0.105.234
Weiterleiten von Client11 an server3 auf 10.0.104.130
Weiterleiten von Client9 an server1 auf 10.0.105.234
Weiterleiten von Client10 an server2 auf 10.0.105.234
Weiterleiten von Client12 an server3 auf 10.0.104.130
Weiterleiten von Client14 an server3 auf 10.0.104.130
Weiterleiten von Client13 an server3 auf 10.0.104.130
Weiterleiten von Client10 an server2 auf 10.0.105.234
Weiterleiten von Client10 an server2 auf 10.0.105.234
Weiterleiten von Client15 an server3 auf 10.0.104.130
```

Abbildung 8 Verteilung vom Load Balancer

Zusätzlich zu der Ausgabe am Load Balancer gibt auch jeder Server einzeln aus, dass er für die Ausführung der Methode gewählt wurde. Hier zwei Beispiele von Server2(laufend auf Rechner1) und Server3(laufend auf Rechner2) dargestellt:

Server2:

```
StartServer [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (03.03.2016 16:45:43)
Ausfuehren der Methode auf server2
Ausfuehren der Methode auf server2
Ausfuehren der Methode auf server2
Ausfuehren der Methode auf server2
Ausfuehren der Methode auf server2
Ausfuehren der Methode auf server2
```

Abbildung 9 Ausgabe auf Server2

Server3:

```
StartServer [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Home/bin/java (03. März 2016, 16:53:45)
Ausfuehren der Methode auf server3
Ausfuehren der Methode auf server3
Ausfuehren der Methode auf server3
Ausfuehren der Methode auf server3
Ausfuehren der Methode auf server3
Ausfuehren der Methode auf server3
Ausfuehren der Methode auf server3
Ausfuehren der Methode auf server3
Ausfuehren der Methode auf server3
Ausfuehren der Methode auf server3
```

Abbildung 10 Ausgabe auf Server3

3 Probleme

RMI

Im Verlaufe der Aufgabe traten einige Fehler bei der Kommunikation mittels RMI auf. Lokal auf einem Rechner konnte das Programm schnell ausgeführt werden und funktionierte einwandfrei. Sobald es jedoch auf mehreren Rechnern, auf einem der Balancer und auf einem anderer ein Server, ausgeführt werden sollte, trat ein Fehler nach dem anderen auf. Das erste Problem konnte dadurch behoben werden, dass in der java.policy die benötigten Berechtigungen gegeben wurden. Ein weiteres Problem war es, dass von einem Rechner immer nur auf localhost gebinded werden konnte. Da dies jedoch nicht genügend war für den fehlerfreien Ablauf des Programms, musste zu einer etwas komplizierteren Methode gegriffen werden, welche dieses Problem umgeht.

Nginx

In Bezug auf nginx sind die Fehler bei der Konfiguration aufgetreten. Es war nicht ersichtlich in welches Konfigurationsfile die Änderungen eingetragen hätten werden sollen. Anfangs wurde versucht in folgendes Konfigurationsfile zu schreiben:

```
/etc/nginx/nginx.conf
```

Dabei hätte man die Änderungen in folgendem File ändern sollen:
`/etc/nginx/sites-available/default`

Der Fehler konnte durch folgende Befehle identifiziert werden:

```
$ sudo service nginx status
```

```
$ nginx –t
```

Der erste Befehl zeigt den Status an und der zweite testet die Konfiguration.

4 Zeitaufzeichnung

Tätigkeit	Dauer
Designüberlegung	2h
Weighted Distribution Einbindung	1h
Least Connection Einbindung	1h
Kommunikation mittels RMI	2h
Session Percistence	30min
Loadbalancing Software	2h
Problembearbeitung und Fehlerbehebung	6h
Protokollierung	2h
Gesamt	**16,5h**

5 Quellen

[1] http://nginx.org/en/docs/http/load_balancing.html

[2] https://www.nginx.com/resources/glossary/session-persistence/

[3] https://github.com/jkopec/DEZSYS-10.git