JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

Load Balancing

Systemtechnik Labor
5BHITT 2015/16

Stefan Erceg & Martin Kritzl

Version 1.0

Note:

Betreuer: Prof. Borko & Micheler

Begonnen am 12. Februar 2016

Beendet am 12. März 2016

Inhaltsverzeichnis

1 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" (2 Punkte) implementiert werden.

Vertiefend soll eine Open-Source Applikation aus folgender Liste ausgewählt und installiert werden. (2 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 aufkommende 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

Bewertung: 16 Punkte
- 2 Load Balancing Methoden (jeweils 6 Punkte)
- Session Persistenz (2 Punkte)
- Einsatz Load Balancing Software (2 Punkte)

Viel Erfolg!

Quellen

[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] Available at: http://www.tik.ee.ethz.ch/education/lectures/VS/SS02/Praktikum/loesung2.zip

2 Ergebnisse

2.1 Aufwandabschätzung

Teilaufgabe	benötigte Gesamtzeit
UML-Diagramm erstellen	2 h
Implementierung des Programms inkl. JavaDoc	20 h
Testen des Programms	4 h
Protokoll schreiben	6 h
Gesamt	**32 h**

2.2 anschließende Endzeitaufteilung

2.2.1 Erceg

Arbeit	Datum	Zeit
UML	12.02.2016	1 h
UML fertiggestellt	19.02.2016	1 h 30 min
Implementierung	23.02.2016	2 h 30 min
Implementierung	26.02.2016	1 h 30 min
Implementierung	03.03.2016	3 h 30 min
Implementierung	04.03.2016	1 h 30 min
Implementierung	10.03.2016	3 h 30 min
Protokoll	12.03.2016	2 h
Testen	12.03.2016	2 h
Gesamt	*12.03.2016*	**19 h**

2.2.2 Kritzl

Arbeit	Datum	Zeit
UML	12.02.2016	1 h
UML fertiggestellt	19.02.2016	1 h
Implementierung	19.02.2016	30 min
Implementierung	23.02.2016	2 h 30 min
Implementierung	26.02.2016	1 h 30 min
Implementierung	03.03.2016	3 h 30 min
Implementierung	04.03.2016	1 h 30 min
Implementierung	10.03.2016	6 h 30 min
Protokoll	12.03.2016	2 h
Testen	12.03.2016	2 h
Gesamt	*12.03.2016*	**22 h**

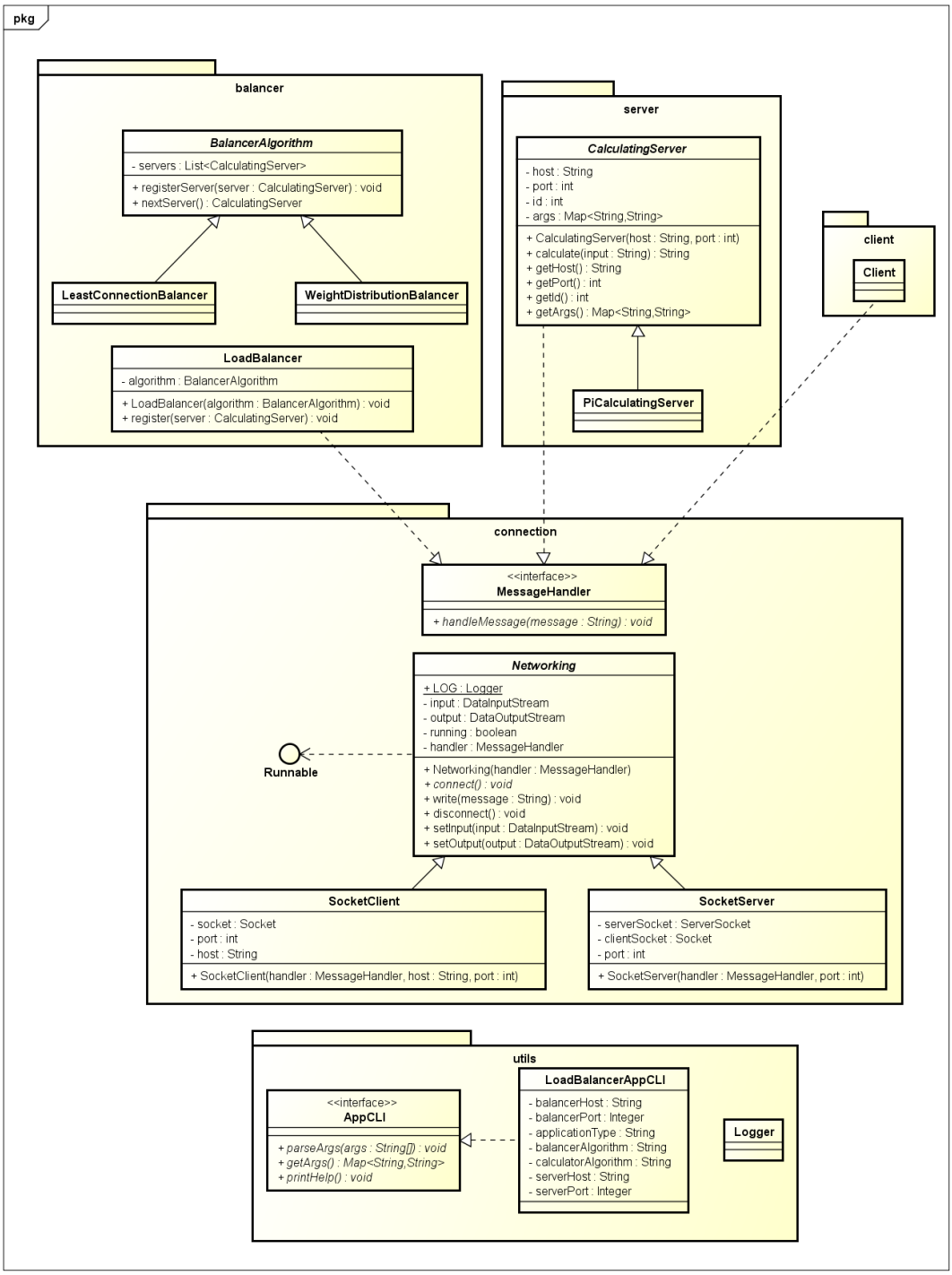
2.2.3 Gesamtsumme

Insgesamt haben wir für diese Übung **41 Stunden** benötigt. Geschätzt wurden 32 Stunden, daher lagen wir mit unserer Einschätzung ziemlich daneben. Dies liegt vor allem daran, weil wir für die Implementierung des Programms 10 Stunden länger benötigten als geplant.

2.3 Designüberlegung

2.3.1 Abbildung

Das UML-Diagramm wurde mit dem Programm „Astah" erstellt.



2.3.2 Überlegungen zur Struktur

Das Programm wurde in 5 Packages unterteilt:

- **„connection"-Package**

  Hierbei wird eine Socket-Verbindung zur Kommunikation zwischen dem Load Balancer, dem Server und dem Client implementiert. Die „Networking"-Klasse besitzt sowohl die Methoden zum Connecten und Disconnecten einer Verbindung, als auch zum Schreiben einer Message. Der „SocketServer" eröffnet einen Port, der „ClientServer" verbindet sich hingegen zu dem „SocketServer" mittels Angabe des Ports und der Host-Adresse.

  Der „SocketServer" stellt den Load Balancer dar und der „SocketClient" den Server und den Client.

- **„server"-Package**

  In diesem Package befinden sich 2 Klassen. Einerseits die Klasse „CalculatingServer", welche eine abstrakte Klasse darstellt und die Methode zum Berechnen bereitstellt, und anderseits die Klasse „PiCalculatingServer", welche die Klasse „CalculatingServer" erweitert und die Methode zum Berechnen der Nachkommastellen von Pi ausführt.

- **„balancer"-Package**

  Die 2 konkreten Algorithmen Least-Connection und Weighted-Distribution, welche wir ausgewählt haben, erweitern hierbei die abstrakte Klasse "BalancerAlgorithm". Im Parameter der „LoadBalancer"-Klasse kann der Algorithmus, welcher verwendet werden soll, angegeben werden.

- **„client"-Package**

  Die "Client"-Klasse erstellt hier lediglich ein Attribut vom "SocketClient" und implementiert das Interface "MessageHandler".

- **„utils"-Package**

  Dieses Package enthält das Interface und die konkrete Implementierung einer Apache-Common-CLI. Dies wird implementiert, um vor dem Start des Programms beispielsweise angeben zu können, welcher Applikationstyp (Load Balancer, Client, Server) und welcher Load Balancer-Algorithmus verwendet werden soll.

2.4 Umsetzung

2.4.1 Balancer-Algorithmen

Für die Umsetzung der Algorithmen wurde eine Abstrakte Klasse verwendet die als Vorlage für weitere Implementierungen dient. In dieser ist die Liste der verfügbaren Server abgelegt, genauso wie die Methode zur Registrierung der Server implementiert.

```
public void registerServer(CalculatingServer server, String type) {
   if (this.servers.get(type)==null)
      this.servers.put(type, new ArrayList<CalculatingServer>());

   this.servers.get(type).add(server);
}
```

Zur möglichst schnellen Ermittlung des am besten geeigneten Servers, wurde ein Comparator geschrieben, welcher die Server nach bestimmten Attributen, wie die Anzahl der offenen Connections oder der Gewichtung sortiert. Somit muss nur noch der oberste Server herausgenommen werden und damit eine Berechnung durchgeführt werden.

```
Collections.sort(servers, new Comparator<CalculatingServer>() {
    @Override
    public int compare(CalculatingServer o1, CalculatingServer o2) {
        if (asc)
            return Integer.parseInt(o1.getArgs().get(attribute))-
(Integer.parseInt(o2.getArgs().get(attribute)));
        else
            return Integer.parseInt(o2.getArgs().get(attribute))-
(Integer.parseInt(o1.getArgs().get(attribute)));
    }
});
```

Sowohl der Balancer für Least-Connections als auch für Weighted-Distribution erbt von BalancerAlgorithm. Somit können diese flexibel verwendet werden. Der Algorithmus für Least-Connections und Weighted-Distribution ist sehr ähnlich.

Bei ersterem wird ein Attribut bei einer neuen Berechnung erhöht und bei erhalten des Ergebnisses wieder dekrementiert. Somit sind immer die offenen Verbindungen im Attribut einsehbar. Aufgrund der sortierten Liste, kann immer der oberste Server genommen werden.

```
List<CalculatingServer> servers =
ServerSorter.sortByValues(this.getServers().get(type), "connections", true);
CalculatingServer nextServer = servers.get(0);
nextServer.getArgs().put("connections",
Integer.parseInt(nextServer.getArgs().get("connections"))+1+"");
return nextServer;
```

Bei Weighted-Distribution ist ein Attribut vorhanden welche die Gewichtung des Servers darstellt. Ein zweites Attribut wird zu Beginn mit dem Wert der Gewichtung befüllt und stellt damit die verbleibenden Berechnungen dar. Wählt der Balancer einen Server aus, so wird dieses Attribut vermindert. Wegen der sortierten Liste, kann immer der oberste Server herausgenommen werden. Sollte dieser Server jedoch eine verbleibende Anzahl von Berechnungen gleich 0 haben, so hat kein anderer Server verbleibende Berechnungen. Somit werden alle verbleibenden Berechnungen wieder mit dem Wert der Gewichtung befüllt und der Vorgang beginnt erneut.

```
List<CalculatingServer> servers =
ServerSorter.sortByValues(this.getServers().get(type), "remainingCalculations",
false);
if (servers==null) return null;
CalculatingServer nextServer = servers.get(0);
if (nextServer.getArgs().get("remainingCalculations").equals("0"))
    this.resetWeight(type);

nextServer.getArgs().put("remainingCalculations",
Integer.parseInt(nextServer.getArgs().get("remainingCalculations"))-1+"");
return nextServer;
```

2.5 Testszenarien

2.5.1 Weighted-Distribution

Als Testszenario wurde ein Balancer gestartet, der den Weighted-Distribution Algorithmus verwendet. Zu diesem Balancer haben sich zwei Server verbunden, die eine unterschiedliche Gewichtung haben (Abbildung 1: Load-Balancer (zwei Server angemeldet) Abbildung 1 Zeile 4-7). Danach haben sich mehrere Clients (siehe Abbildung 4) (7 Stück) mit dem Balancer verbunden und nach der Berechnung von Pi gefragt. Zu sehen ist, dass Server 1 (siehe Abbildung 2) wesentlich mehr Anfragen erhalten hat, als der andere Server (siehe Abbildung 3).

```
2016-03-12 14:42:49,304 [main          ] INFO  LoadBalancer                - Started Load-Balancer
2016-03-12 14:42:49,305 [main          ] INFO  LoadBalancer                - Server Port: 33000
2016-03-12 14:42:49,305 [main          ] INFO  LoadBalancer                - Client Port: 44000
2016-03-12 14:42:53,311 [Thread-2      ] INFO  ServerMessageHandler        - Got message from Server: /register pi 3
2016-03-12 14:42:53,324 [Thread-2      ] INFO  ServerMessageHandler        - Server registered: 127.0.0.1:51658
2016-03-12 14:42:55,729 [Thread-3      ] INFO  ServerMessageHandler        - Got message from Server: /register pi 1
2016-03-12 14:42:55,729 [Thread-3      ] INFO  ServerMessageHandler        - Server registered: 127.0.0.1:51662
2016-03-12 14:43:08,290 [Thread-4      ] INFO  ClientMessageHandler        - Got message from client: pi 1000
2016-03-12 14:43:08,298 [Thread-4      ] INFO  ClientMessageHandler        - Sending request to 127.0.0.1:51667
2016-03-12 14:43:08,325 [Thread-2      ] INFO  ServerMessageHandler        - Got message from Server: 0 3.14159265358979
2016-03-12 14:43:08,325 [Thread-2      ] INFO  ServerMessageHandler        - Sending response to client: 127.0.0.1:51667
2016-03-12 14:43:08,346 [Thread-4      ] INFO  ClientNetworking            - Connection closed: 127.0.0.1:51667
2016-03-12 14:43:10,879 [Thread-5      ] INFO  ClientMessageHandler        - Got message from client: pi 1000
2016-03-12 14:43:10,879 [Thread-5      ] INFO  ClientMessageHandler        - Sending request to 127.0.0.1:51671
2016-03-12 14:43:10,888 [Thread-2      ] INFO  ServerMessageHandler        - Got message from Server: 1 3.14159265358979
2016-03-12 14:43:10,890 [Thread-2      ] INFO  ServerMessageHandler        - Sending response to client: 127.0.0.1:51671
2016-03-12 14:43:10,892 [Thread-5      ] INFO  ClientNetworking            - Connection closed: 127.0.0.1:51671
```

Abbildung 1: Load-Balancer (zwei Server angemeldet)

```
2016-03-12 14:42:53,309 [main          ] INFO  PiCalculatingServer         - Register Server on Balancer
2016-03-12 14:43:08,297 [Thread-0      ] INFO  CalculatingServer           - Got message from balancer: 0 1000
2016-03-12 14:43:08,324 [Thread-0      ] INFO  CalculatingServer           - Calculated result for client 0: 1000
2016-03-12 14:43:10,879 [Thread-0      ] INFO  CalculatingServer           - Got message from balancer: 1 1000
2016-03-12 14:43:10,888 [Thread-0      ] INFO  CalculatingServer           - Calculated result for client 1: 1000
2016-03-12 14:43:13,149 [Thread-0      ] INFO  CalculatingServer           - Got message from balancer: 2 1000
2016-03-12 14:43:13,155 [Thread-0      ] INFO  CalculatingServer           - Calculated result for client 2: 1000
2016-03-12 14:43:27,655 [Thread-0      ] INFO  CalculatingServer           - Got message from balancer: 5 1000
2016-03-12 14:43:27,664 [Thread-0      ] INFO  CalculatingServer           - Calculated result for client 5: 1000
2016-03-12 14:43:30,077 [Thread-0      ] INFO  CalculatingServer           - Got message from balancer: 6 1000
2016-03-12 14:43:30,081 [Thread-0      ] INFO  CalculatingServer           - Calculated result for client 6: 1000
2016-03-12 14:43:38,370 [Thread-0      ] INFO  CalculatingServer           - Got message from balancer: 7 1000
2016-03-12 14:43:38,375 [Thread-0      ] INFO  CalculatingServer           - Calculated result for client 7: 1000
```

Abbildung 2: Server 1 (Weight: 3)

```
2016-03-12 14:42:55,727 [main          ] INFO  PiCalculatingServer         - Register Server on Balancer
2016-03-12 14:43:18,835 [Thread-0      ] INFO  CalculatingServer           - Got message from balancer: 3 1000
2016-03-12 14:43:18,855 [Thread-0      ] INFO  CalculatingServer           - Calculated result for client 3: 1000
2016-03-12 14:43:22,966 [Thread-0      ] INFO  CalculatingServer           - Got message from balancer: 4 1000
2016-03-12 14:43:22,975 [Thread-0      ] INFO  CalculatingServer           - Calculated result for client 4: 1000
```

Abbildung 3: Server 2 (Weight: 1)

```
2016-03-12 14:43:38,368 [main          ] INFO  Client                    - Asking balancer for pi with 1000 digits
2016-03-12 14:43:38,376 [Thread-0      ] INFO  Client                    - Got message from balancer: 3.1415926535
```

Abbildung 4: Client-Aufruf

2.5.2 Least Connections

2.5.2.1 Testfall 1

Bei diesem Testfall wird ein Balancer gestartet, welcher den „Least Connections"-Algorithmus verwendet. Bei diesem Balancer registrieren sich 2 Server, welche Pi berechnen. 2 Clients schicken gleichzeitig eine Anfrage zum Berechnen von 100 000 Nachkommastellen von Pi. Wie man an den Abbildungen 5 bis 9 erkennen kann, arbeiten die beiden Server dies parallel ab und somit werden die Anfragen vom Balancer gerecht auf die Server verteilt.

```
2016-03-12 14:42:13,470 [main          ] INFO  LoadBalancer              - Started Load-Balancer
2016-03-12 14:42:13,472 [main          ] INFO  LoadBalancer              - Server Port: 33000
2016-03-12 14:42:13,473 [main          ] INFO  LoadBalancer              - Client Port: 44000
2016-03-12 14:42:17,547 [Thread-2      ] INFO  ServerMessageHandler      - Got message from Server: /register pi 10
2016-03-12 14:42:17,568 [Thread-2      ] INFO  ServerMessageHandler      - Server registered: 127.0.0.1:50676
2016-03-12 14:42:21,190 [Thread-3      ] INFO  ServerMessageHandler      - Got message from Server: /register pi 10
2016-03-12 14:42:21,190 [Thread-3      ] INFO  ServerMessageHandler      - Server registered: 127.0.0.1:50680
2016-03-12 14:42:29,121 [Thread-4      ] INFO  ClientMessageHandler      - Got message from client: pi 100000
2016-03-12 14:42:29,129 [Thread-4      ] INFO  ClientMessageHandler      - Sending request to 127.0.0.1:50684
2016-03-12 14:42:32,827 [Thread-5      ] INFO  ClientMessageHandler      - Got message from client: pi 100000
2016-03-12 14:42:32,828 [Thread-5      ] INFO  ClientMessageHandler      - Sending request to 127.0.0.1:50688
2016-03-12 14:43:02,487 [Thread-2      ] INFO  ServerMessageHandler      - Got message from Server: 0 3.14159265358979323846264338327950288419716939
2016-03-12 14:43:02,527 [Thread-2      ] INFO  ServerMessageHandler      - Sending response to client: 127.0.0.1:50684
2016-03-12 14:43:02,566 [Thread-4      ] INFO  ClientNetworking          - Connection closed: 127.0.0.1:50684
2016-03-12 14:43:04,882 [Thread-3      ] INFO  ServerMessageHandler      - Got message from Server: 1 3.14159265358979323846264338327950288419716939
2016-03-12 14:43:04,930 [Thread-3      ] INFO  ServerMessageHandler      - Sending response to client: 127.0.0.1:50688
2016-03-12 14:43:04,965 [Thread-5      ] INFO  ClientNetworking          - Connection closed: 127.0.0.1:50688
```

Abbildung 5: Load-Balancer (zwei Server angemeldet)

```
2016-03-12 14:42:17,545 [main          ] INFO  PiCalculatingServer       - Register Server on Balancer
2016-03-12 14:42:29,128 [Thread-0      ] INFO  CalculatingServer         - Got message from balancer: 0 100000
2016-03-12 14:43:02,479 [Thread-0      ] INFO  CalculatingServer         - Calculated result for client 0: 100000
```

Abbildung 6: Server 1

```
2016-03-12 14:42:21,187 [main          ] INFO  PiCalculatingServer       - Register Server on Balancer
2016-03-12 14:42:32,828 [Thread-0      ] INFO  CalculatingServer         - Got message from balancer: 1 100000
2016-03-12 14:43:04,880 [Thread-0      ] INFO  CalculatingServer         - Calculated result for client 1: 100000
```

Abbildung 7: Server 2

```
2016-03-12 14:42:29,115 [main          ] INFO  Client                    - Asking balancer for pi with 100000 digits
2016-03-12 14:43:02,535 [Thread-0      ] INFO  Client                    - Got message from balancer: 3.14159265358979323846264338327950288419716939937510
```

Abbildung 8: Client 1

```
2016-03-12 14:42:32,825 [main          ] INFO  Client                    - Asking balancer for pi with 100000 digits
2016-03-12 14:43:04,940 [Thread-0      ] INFO  Client                    - Got message from balancer: 3.14159265358979323846264338327950288419716939937510
```

Abbildung 9: Client 2

2.5.2.2 Testfall 2

Beim 2. Testfall des "Least Connections"-Algorithmus existieren ebenfalls ein Balancer und 2 Server. Hier wollen jedoch 5 Clients 100 000 Nachkommastellen von Pi berechnet bekommen. Wie man an den Abbildungen 10 bis 17 erkennen kann, arbeitet ein Server 3 Client-Anfragen ab und der zweite 2 Client-Anfragen. Die Abarbeitung erfolgt parallel und somit wurde eine gerechte Aufteilung der Client-Anfragen durchgeführt.

```
2016-03-12 14:48:52,588 [main          ] INFO  LoadBalancer               - Started Load-Balancer
2016-03-12 14:48:52,590 [main          ] INFO  LoadBalancer               - Server Port: 33000
2016-03-12 14:48:52,590 [main          ] INFO  LoadBalancer               - Client Port: 44000
2016-03-12 14:48:59,906 [Thread-2      ] INFO  ServerMessageHandler       - Got message from Server: /register pi 10
2016-03-12 14:48:59,929 [Thread-2      ] INFO  ServerMessageHandler       - Server registered: 127.0.0.1:50712
2016-03-12 14:49:04,401 [Thread-3      ] INFO  ServerMessageHandler       - Got message from Server: /register pi 10
2016-03-12 14:49:04,401 [Thread-3      ] INFO  ServerMessageHandler       - Server registered: 127.0.0.1:50716
2016-03-12 14:49:13,337 [Thread-4      ] INFO  ClientMessageHandler       - Got message from client: pi 100000
2016-03-12 14:49:13,342 [Thread-4      ] INFO  ClientMessageHandler       - Sending request to 127.0.0.1:50720
2016-03-12 14:49:16,633 [Thread-5      ] INFO  ClientMessageHandler       - Got message from client: pi 100000
2016-03-12 14:49:16,634 [Thread-5      ] INFO  ClientMessageHandler       - Sending request to 127.0.0.1:50724
2016-03-12 14:49:20,811 [Thread-6      ] INFO  ClientMessageHandler       - Got message from client: pi 100000
2016-03-12 14:49:20,812 [Thread-6      ] INFO  ClientMessageHandler       - Sending request to 127.0.0.1:50729
2016-03-12 14:49:24,700 [Thread-7      ] INFO  ClientMessageHandler       - Got message from client: pi 100000
2016-03-12 14:49:24,703 [Thread-7      ] INFO  ClientMessageHandler       - Sending request to 127.0.0.1:50733
2016-03-12 14:49:29,212 [Thread-8      ] INFO  ClientMessageHandler       - Got message from client: pi 100000
2016-03-12 14:49:29,213 [Thread-8      ] INFO  ClientMessageHandler       - Sending request to 127.0.0.1:50737
2016-03-12 14:49:51,601 [Thread-2      ] INFO  ServerMessageHandler       - Got message from Server: 0 3.14159265358979323846264338327950288419716939937
2016-03-12 14:49:51,675 [Thread-2      ] INFO  ServerMessageHandler       - Sending response to client: 127.0.0.1:50720
2016-03-12 14:49:51,703 [Thread-4      ] INFO  ClientNetworking           - Connection closed: 127.0.0.1:50720
2016-03-12 14:49:53,886 [Thread-3      ] INFO  ServerMessageHandler       - Got message from Server: 1 3.14159265358979323846264338327950288419716939937
2016-03-12 14:49:53,912 [Thread-3      ] INFO  ServerMessageHandler       - Sending response to client: 127.0.0.1:50724
2016-03-12 14:49:53,970 [Thread-5      ] INFO  ClientNetworking           - Connection closed: 127.0.0.1:50724
2016-03-12 14:50:21,981 [Thread-2      ] INFO  ServerMessageHandler       - Got message from Server: 3 3.14159265358979323846264338327950288419716939937
2016-03-12 14:50:22,104 [Thread-2      ] INFO  ServerMessageHandler       - Sending response to client: 127.0.0.1:50733
2016-03-12 14:50:22,130 [Thread-7      ] INFO  ClientNetworking           - Connection closed: 127.0.0.1:50733
2016-03-12 14:50:24,101 [Thread-3      ] INFO  ServerMessageHandler       - Got message from Server: 2 3.14159265358979323846264338327950288419716939937
2016-03-12 14:50:24,110 [Thread-3      ] INFO  ServerMessageHandler       - Sending response to client: 127.0.0.1:50729
2016-03-12 14:50:24,144 [Thread-6      ] INFO  ClientNetworking           - Connection closed: 127.0.0.1:50729
2016-03-12 14:50:49,882 [Thread-2      ] INFO  ServerMessageHandler       - Got message from Server: 4 3.14159265358979323846264338327950288419716939937
2016-03-12 14:50:49,892 [Thread-2      ] INFO  ServerMessageHandler       - Sending response to client: 127.0.0.1:50737
2016-03-12 14:50:49,915 [Thread-8      ] INFO  ClientNetworking           - Connection closed: 127.0.0.1:50737
```

Abbildung 10: Load-Balancer (zwei Server angemeldet)

```
2016-03-12 14:48:59,904 [main          ] INFO  PiCalculatingServer        - Register Server on Balancer
2016-03-12 14:49:13,341 [Thread-0      ] INFO  CalculatingServer          - Got message from balancer: 0 100000
2016-03-12 14:49:51,592 [Thread-0      ] INFO  CalculatingServer          - Calculated result for client 0: 100000
2016-03-12 14:49:51,593 [Thread-0      ] INFO  CalculatingServer          - Got message from balancer: 3 100000
2016-03-12 14:50:21,980 [Thread-0      ] INFO  CalculatingServer          - Calculated result for client 3: 100000
2016-03-12 14:50:21,990 [Thread-0      ] INFO  CalculatingServer          - Got message from balancer: 4 100000
2016-03-12 14:50:49,877 [Thread-0      ] INFO  CalculatingServer          - Calculated result for client 4: 100000
```

Abbildung 11: Server 1

```
2016-03-12 14:49:04,398 [main          ] INFO  PiCalculatingServer        - Register Server on Balancer
2016-03-12 14:49:16,634 [Thread-0      ] INFO  CalculatingServer          - Got message from balancer: 1 100000
2016-03-12 14:49:53,932 [Thread-0      ] INFO  CalculatingServer          - Calculated result for client 1: 100000
2016-03-12 14:49:53,932 [Thread-0      ] INFO  CalculatingServer          - Got message from balancer: 2 100000
2016-03-12 14:50:24,100 [Thread-0      ] INFO  CalculatingServer          - Calculated result for client 2: 100000
```

Abbildung 12: Server 1

```
2016-03-12 14:49:13,334 [main          ] INFO  Client                     - Asking balancer for pi with 100000 digits
2016-03-12 14:49:51,682 [Thread-0      ] INFO  Client                     - Got message from balancer: 3.14159265358979323846264338327950288419716939937510582097494
```

Abbildung 13: Client 1

```
2016-03-12 14:49:16,630 [main          ] INFO  Client                    - Asking balancer for pi with 100000 digits
2016-03-12 14:49:53,926 [Thread-0      ] INFO  Client                    - Got message from balancer: 3.14159265358979323846264338327950288419716939937510582097494
```

Abbildung 14: Client 2

```
2016-03-12 14:49:20,808 [main          ] INFO  Client                    - Asking balancer for pi with 100000 digits
2016-03-12 14:50:24,118 [Thread-0      ] INFO  Client                    - Got message from balancer: 3.14159265358979323846264338327950288419716939937510582097494
```

Abbildung 15: Client 3

```
2016-03-12 14:49:24,675 [main          ] INFO  Client                    - Asking balancer for pi with 100000 digits
2016-03-12 14:50:22,111 [Thread-0      ] INFO  Client                    - Got message from balancer: 3.14159265358979323846264338327950288419716939937510582097494
```

Abbildung 16: Client 4

```
2016-03-12 14:49:29,208 [main          ] INFO  Client                    - Asking balancer for pi with 100000 digits
2016-03-12 14:50:49,897 [Thread-0      ] INFO  Client                    - Got message from balancer: 3.14159265358979323846264338327950288419716939937510582097494
```

Abbildung 17: Client 5

3 Github-Link

https://github.com/serceg-tgm/DezSys10_Load_Balancing