Ein Performance-Benchmark für Java-Web-Frameworks

Diplomarbeit

im Studiengang Informatik

am Fachbereich Mathematik und Informatik

der Freien Universität Berlin

Christophe Sokol

sokol@inf.fu-berlin.de

3984302

Eigenständigkeitserklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, 29. April 2013

Inhaltsverzeichnis

1 Einleitung....4
2 Umfeld....4
2.1 Web-Anwendungen....5
2.2 Web-Frameworks....5
2.3 Java-Web-Frameworks....13
2.4 Apache Tapestry....14
2.5 Spring MVC....14
2.6 JSR: Java-Web-Standards....15
2.7 Funktionalitäten von Frameworks....16
2.8 Benchmarking....17
2.9 Bestehende Benchmarks....19
3 Durchführung....26
3.1 Erwartete Probleme....26
3.2 Iteratives Vorgehen....27
3.3 Infrastruktur....27
4 Ergebnisse....28
4.1 Automatisierung des Prozesses....29
4.2 Teilbenchmarks....30
4.3 Interaktion und Transparenz....35
5 Ausblick....41
6 Schluss....42
7 Glossar....42
8 Verweise....45

1 Einleitung

Mit dieser Arbeit stelle ich einen Performance-Benchmark für Java-Web-Frameworks vor. Das prinzipielle Vorgehen bei einem Benchmark erläutere ich in Kapitel 2.8. Die Schwierigkeit beim Vergleichen von Web-Frameworks liegt in ihrem unterschiedlichen Umfang und verschiedenen Einsatzgebieten. Um eine differenziertere Betrachtung zu ermöglichen, habe ich in den Kapiteln 2.1 und 2.2 Bestandteile von Web-Anwendungen und Frameworks erarbeitet, die eine Technologie- und Architektur-neutrale Betrachtung von Web-Frameworks ermöglichen. Diese sind eine Grundlage der in Kapitel 4.2 vorgestellten Teilbenchmarks.

Der von mir gewählte Ansatz resultiert ebenfalls und in erheblichem Maße aus einer Betrachtung von bestehenden Benchmarks, die ich in Kapitel 2.9 beschreibe. Neben Stärken und Schwächen beim Vorgehen, beim Umfang oder der technischen Realisierung haben sie fast alle eines gemein: Ihre Ergebnisse sind letztendlich nur zum Zeitpunkt der Durchführung gültig. Für eine neue Version eines Frameworks können sich die Ergebnisse stark verändert haben.

In einem weiteren Teil dieser Arbeit werde ich daher eine Plattform zur Automatisierung der Benchmarks vorstellen. Diese berücksichtigt, dass die Geschwindigkeit einer Web-Anwendung von der Infrastruktur deren Konfiguration abhängig ist. Sie versucht durch die Nutzung von verteilen Versionsverwaltungssystemen und Web-Oberflächen die Schwelle für Beiträge von externen Entwicklern möglichst gering zu halten. Sie soll von den Entwicklern der Frameworks als nützliches Werkzeug für automatisiere Performance-Messungen als Teil der analytischen Qualitätssicherung wahrgenommen werden.

Der Fokus dieser Arbeit lag zunächst im Adressieren der Probleme, die bei der Durchführung eines solchen Benchmarks auftreten. Die Implementierungen für die Frameworks Spring MVC und Tapestry sowie die durchgeführten Messungen haben, vor allem aus Zeitmangel, die Rolle eines Machbarkeitsnachweises und dienen der Veranschaulichung.

Wie in vielen Domänen gibt es im Web und bei der Java-Plattform unzählige Abkürzungen und Fachbegriffe. Diese habe ich im Glossar erläutert.

2 Umfeld

Web-Frameworks werden genutzt, um Web-Anwendungen zu entwickeln. Betrachtet man die Geschwindigkeit von Web-Frameworks, dann geschieht dies, weil diese die Geschwindigkeit einer damit entwickelten Anwendung beeinflusst. Dieses Kapitel geht der Frage nach, ob es eine typische

Anwendung gibt, mit Hilfe derer sich Frameworks betrachten lassen. Da die Natur von Web-Anwendungen auch mit der Verfügbarkeit von Web-Technologien im Laufe der Zeit verändert hat, gehe ich darauf ebenfalls ein.

2.1 Web-Anwendungen

Das Web hat sich in seit seiner Erfindung durch Tim Berners-Lee im Jahr 1989 laufend verändert. Die statischen Inhalte der Anfangsjahre wurden von interaktiven Anwendungen abgelöst. Derzeit gewinnt die Verknüpfung insbesondere sozialer Daten von verschiedenen Plattformen an Bedeutung. Zudem entwickelt es sich von einem Informationsmedium zu einer Anwendungsplattform, die klassische Desktop-Anwendungen ablöst.

In diesem Zusammenhang spielt aber auch die Bandbreite an Technologien eine große Rolle. Die sogenannten Browserkriege, die Programmiersprachen Java und PHP, die Web-Server Apache und IIS sind nur die bekanntesten Beispiele für unterschiedliche Technologien, die im Web aufeinander prallen. Es scheint keinen Ort im Web zu geben, an dem nicht Firmen und lose organisierte Gemeinschaften, Standards und Konventionen, kommerzielle und gemeinnützige Ziele miteinander konkurrieren, kooperieren oder nebeneinander existieren.

Dazu kommt, dass es im Web unterschiedliche Personengruppen mit eigenen Anforderungen an Anwendungen gibt, insbesondere hinsichtlich einer privaten oder geschäftlichen Nutzung. Es fällt sehr schwer, eine typische Web-Anwendung zu beschreiben, denkbar wäre höchstens eine Gruppierung. Die oben angesprochene Veränderung sowie die technologische und inhaltliche Vielfalt sind möglicherweise eine Ursache für das Fehlen von einheitlichen, Plattform übergreifenden Anwendungs-Architekturen für das Web.[1]

Es wäre aber falsch, von einer völligen Abwesenheit von Anwendungs-Architekturen zu sprechen. Es existieren Entwurfsmuster, Technologie-Kombinationen und Anforderungen, die häufig gemeinsam eingesetzt werden.

2.2 Web-Frameworks

Der Begriff „Web-Framework“ bezeichnet Bibliotheken und Programme mit ihren dazugehörigen Software-Architekturen, die zum Erstellen von Web-Anwendungen dienen. Dies kann sowohl ein

1 Eichinger, “Web Application Architectures.” lässt sich gleich doppelt als Beispiel heranziehen. Er beschreibt eindrucksvoll die Vielfalt an Technologien und Anforderungen, die es im Kontext von Architekturen gibt. Gleichzeitig ist er ein gutes Beispiel für eine technologische Blase, indem er Technologien und Entwicklungen außerhalb der J2EE-Welt ignoriert.

„Framework für Web-Anwendungen“ als auch ein „Web-Framework für Anwendungen“ sein. Das heißt Komponenten, die nicht ausschließlich mit dem Web zu tun haben, werden je nach Umfeld sprachlich ein- oder ausgeschlossen. Aus technischer oder praktischer Sicht hat dies keine Bedeutung, muss aber beachtet werden, wenn man den Umfang eines Benchmarks festlegen möchte.

Web-Frameworks unterscheiden sich teilweise erheblich im Funktionsumfang. Dies hat damit zu tun, dass auf einigen Plattformen Funktionalitäten bereits durch andere Anwendungs-Frameworks bereitgestellt werden. Zu den Kernaufgaben eines Web-Frameworks gehört das Routing, d. h. die Zuordnung von URLs zu Methodenaufrufen sowie das Darstellen von Daten mittels Templates. Solche Frameworks werden auch Microframeworks genannt, als Urvater gilt das in Ruby geschriebene Sinatra.[2]

Umfangreichere Frameworks, dann Full-Stack-Framework genannt, bieten häufig benötigte Funktionalitäten wie die Verarbeitung von Formulareingaben, Datei-Uploads, Validierung von Nutzerdaten, Datenabstraktion und -persistenz, das Erstellen oder Versenden von E-Mails, Methoden zur Authentifizierung von Nutzern, Autorisierung von Zugriffen oder die Internationalisierung der Anwendung, also die Bereitstellung der Inhalte in verschiedenen Sprachen, Konvertierung von Zeichensätzen.[3]

Ein Großteil der Web-Frameworks verweist direkt und indirekt auf das Architekturmuster MVC, indem sie darauf ausgelegt sind oder Komponenten bereitstellen mit denen es umgesetzt werden kann.[4] In Kombination mit einem relationalen Datenbanksystem ist dies eine populäre Anwendungsarchitektur, es haben sich aber auch weitere Bestandteile wie NoSQL, MapReduce oder verteilte Warteschlangen etabliert, insbesondere bei umfangreicheren Anwendungen.[5]

Routing

Das Routing legt fest, welcher Teil des Anwendungscodes bei einem Zugriff aufgerufen wird. Dies geschieht vorwiegend anhand des Request-URI, bei vielen Frameworks gibt es eine feste Zuordnung zwischen URI und Komponenten der Anwendung. Darüber hinaus werden Request-Header genutzt, insbesondere der Accept-Header, um Daten in einem anderen Format anzufragen. Die Request-Methode wird genutzt, um Web-services im REST-Stil zu realisieren.

2 Weier O’Phinney, “On Microframeworks.”
3 Donald, “Spring MVC 3 Showcase.”
4 Eichinger, “Web Application Architectures.”
5 Perera, “Srinath’s Blog.”

View / Templates

Zu einer weiteren Kernfunktion von Frameworks gehört das Anzeigen von Daten, nachdem die Anwendung diese abgerufen und verarbeitet hat. Dies bedeutet, mit Templates HTML-Code zu generieren, dort Verweise oder Formulare hinzuzufügen und sie in ein Layout einzubetten. Umfangreiche Frameworks bieten hier die Möglichkeit, Inhalte aus verschiedenen Teilen der Anwendung zusammenzuführen, d. h. nebeneinander darzustellen oder umfangreiche Layouts durch Verschachtelung von Templates zu modularisieren. Frameworks enthalten häufig Vorlagen für gängige Ansichten oder können diese erzeugen, dies können Ansichten zum Auflisten, Anzeigen, Bearbeiten oder Löschen einer Menge von Datenobjekten sein, die Integration einer Bibliothek oder eines externen Dienstes.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<table>
  <c:forEach var="person" items="${persons}">
    <tr>
      <c:url var="editUrl" value="/person.html">
        <c:param name="id" value="${person.id}" />
      </c:url>
      <td>${person.firstName}</td>
      <td>${person.lastName}</td>
    </tr>
  </c:forEach>
</table>
```

Text 1: JSP mit Tags aus der JSTL um eine Tabelle mit Personen anzuzeigen (Aus Winterfeldt, "Spring Web Flow Subflow Webapp: JSP Example.*")*

Ein gängiger Anwendungsfall ist die Internationalisierung. Dafür bieten Frameworks Funktionen zur Verwaltung von Übersetzungen und Konvertierung von unterschiedlichen Darstellungsvarianten von numerischen Werten. Außerdem kann eine Aufbereitung der Daten stattfinden, beispielsweise die Erhöhung der Lesbarkeit von Datumsangaben, indem sie durch geeignete Terme wie „vor mehr als einem Jahr" ausgedrückt werden.

I18n / L10n

Unter Lokalisierung und Internationalisierung versteht man die Anpassung einer Anwendung an Nutzer in verschiedenen Ländern, Sprach- und Kulturräumen. Dazu gehört das Übersetzen der Anwendung in unterschiedliche Sprachen, die korrekte Darstellung von numerischen Werten, der Umgang mit verschiedenen Zeichensätzen, die Anpassung der Oberfläche aufgrund von unterschiedlichen kulturellen Präferenzen und das Anzeigen geographisch passender Inhalte.

Der Standort und die Sprache des Nutzers werden entweder manuell vom Nutzer eingegeben, oder

über die IP-Adresse[6], den Accept-Language-Header und die Geolocation-API[7] des Browsers ermittelt. Letzteres spielt vor allem bei mobilen Clients eine Rolle, da diese eine eingebaute Positionsbestimmung besitzen.

Scaffolding

Scaffolding bezeichnet ist eine Form von Code-Generierung, die bei MVC-Frameworks beispielsweise genutzt wird, um aus der Spezifikation der Anwendungsobjekte ein Anwendungsgerüst zu erzeugen.

Model / Persistenz

Die ersten Web-Anwendungen nutzten keine Abstraktionsmöglichkeiten zur Datenverwaltung. Sie lasen und schrieben ihre Daten mit SQL direkt in Datenbanken oder nutzten strukturierte Textdateien. Diese Vorgehensweise führt bei wachsender Anwendungsgröße zu schwer wartbarem Code, erschwert eine Portierung auf ein anderes Datenbank-System und ermöglicht viele sicherheitskritische Fehler. Für größere Anwendungen werden daher überwiegend OR-Mapper eingesetzt, d. h. Bibliotheken, die die Elemente der Datenbank als Objekte in der Anwendung repräsentieren. Viele Web-Frameworks beinhalten ein eigenes ORM oder sind auf die Integration von etablierten, externen ORM-Frameworks ausgelegt. Eines der bekanntesten ORM-Frameworks für Java ist Hibernate, für Ruby gibt es Active Record, ein Teil des Ruby-on-Rails-Frameworks, für PHP gibt es beispielsweise Propel oder Doctrine. Mit der zunehmenden Verlagerung der Anwendungslogik in den Browser des Clients sind auch in JavaScript geschriebene OR-Mapper entstanden, z. B. backbone.js. Deren Integration in klassische Web-Frameworks ist allerdings noch nicht besonders weit fortgeschritten.

Authentifizierung / Autorisierung

Authentifizierung ist der Vorgang, die Identität eines Nutzers festzustellen. Die anschließende Überprüfung, ob der Nutzer auf die angeforderte Ressource zugreifen darf, wird als Autorisierung bezeichnet. Beides kann auf unterschiedlichen Schichten einer Web-Anwendung realisiert sein. Nutzer (oder Nutzergruppen) können anhand ihrer IP-Adressen identifiziert werden, das HTTP-Protokoll selbst unterstützt Authentifizierung, die Nutzer oder Berechtigungen können innerhalb der Anwendung verwaltet werden, aber auch außerhalb, beispielsweise in Form eines LDAP-Servers oder OpenID-Dienstes. Die tatsächlich genutzte Methode ergibt sich häufig aus dem Kontext, in dem die Anwendung eingesetzt wird.

6 http://dev.maxmind.com/geoip/geolite
7 http://www.w3.org/TR/geolocation-API/

Caching

Das Zwischenspeichern von Daten und Inhalten ist in einer Web-Anwendung an vielen Stellen sinnvoll, um eine optimale Geschwindigkeit zu gewährleisten und schonend mit Ressourcen umzugehen. Auf dem Weg von einer Datenquelle in den Browser eines Nutzers findet Caching beispielsweise an folgenden Stellen statt: Der OR-Mapper verwendet Objekte, die er aus den Ergebnismengen einer Datenbank erzeugt hat, mehrmals. Mit Hilfe von verteilen Datenbanken wie memcached, kann dies sogar auf mehreren Anwendungsservern übergreifend geschehen. Bei einer Seite, die aus mehreren Bestandteilen besteht, werde die Teile einer Seite, die selten aktualisiert werden oder besonders aufwändig zu erzeugen sind, zwischengespeichert und bei weiteren Zugriffen wiederverwendet. Seiten, die sich selten ändern oder besonders vielen Zugriffen ausgesetzt sind, können auch im Ganzen zwischengespeichert werden. Dies kann auch über einen externen Dienst, z. B. einen CDN geschehen. Unter Verwendung der Cache- und If-Modified-Since-Header kann eine Seite auch im Browser-Cache des Nutzer gespeichert werden, so dass sie bis zur nächsten Änderung nicht mehr neu übertragen werden muss. Mit modernen Web-Technologien wie dem HTML5-Offline-Storage und dem Cache-Manifest lassen sich sogar Anwendungen erstellen, die sich ohne Internetverbindung nutzen lassen.

APIs

Daten über maschinenlesbare Schnittstellen anzubieten oder zu konsumieren ist heute Bestandteil vieler moderner Web-Anwendungen und ein zentraler Aspekt des sogenannten Web 2.0. Unabhängig davon haben sich Web-Browser in den letzten Jahren stark weiterentwickelt. Ursprünglich waren sie Programme, die HTML-Seiten anzeigten, mittlerweile sind sie mit optimierten JavaScript-Laufzeitumgebungen und HTML5-APIs zu eigenen Anwendungsplattformen geworden. Beispiele hierfür sind Twitter, die ihre Oberfläche zeitweilig vollständig clientseitig aufbauen ließen, oder das Firefox OS von Mozilla, ein Betriebssystem für Mobiltelefone, welches ausschließlich auf HTML5-Anwendungen setzt.[8] In kleineren Anwendungen ist der Zugriff auf APIs per AJAX zum Anzeigen von Suchvorschlägen, zur clientseitigen Validierung von Formulareingaben oder zum Nachladen von Inhalten üblich. JavaScript-Bibliotheken wie JQuery oder prototype.js bieten hierfür die notwendige Unterstützung.

Zur Implementierung von Web-APIs konkurrieren zwei Gruppen von Technologien. Das XML-basierende SOAP ist durch eine Reihe von Standards definiert und ist nicht auf HTTP als darunter-

8 http://engineering.twitter.com/2012/05/improving-performance-on-twittercom.html
https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Platform/Architecture

liegendes Protokoll beschränkt.[9] REST wird dagegen von seinem Erfinder, Roy Fielding, einem der Autoren des HTTP-Standards, als Architekturstil bezeichnet und ist eng mit dem HTTP verknüpft.[10] Anwendungen, die diesem Stil folgen, werden als RESTful bezeichnet und nutzen häufig JSON als Format zum Übertragen der Daten. APIs im REST-Stil haben sich bei öffentlich zugänglichen Web-Diensten durchgesetzt.[11]

Input / Eingaben

Formulare, selbst wenn sie nicht immer als solche zu erkennen sind, sind die zentralen Elemente zum Abfragen von Nutzerdaten im Web. Die HTML4-Spezifikation sieht vor, dass ein Formular aus einem Form-Tag und einem oder mehreren Eingabefeldern besteht. Wenn der Nutzer auf den „Absenden“-Button klickt, werden die Daten vom Browser an die im Formular hinterlegte Adresse geschickt. Jede darüber hinausgehende Funktionalität, wie eine clientseitige Validierung der Eingaben, eine Modifikation der Eingabefelder in Abhängigkeit bereits getätigter Eingaben, eine Vorschau von Medieninhalten, insbesondere Fotos oder eine Fortschrittsanzeige werden mit JavaScript oder zusätzlichen Technologien wie Flash oder Java umgesetzt. Letzteres hat dazu geführt, dass mit der HTML5 File API[12] die Möglichkeiten der Anwendung, im Browser des Nutzers auf Dateien zuzugreifen, stark erweitert wurde.

Ist ein Formular zu groß und soll auf mehrere Bildschirmseiten aufgeteilt werden, lässt sich dies technisch auf unterschiedliche Arten realisieren. Das Formular kann auf mehrere Seiten verteilt werden, die Anwendung muss sich dann um das Zusammensetzen kümmern. Alternativ kann das Formular in Bereiche unterteilt werden, die nacheinander mit geeignetem JavaScript-Code im Browser des Nutzers ein- und ausgeblendet werden.

Gerade bei großen Webanwendungen wird eine Datei, nachdem sie zum Server übertragen und dort ggf. weiterverarbeitet wurde, auf einen Datei-Server oder an ein externes CDN weitergegeben. Für Ruby-Frameworks gibt es beispielsweise die Bibliothek CarrierWave, die mit Hilfe der Bibliotheken MiniMagick und Fog das Bearbeiten von Bildern und Übertragen von Dateien an Cloud-Dienste integriert.[13]

---

9 http://soapclient.com/standards.html
10 Fielding, “Architectural Styles and the Design of Network-based Software Architectures.”
11 Mason, “How REST Replaced SOAP on the Web: What It Means to You.”
12 http://www.w3.org/TR/FileAPI/
13 https://github.com/jnicklas/carrierwave
https://github.com/fog/fog
https://github.com/minimagick/minimagick

Validierung

Die Validierung von Nutzereingaben findet häufig sowohl im Browser des Nutzers, als auch auf dem Server statt. Die clientseitige Validierung dient dazu, dem Nutzer ein möglichst schnelles Feedback zu seinen Eingaben zu geben. Sie sollte eine serverseitige Validierung auf keinen Fall ersetzen, da nicht auszuschließen ist, dass die Daten vor dem Abschicken modifiziert werden. Eine Validierung umfasst sowohl sicherheitsrelevante als auch anwendungsspezifische Kriterien.

Validierungen können an unterschiedlichen Stellen innerhalb der Anwendung stattfinden. Wenn die Anwendung die Eingaben intern auf Objekte abbildet, bietet es sich an, dort auch die Validierung vorzunehmen, um die Daten konsistent zu halten. Web-Frameworks bieten dafür passende Methoden an.[14] Für Java-basierende Anwendungen gibt es mit dem JSR 303 einen Framework-übergreifenden Standard für Validierungen.[15] Hat eine Validierung das Ziel, das Ausnutzen von Sicherheitslücken innerhalb der Anwendungen oder anderer Komponenten auszuschließen, kann sie außerhalb der Anwendung durchgeführt werden, d. h. eine Anfrage wird ausgewertet, bevor sie an die Anwendung weitergereicht wird.[16]

Statische Dateien

Web-Anwendungen umfassen neben dem serverseitigen Anwendungscode noch weitere Bestandteile wie Bilder, JavaScript- und CSS-Dateien. Mit der bereits erwähnten zunehmenden Verlagerung von Anwendungslogik in den Browser des Nutzers steigen Umfang und Komplexität dieser Bestandteile, beispielsweise durch in JavaScript geschriebene Templates.[17] Die Anzahl und die Größe dieser zusätzlichen Dateien haben einen erheblichen Einfluss auf die Ladegeschwindigkeit einer Web-Anwendung.

Auf Abbildung 1 sieht man die Ladezeit der Startseite des Fachbereichs Informatik mit leerem Cache dargestellt von den Chromium DevTools.[18] Der Browser muss alle referenzierten Elemente vom Webserver abrufen, er baut dafür eine größere Anzahl von gleichzeitigen Verbindungen auf. Würde

---

14 http://static.springsource.org/spring/docs/3.2.x/spring-framework-reference/html/validation.html
http://guides.rubyonrails.org/active_record_validations_callbacks.html

15 http://jcp.org/en/jsr/detail?id=303

16 In diesem Bereich gibt es unzählige Systeme die sich bezüglich ihrer Funktionalität grundlegend unterscheiden, z. B. ModSecurity, das sich als „Web Application Firewall“ bezeichnet und Suhosin, welches sich ausschließlich auf den PHP-Interpreter bezieht.
http://www.modsecurity.org/
http://www.hardened-php.net/suhosin/

17 http://handlebarsjs.com/

18 http://www.mi.fu-berlin.de/inf/
https://developers.google.com/chrome-developer-tools/

er die Elemente so wie ältere Browser mit weniger gleichzeitigen Verbindungen hintereinander abrufen, würde sich die Ladezeit stark vergrößern. Ebenfalls kann man erkennen, dass die URLs aller statischen Dateien mit einer Folge von zehn Ziffern enden, die vom Framework an die Dateinamen angehängt wurden. Diese Technik ermöglicht es, Dateien mit Hilfe des Cache-Control-Headers im Cache des Browsers zu speichern und sie bei einer Änderung durch Hochzählen der Ziffern kurzfristig neu laden zu lassen.

Name	Method	Status	Type	Initiator	Size	Time	Timeline
inf/	GET	200	text/html	Other	9.9 KB	460 ms	
yaml_base.css?2013041107	GET	200	text/css	www.mi.fu-berlin.de:26	3.2 KB	918 ms	
bootstrap_min.css?2013041107	GET	200	text/css	www.mi.fu-berlin.de:27	9.4 KB	919 ms	
nav_vlist.css?2013041107	GET	200	text/css	www.mi.fu-berlin.de:29	1.3 KB	873 ms	
basemod_fu_berlin.css?2013041107	GET	200	text/css	www.mi.fu-berlin.de:30	5.2 KB	932 ms	
nav_hlist.css?2013041107	GET	200	text/css	www.mi.fu-berlin.de:28	2.5 KB	927 ms	
content_fu.css?2013041107	GET	200	text/css	www.mi.fu-berlin.de:31	9.8 KB	929 ms	
design_mi.css?2013041107	GET	200	text/css	www.mi.fu-berlin.de:32	1.4 KB	1.03 s	
yaml_print_base.css?2013041107	GET	200	text/css	www.mi.fu-berlin.de:33	2.0 KB	1.03 s	
print.css?2013041107	GET	200	text/css	www.mi.fu-berlin.de:34	1.8 KB	1.04 s	
jquery.fancybox-1.3.4.css?1332491595	GET	200	text/css	www.mi.fu-berlin.de:44	2.3 KB	1.04 s	
jquery-1.8.0.min.js?1365665958	GET	200	text/x-js	www.mi.fu-berlin.de:51	32.9 KB	1.06 s	
jquery.cycle.all.min.js?1365665958	GET	200	text/x-js	www.mi.fu-berlin.de:51	6.5 KB	1.04 s	
jrails.js?1365665958	GET	200	text/x-js	www.mi.fu-berlin.de:51	1.5 KB	1.05 s	
jquery.fancybox-1.3.4.pack.js?1332491595	GET	200	text/x-js	www.mi.fu-berlin.de:51	5.9 KB	1.13 s	
superfish.js?1365665958	GET	200	text/x-js	www.mi.fu-berlin.de:51	1.9 KB	1.13 s	
equalize.js?1365665958	GET	200	text/x-js	www.mi.fu-berlin.de:51	1.0 KB	1.13 s	
bootstrap-2.3.1.min.js?1365665958	GET	200	text/x-js	www.mi.fu-berlin.de:51	4.6 KB	1.13 s	
hoverIntent.js?1365665958	GET	200	text/x-js	www.mi.fu-berlin.de:51	1.6 KB	1.13 s	
jwplayer.js?1332491595	GET	200	text/x-js	www.mi.fu-berlin.de:51	36.5 KB	1.15 s	
22123921?locale=de&more_link=1&page...	POST	200	text/html	jquery-1.8.0.min.js:2	1.3 KB	92 ms	
17142394?locale=de&more_link=1&page...	POST	200	text/html	jquery-1.8.0.min.js:2	1.5 KB	100 ms	
fu_logo.gif?1365665958	GET	200	image/gif	www.mi.fu-berlin.de:132	6.5 KB	1.05 s	
icon_service_menue.gif?2013041107	GET	200	image/gif	www.mi.fu-berlin.de:185	710 B	1.05 s	
search_button.gif?1365665958	GET	200	image/gif	www.mi.fu-berlin.de:310	941 B	1.05 s	
iiit-b.jpg?1365577890	GET	200	image/jpeg	www.mi.fu-berlin.de:590	44.8 KB	1.08 s	
girlsday.jpg?1360592960	GET	200	image/jpeg	www.mi.fu-berlin.de:651	253 KB	1.32 s	
auszeichnung.png?1349941334	GET	200	image/png	www.mi.fu-berlin.de:720	21.8 KB	1.12 s	
proinformatik.jpg?1355933088	GET	200	image/jpeg	www.mi.fu-berlin.de:609	352 KB	1.17 s	
volkerroth.jpg?1350981913	GET	200	image/jpeg	www.mi.fu-berlin.de:845	54.0 KB	1.20 s	
voisard-ordreChevalier-k.jpg?1360847524	GET	200	image/jpeg	www.mi.fu-berlin.de:741	103 KB	1.16 s	
icon_print.gif?1365665958	GET	200	image/gif	www.mi.fu-berlin.de:316	432 B	1.05 s	
spinner.gif?1365665958	GET	200	image/gif	www.mi.fu-berlin.de:892	2.1 KB	1.14 s	
MatthiasWaehlisch.jpg?1358850218	GET	200	image/jpeg	www.mi.fu-berlin.de:760	98.2 KB	1.13 s	
open_nav_button.gif?1365665958	GET	200	image/gif	www.mi.fu-berlin.de:382	399 B	1.06 s	
link_list_arrow.png?2013041107	GET	200	image/png	www.mi.fu-berlin.de:595	912 B	1.06 s	
icon_rss.gif?1365665958	GET	200	image/gif	www.mi.fu-berlin.de:318	703 B	1.05 s	
close_nav_button.gif?1365665958	GET	200	image/gif	www.mi.fu-berlin.de:380	402 B	1.05 s	
prev_arrow_disabled.png?1365665958	GET	200	image/png	jquery-1.8.0.min.js:2	598 B	1.15 s	
last_arrow_disabled.png?1365665958	GET	200	image/png	jquery-1.8.0.min.js:2	600 B	1.16 s	
next_arrow_disabled.png?1365665958	GET	200	image/png	jquery-1.8.0.min.js:2	602 B	1.15 s	
overview.png?1365665958	GET	200	image/png	jquery-1.8.0.min.js:2	563 B	1.14 s	
first_arrow_disabled.png?1365665958	GET	200	image/png	jquery-1.8.0.min.js:2	597 B	1.14 s	
next_arrow.png?1365665958	GET	200	image/png	jquery-1.8.0.min.js:2	601 B	1.15 s	
last_arrow.png?1365665958	GET	200	image/png	jquery-1.8.0.min.js:2	598 B	1.16 s	

Abbildung 1: Ladezeit der Homepage des FB Informatik (Ausschnitt)

Weitere Techniken zur Beschleunigung der Auslieferung von Webseiten sind die Komprimierung von CSS und JavaScript-Dateien oder das Zusammenfassen von kleinen Dateien.[19]

Ausblick

Unter anderem die Mozilla-Foundation treibt mit dem „Firefox OS“ die Entwicklung des Web zu einer Anwendungsplattform voran:

> *„For Web developers, the most important part to understand is that the entire user interface is a Web app, [...] Any modifications you make to the user interface and any applications you create to run on Firefox OS are Web pages, albeit with enhanced access to the mobile device's hardware and services.“*[20]

19 http://yui.github.io/yuicompressor/
http://guides.rubyonrails.org/asset_pipeline.html

20 https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Introduction

In diesem Zusammenhang sind auch die Bestrebungen des W3C zu betrachten, Kommunikationsprotokolle jenseits von HTTP zu standardisieren. Das WebSocket-Protokoll realisiert bidirektionale Kommunikation zwischen Web-Anwendungen und Web-Servern und wird zur Zeit von rund 60 % der weltweit eingesetzten Browser implementiert. Mit Hilfe von Bibliotheken kann dessen Funktionalität auch in den Browsern genutzt werden, die WebSockets noch nicht nativ implementieren.[21] Die Unterstützung auf der Serverseite ist noch nicht so weit fortgeschritten, das Spring-Framework beispielsweise wird die Java-WebSockets-API erst in der nächsten Version unterstützen, die Ende 2013 erscheinen soll.[22] Viele andere Frameworks bieten derzeit noch keine eigene Unterstützung an.

Weiterhin befindet sich WebRTC in Entwicklung, eine Browser-Schnittstelle für Video- und Audio-Chats. Das Projekt wird von Google und Mozilla unterstützt, die Implementierungen in ihren Browsern Chrome und Firefox beisteuerten. WebRTC soll vom W3C standardisiert werden. Microsoft, der dritte große Browserhersteller und Eigentümer von Skype, bemüht sich dagegen um die Etablierung des eigenen Formates CU-RTC.[23] Es ist nicht abzusehen, welches der Formate sich letztendlich durchsetzen wird, auch nicht in welchem Ausmaß diese neuen Standards und Technologien Verbreitung finden werden. Es gibt allerdings mit sipML5 eine Implementierung eines in HTML5 geschriebenen SIP-Clients, der auf WebSockets und WebRTC aufsetzt, und so mit dem Browser ohne zusätzliche Plugins Telefonate ins reguläre Telefonnetz ermöglicht.[24]

2.3 Java-Web-Frameworks

Es gibt mindestens 150 in Java geschriebene Web-Frameworks, allein diese aufzuzählen sprengt den Umfang dieses Kapitels. Im übrigen ist dies nicht sinnvoll, da ein Großteil der Projekte tot ist und nicht weiterentwickelt wird.[25]

Eine in den Jahren 2009-2011 vermutlich in Deutschland bei professionellen Softwareentwicklern durchgeführte Umfrage ergab, dass in 80 % der Java-Projekte eines der Frameworks Spring MVC, GWT, Grails, Struts 2 oder JSF genutzt wird.[26] Der J2EE-Experte Matt Raible hat eine Tabelle[27] entworfen, in der er dreizehn Frameworks bezüglich verschiedener Kriterien bewertet und vergleichbar

21 http://caniuse.com/websockets
http://socket.io/#browser-support
22 Hoeller, “Next Stop: Spring Framework 4.0.”
http://jcp.org/en/jsr/detail?id=356
23 Schwan, “Formatkrieg Beim Videochat.”
24 http://sipml5.org/
25 Dies ergab eine Recherche im Rahmen des Plat_Forms-Projektes im September 2012.
26 Sohn and Taboada, *OIO Kompass: Java Web-Frameworks*.
27 http://bit.ly/jvm-frameworks-matrix

macht. Er betrachtet darin Struts 2, Spring MVC, Wicket, Tapestry, Stripes, GWT, Grails, Rails, Flex, Vaadin, Lift und Play.[28] Eine im Jahr 2010 im englischsprachigen Raum durchgeführte Umfrage kommt zum Schluss, dass fest jeder zweite der 1000 Teilnehmer Spring und Hibernate einsetzt, weiterhin werden JSF, GWT, Struts 1, Seam, Velocity, Struts 2, Wicket, Grails, Stripes und Tapestry genutzt.[29] Die drei Studien sind letztendlich nur ein Beleg für die Heterogenität des Java-EE-Ökosystems, in dem Spring und JSF eine herausragende Stellung haben.

Die Auswahl der Frameworks für diese Arbeit wurde eher anhand pragmatischer Aspekte getroffen: Spring MVC aufgrund seiner Popularität und Tapestry, da Uli Stärk, der Betreuer dieser Arbeit, Entwickler im Tapestry-Projekt ist. Weiterhin bietet sich JSF an, da man mit nur einer Implementierung eine Vielzahl von den JSF-Standard implementierenden Frameworks nutzen kann.

2.4 Apache Tapestry

Tapestry ist als Projekt der Apache Software Foundation unter der Apache License 2.0 lizensiert. Es bezeichnet sich als „component oriented framework“, das heißt, es versucht die technischen Details und Eigenschaften einer HTTP-Anfrage zu verbergen. Die Architektur basiert auf dem Inversion of Control (IoC) genannten Entwurfsmuster in Form von Dependency Injection. Tapestry ist ein Projekt der Apache Foundation, welches von Howard M. Lewis Ship geleitet wird, er hat die erste Version von Tapestry im Jahr 2000 erschaffen. Entscheidungen bezüglich des Projekts werden bei der Apache Foundation vom Project Management Committee getroffen, dem neun Personen angehören.

```
<dependency>
  <groupId>org.apache.tapestry</groupId>
  <artifactId>tapestry-core</artifactId>
  <version>5.3.6</version>
</dependency>
```

Text 2: Ausschnitt aus einer Maven-Konfigurationsdatei mit Verweis auf Tapestry

Tapestry ist zuletzt am 12. Oktober 2012 als Version 5.3.6 erschienen. Das Projekt empfiehlt eine Installation mit Maven.

2.5 Spring MVC

Spring ist ein Anwendungs-Framework für die Java-Plattform und unter der Apache-2.0-Lizenz nutzbar. Spring MVC ist Springs Web-Framework, es gehört zu den meistgenutzte Web-Frameworks für Java. Wie der Name bereits sagt, basiert seine Architektur auf dem MVC genannten Entwurfsmuster. Spring MVC wird häufig zusammen Kombination mit dem ORM-Framework Hiber-

28 Raible, “Comparing JVM Web Frameworks.”
29 ZeroTurnaround, *Java EE Productivity Report 2011*.

nate oder Spring Security eingesetzt.

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>3.1.1.RELEASE</version>
</dependency>
```

Text 3: Maven-Konfiguration für das Spring-Framework

2.6 JSR: Java-Web-Standards

Im Gegensatz zu anderen Web-Plattformen gibt es für Java formale Standards, die erhebliche Teile einer Anwendungsarchitektur abdecken. Ein Entwurf durchläuft den Java Community Process, an dem als Expert Group beteilige Parteien mitwirken, und wird letztendlich als Java Specification Request veröffentlicht.

Die Java Servlet API stellt das Kernstück der Java-Web-Architektur dar, sie ist die Schnittstelle zwischen Anwendungsserver und Web-Anwendung. Sie ist vergleichbar mit CGI oder Rack, allerdings deutlich umfangreicher.[30]

Alle verbreiteten Anwendungsserver implementieren die Java Servlet API, zum Betrieb von Webanwendungen werden häufig Tomcat oder Jetty eingesetzt.[31] Da Anwendungsserver üblicherweise hinter Proxy-Servern verborgen sind, gibt es keine aussagekräftigen Statistiken über die tatsächliche Verteilung. Als Indiz für die vorherrschende Stellung dieser beiden Server lässt sich aber anführen, dass Google für seine App Engine zunächst Tomcat und später Jetty eingesetzt hat.[32]

Die Java Persistence API spezifiziert eine Schnittstelle für OR-Mapper. Hibernate ist das verbreiteste ORM-Framework und kann sowohl über die JPA, aber auch über eine eigene Schnittstelle genutzt werden.

JavaServer Pages ist eine der ältesten Technologien zum Erzeugen von Web-Inhalten mit Java. Mit Tags in XML-Syntax können Inhalte in Templates eingebunden und somit die Ausgabe generiert werden. Die Tags können durch sogenannte Tag Libraries erweitert werden, die JavaServer Pages Standard Tag Library (JSTL) ist eine Sammlung solcher Bibliotheken.

---

30 Mordani and Roh, “JSR 315: Java Servlet 3.0 Specification”; Robinson and Coar, “RFC 3875: The Common Gateway Interface (CGI) Version 1.1.”
http://rack.github.io/

31 http://tomcat.apache.org/
http://jetty.codehaus.org/jetty/

32 Wickesser, “Google Chose Jetty for App Engine.”

JavaServer Faces ist ein Standard für Web-Frameworks, der das Erzeugen von Benutzerschnittstellen und die Benutzerinteraktion abdeckt, d. h. das Generieren von HTML, das Session-Management und die Internationalisierung.[33] Mit der Veröffentlichung von Version 2.0 des Standards wurde JSP durch Facelets als Template-Technologie abgelöst, um Probleme beim Zusammenspiel von JSP und JSF zu beheben.[34] Die bekanntesten Implementierungen von JSF sind die Referenz-Implementierung Mojarra und Apache MyFaces.

2.7 Funktionalitäten von Frameworks

Die von Web-Frameworks implementierten Funktionalitäten unterscheiden sich teilweise deutlich, was allerdings nicht immer auf einen unterschiedlichen Entwicklungsstand der Projekte zurückzuführen ist. Microframeworks wie Sinatra decken bewusst nicht das ganze Spektrum von Funktionen ab.

In Tabelle 1 ist für sieben populäre Frameworks angegeben, ob sie eine Funktionalität selbst implementieren (●) oder die Anbindung an eine externe Bibliothek existiert und üblich ist (○). Der Umfang oder die Qualität der Implementierung spielen hier weniger eine Rolle, als vielmehr ob es einen etablierten Weg gibt, ein Feature umzusetzen, und ob dies durch Code des Frameworks geschieht. Das „Fehlen“ einen Features bedeutet insbesondere nicht, dass eine Funktion mit diesem Framework nicht umsetzbar ist, sondern lediglich, dass das Framework dem Entwickler keine Vorgaben darüber macht, wie es umzusetzen sei.

33 Burns and Kitain, “JSR 314: JavaServer Faces 2.1.”

34 Sangeetha, Nitin, and Ananya, “JSF 2.0 Views: Hello Facelets, Goodbye JSP”; Bergsten, “Improving JSF by Dumping JSP.”

	Sprache	**Routing**	**Views / Templates**	**Input Validation**	**Session Handling**	**ORM**	**Form Handling**	**Authentication**	**Authorization**	**I18n / L10n**	**Scaffolding**	**Caching**	**Asset Management**	**Web services**	**AJAX**
Spring MVC	Java	●	●	●	●	○	●	○	○	●	○	●			
Tapestry	Java	●	●	○	●	○	●	○	○	●		●			○
Ruby on Rails	Ruby	●	●	●	●	●	●	○	○	●	●	●	●	●	●
Sinatra	Ruby	●	●		●										
Django	Python	●	●	●	●	●	●	●		●	●	●			
Symfony	PHP	●	●	●	●	○	●	●	●	●	●	●	○	○	○
JSF	Java	●	●	●	●	○	●	○	○	●					●

Tabelle 1: Web-Frameworks implementieren nicht alle Funktionalitäten selbst

2.8 Benchmarking

Um einen Benchmark von Web-Frameworks durchzuführen, benötigt man eine möglichst repräsentative Web-Anwendung, ein passendes Lastmuster und eine geeignete Metrik.

Die Spezifikation der Anwendung ist dabei eine der größten Herausforderungen. Web-Anwendungen enthalten zwar oft wiederkehrende Elemente, beispielsweise ein Login-Formular, können aber unterschiedliche Charakteristika aufweisen, nämlich den Anteil der angemeldeten Nutzer. Dieser kann problemlos zwischen null und hundert Prozent schwanken. Dies kann dramatische Unterschiede in der Performance bedeuten, da sich Inhalte für nicht angemeldete Nutzer oft leichter zwischenspeichern lassen.[35] Weiterhin beinhalten Web-Anwendungen immer wieder neue Funktionen wie beispielsweise APIs, die sich in immer mehr Anwendungen finden lassen.[36] Ob die Performance einer bestimmten Funktionalität relevant ist, hängt von den konkreten Anforderungen eines Projektes ab. Drittens haben die unterschiedlichen Eigenschaften von Web-Anwendungen hinsichtlich ihres Umfangs oder ihrer Natur bei Web-Frameworks zu Spezialisierungen geführt.

Um diesen Gegebenheiten gerecht zu werden, betrachtet der hier vorliegende Benchmark die Performance der Frameworks innerhalb mehrerer Teilbereiche. Die Auswahl dieser Teilbereiche stützt sich auf eine Betrachtung aus drei Perspektiven: Funktionen von Anwendungen, Features von Web-

35 Für einen Leser der Wikipedia ist es kein Problem, eine mehrere Minuten alte Seite zu betrachten. In Form einer „gesichteten Version“ ist dies sogar ausdrücklich erwünscht. Ein Autor hingegen muss die letzte Änderung angezeigt bekommen, beispielsweise um zu verhindern, dass diese versehentlich überschrieben wird. Mehr Informationen zur Architektur der Wikipedia findet man bei Bergsma, “Wikimedia Architecture.”

36 Taylor, “Codecademy Partners With Twitter, Evernote, Box, And Others To Offer A Suite Of New API Lessons.”

Frameworks und Bestandteile bestehender Benchmarks.

Auch die Wahl des Lastmusters hat eine große Auswirkung auf die Aussagekraft eines Benchmarks. Anwendungen können sich beispielsweise hinsichtlich ihres Verhaltens bei gleichzeitigen Anfragen unterscheiden. Ein Benchmark, der eine Messung mit für die jeweilige Anwendung ungewöhnlich wenig gleichzeitigen Anfragen durchführt, wird keine aussagekräftigen Ergebnisse liefern.[37] Zur Generierung von Anfragen und Durchführung von Lasttests gibt es eine Reihe von Werkzeugen, die sich hinsichtlich ihres Funktionsumfangs, ihres Ressourcenverbrauchs und ihrer Genauigkeit unterscheiden.[38] Zu den bekanntesten gehören ApacheBench, http_load, httperf, siege, sowie das deutlich umfangreichere JMeter.

Ein weiterer wichtiger Bestandteil eines Benchmarks ist die Wahl der Metrik.[39] Unter „requests per second“ versteht man die Anzahl von Anfragen, die das untersuchte System pro Sekunde beantworten kann. Diese Eigenschaft wird auch als Durchsatz bezeichnet. Dabei muss sichergestellt werden, dass die Anfragen richtig bearbeitet wurden und die Antworten nicht etwa Fehlermeldungen oder leere Dokumente enthalten. Im Kontext einer Anwendung besteht eine Anfrage häufig nicht aus einem, sondern aus mehreren HTTP-Requests. Diese inhaltlich zusammenhängenden Anfragen, sie können sich beispielsweise die selbe Session nutzen, werden als Transaktion bezeichnet.

Eine weitere Messgröße ist die „response time“, also die Antwortzeit. Bei HTTP-Benchmarks unterscheidet man zudem nach „time to first byte“ (TTFB) und „time to last byte“ (TTLB), also dem Zeitpunkt, an dem das erste bzw. letzte Byte der Antwort beim Client angekommen ist. Dies ist anzustreben, da Browser mit dem Abrufen weiterer referenzierter Dateien und der Darstellung einer Seite beginnen bevor sie komplett übertragen wurde.

Eine weitere Differenzierung kann mit Werkzeugen erfolgen, die mit Hilfe der Navigation Timing API[40] und JavaScript-Events den Fortschritt des Seitenaufbaus im Browser messen. Dies kann insbesondere dann nötig werden, wenn die Ladezeit der Seite im Browser von Zugriffen auf externe Server abhängt. Da der Betreiber keine Kontrolle über diese Server hat, kann der tatsächliche Einfluss auf die Ladezeit nur im Browser gemessen werden. Diese Methode wird daher auch vornehm-

37 Vgl. Pereira et al., “Assessing the Impact of Reactive Workloads on the Performance of Web Applications”; Zinke, Habenschuss, and Schnor, “Servload.”

38 „totally unscientific (but quite useful) comparison“: Lev Walkin, “Stress-testing Httperf, Siege, Apache Benchmark, and Pronk.”; Zur Anfälligkeit eines Java-basierenden Werkzeuges: Hashemian, Krishnamurthy, and Arlitt, “Web Workload Generation Challenges – an Empirical Investigation.”

39 Zum Einfluss der Metrik auf das Ergebnis eines Benchmarks: Feitelson, “Metric and Workload Effects on Computer Systems Evaluation.”

40 https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/NavigationTiming/Overview.html

lich für Messungen während des laufenden Betriebes und nicht für Lasttest eingesetzt.

Betrachtet man sowohl den Durchsatz als auch die Antwortzeit, d. h. die Anzahl der Anfragen, die ein System innerhalb einer vorgegebenen Reaktionszeit beantworten kann, erhält man eine gute Aussage über die Leistungsfähigkeit des Systems. Dies setzt natürlich realistische Annahmen über das Nutzungsverhalten voraus.

Aus technischer Sicht gibt es noch weitere wichtige Messwerte bezüglich des Ressourcenverbrauchs. Dazu zählen die Nutzung der CPU, die entweder in Prozent oder in Form von CPU-Sekunden angegeben wird und die Menge des genutzten Arbeitsspeichers, die „Load“ des Systems, ein Wert bei unixartigen Systemen, der angibt, wie viele Prozesse gleichzeitig ausgeführt werden oder darauf warten, ausgeführt zu werden. Weiterhin gibt es relevante Werte der Laufzeitumgebung, für Java sind das u. a. die Heap Allocation Rate und die Garbage Collection Time.[41]

Die Laufzeitumgebung spielt ebenfalls eine Rolle. Bei einer Java-Web-Anwendung sind u. A. folgende Komponenten beteiligt: Der Anwendungsserver / Servlet-Container, die Java-Laufzeitumgebung, der Web-Server, das Betriebssystem, ggf. auch ein Datenbank-Server und die Hardware der Systeme. Will man absolute Werte ermitteln, habe alle Komponenten Auswirkungen auf das Ergebnis. Bei einem Vergleich von Web-Frameworks kann man möglicherweise den Web-Server, das Betriebssystem und die Hardware vernachlässigen, allerdings nur, wenn die Auswahl der beiden Komponenten typisch für den Anwendungsfall ist, und sie keines der Frameworks bevorzugt oder benachteiligt. Bei einem Vergleich von Java-Web-Frameworks trifft dies u. U. auch auf die Java-Laufzeitumgebung und den Servlet-Container zu.

Der Umfang eines Benchmarks wird normalerweise durch den Umfang des betrachteten Systems festgelegt. Dies ist im hier vorliegenden Fall aus zwei Gründen schwierig: Die Frameworks unterscheiden sich deutlich in ihrem Funktionumfang, und es ist sehr schwer, Aussagen über die mit ihnen implementierten Anwendungen zu treffen.

2.9 Bestehende Benchmarks

Ein Benchmarking von Web-Anwendungen wird in der professionellen Web-Entwicklung regelmäßig durchgeführt. Meistens sollen solche Benchmarks die Leistung eines einzelnen Systems messen oder bei der Suche nach Performance-Problemen helfen. Bei der Auswahl von Frameworks spielt dessen Performance offenbar eine geringere Rolle, teilweise wird sie sogar komplett ignoriert.[42]

41 Gousios, Karakoidas, and Spinellis, “Tuning Java’s Memory Manager for High Performance Server Applications.”
42 Buchner, Bottcher, and Storch, “Evaluation of Java-based Open Source Web Frameworks with Ajax Support.”

Im folgenden sind alle Benchmarks beschrieben, die sich mit der Performance von Web-Frameworks befassen. Die Durchführung ist von sehr unterschiedlicher Qualität, die jeweiligen Ergebnisse sind mit großer Wahrscheinlichkeit nicht verallgemeinerbar. Trotzdem kann man davon ausgehen, dass jeder Benchmark ein legitimes Informationsbedürfnis des Autors bedient hat.

CloudStone[43] / Apache Olio[44]

Der Cloudstone-Benchmark aus dem Jahr 2008 gehört zu den aufwändigsten und gründlichsten Benchmarks. Das Ziel des Teams von der UC Berkeley und Sun Microsystems war es, die Auswirkungen von damals neuartigen Web-2.0-Anwendungen auf die damals ebenfalls neuen IaaS-Dienste zu ermitteln. Dafür entwickelten sie eine passende Anwendung mit „sozialen“ Features wie Tagging, Mashups, Uploads und Feeds. Diese implementierten sie in Ruby on Rails, PHP und Java. Um die Ressourcennutzung bei dynamisch nutzbaren Cloud-Diensten quantifizieren zu können, schlagen sie eine neue Metrik vor: „dollars per user per month“. Die Lastgenerierung geschah mit Faban.

Aus den Benchmark-Ergebnissen wird ersichtlich, dass die Konfiguration einen erheblichen Einfluss auf die Performance der Anwendung hat.

> *„We have identified many degrees of freedom in deploying Web 2.0 applications—caching, hardware configuration, tuning, and options we haven't even considered such as hardware SSL acceleration and alternative interpreters“*

Die Anwendungen wurden unter dem Namen Olio als Projekt im Incubator der Apache Foundation veröffentlicht, mit dem Ziel, dass weitere Implementierungen beigesteuert werden. Nimmt man die Mailing-Listen als Indikator, gab es nach 2009 keine nennenswerte Aktivität in diesem Projekt.

Workload Characterization of selected JEE-based Web 2.0 Applications[45]

Dieser Benchmark aus dem Jahr 2008 wurde von IBM durchgeführt, mit dem Ziel die Auswirkungen der aufkommenden Web-2.0-Anwendungen auf ihr IBM Power5-System zu bestimmen. Ihre Infrastruktur bezeichnen sie als „state of the art enterprise application deployment“. Es ist aber kein Benchmark von Web-Anwendungen, sondern einer mit Web-Anwendungen. Sie nutzten drei Anwendungen (Lotus Connections, Java Pet Store 2.0, Trade6), als Anwendungsserver WebSphere und Glassfisch. Die Lasterzeugung geschah mit Grinder und sie betrachten Auswertungen auf CPU- und RAM-Ebene. Sie kommen zu dem Schluss, dass es zwar interessante Unterschiede, aber keine si-

43 Sobel et al., *Cloudstone*.
44 http://incubator.apache.org/olio/
45 Nagpurkar et al., “Workload Characterization of Selected JEE-based Web 2.0 Applications.”

gnifikanten Auswirkungen auf der Server-Seite beim Vergleich von sog. Web-1.0- und -2.0-Anwendungen gibt.

Perfbench[46]

Der Benchmark entstand aus einem Blog-Eintrag von Peter Thomas, der die Geschwindigkeit der Java-Frameworks Seam / JSF und Wicket verglich. Er betrachtet die durchschnittlichen Ladezeiten der einzelnen Seiten seiner Beispiel-Anwendung, sowie den Speicherverbrauch. Die Lasterzeugung geschah mit JMeter.

Der Eintrag erhielt größere Aufmerksamkeit, es gab mehr als 50 Kommentare. Einige Montate später wiederholte er den Benchmark mit Wicket, Tapestry, Seam und Grails, auch dieser Blog-Eintrag wurde deutlich über 50 mal kommentiert. Die Kommentare beziehen unter anderem sich auf die Neutralität des Autors, der dem Wicket-Framework nahesteht oder bemängeln die Verleichbarkeit der gewählten Frameworks.

> *„Peter thomas may not have a direct stake in wicket, but he is known to be a hardcore wicket fan. Somehow in all his articles wicket always wins everything.“*

> *„So you have to either compare Wicket with JSF or Seam/Wicket with Seam/JSF. You can't compare Seam/JSF with Wicket because then you have the overhead of Seam only one one side.“*

Der Kern des Benchmarks ist eine Beispiel-Anwendung des Seam-Projektes mit dem Namen „Hotel Booking“, die einige häufig genutzte Funktionalitäten von Web-Anwendungen abdeckt: Templates, Authentifizierung, Autorisierung, Sessions, AJAX, Datenpersistenz und -validierung.[47]

Den Quellcode der Anwendungen hat Thomas auch auf Google Code veröffentlicht, zusammen mit einem Leitfaden, der Regeln für die Implementierungen enthält. Im Kontrast zur regen Beteiligung in den Kommentaren der Blog-Einträge gibt es dort keine erkennbare externe Beteiligung.[48]

PHP Framework MVC Benchmark[49]

Der Benchmark betrachtet die durchaus beeindruckende Zahl von zwölf PHP-Frameworks. Für jede hat der Autor eine „simple Hallo-Welt-Seite und simple MVC-Logik“ implementert. Bei genauerem Hinsehen fällt allerdings auf, dass die Model-Komponente gar nicht genutzt wird und dementsprechend der View auch keine Daten enthält. Die Lasterzeugung geschah mit dem Werkzeug ab, die

46 Thomas, “Seam / JSF Vs Wicket”; Thomas, “‘Perfbench’ Update: Tapestry 5 and Grails.”
47 https://github.com/seam/examples/tree/master/booking
48 https://code.google.com/p/perfbench/
49 Rui, “PHP Framework MVC Benchmark - V20111201-4.”

Anwendungen wurden aus dem Arbeitsspeicher ausgeführt. Ausgewertet wurden die Anzahl der Zugriffe pro Sekunde bei unterschiedlich vielen gleichzeitigen Verbindungen, der Load-Average-Wert, der Speicherverbrauch, die durchschnittliche Zugriffszeit, sowie die Anzahl der Funktionsaufrufe und die der geöffneten Dateien bei einem Seitenaufruf.

Die Ergebnisse aus dem Jahr 2012 zeigen deutliche Unterschiede in der Geschwindigkeit. Das schnellste Framework bearbeitet mehr als zwanzig mal so viele Zugriffe wie das langsamste. Die schlechte Platzierung umfangreicher Frameworks legt nahe, dass der Umfang der Test-Anwendung einen erheblichen Einfluss auf die Ergebnisse hatte.

Der Autor pflegt den Benchmark auf Github und hat dort auch vereinzelt Aktualisierungen von anderen Nutzern erhalten.

Webofcontacts[50]

Webofcontacts ist eine Web-basierte Anwendung zum Verwalten von Kontakten, die der Autor zum im Jahr 2009 zum Vergleich von folgenden Web-Frameworks genutzt hat: Struts 1, Struts 2, JSF, Ruby on Rails, Spring MVC und GWT. Der Autor veröffentlichte das Projekt zunächst auf seiner Hompage, später migrierte er den Quellcode und einen Teil der Dokumentation zu Google Code.[51] Die Anwendung implementiert rudimentäre Funktionalitäten, das Anlegen, Auflisten und Anzeigen von Kontakten. Das Projekt dient dem Vergleich von Frameworks, hat aber nicht einen Performance-Benchmark als Ziel. Die Projektseite gibt sieben weitere Beitragende an, deren Aktivität scheint allerdings nur aus vereinzeltem Feedback bestanden zu haben. Nach dem Jahr 2009 gab es keine nennenswerte Aktivität mehr.

WWW World Wide Wait[52]

Der Performance-Benchmark mehrerer Java-Frameworks wurde im Jahr 2011 von Stijn Van den Enden auf mehreren Java-Konferenzen vorgestellt. Die bei der Präsentation verwendeten Folien sind gleichzeitig die einzige öffentliche Quelle an Informationen zu diesem Benchmark, die dementsprechend Lückenhaft sind. Van den Enden hat demnach mit zwei Kollegen eine Twitter-artige Anwendung mit AJAX-Funktionen für die Frameworks Wicket, JSF (Mojarra, MyFaces), Spring und GWT implementiert. Sie enthielt „common patterns“ wie eine Überblicksseite, eine Detailseite, und verschiedene AJAX-Elemente, die Implementierungen bestanden je nach Framework

50 McCabe, “Webofcontacts: A Comparison of Web Application Frameworks.”
51 http://oranda.com/webapp/
52 Van den Enden, “WWW: World Wide Wait. A Performance Comparison of Java Web Frameworks.”

aus 130 bis 640 Zeilen Code. Die Lasterzeugung geschah mit JMeter. Das Team evaluierte ebenfalls eine browserbasierte Messung der Ladezeiten mit Hilfe eines Selenium-Clusters, verwarf dies aber aufgrund der Nichteignung von Selenium für Performance-Bechmarks. Außerdem hätten die im Browser ermittelten Ladezeiten sich lediglich um einen konstanten Wert von den mit JMeter ermittelten Werten unterschieden. Die Benchmarks wurden auf Amazons IaaS-Plattform EC2 durchgeführt, dazu passend sind die Ergebnisse in der Form „Kosten in USD für 10.000 Nutzer“ angegeben. Außerdem ermittelten sie CPU-Auslastung, den Durchsatz und die durchschnittliche Ladezeit für verschiedene Mengen gleichzeitiger Nutzer.

JT Dev[53]

In diesem, ebenfalls in Form eines Blog-Beitrages vorliegenden, Benchmark vergleicht der Autor sechs Frameworks, Ruby on Rails, Wicket, Grails, Play, Tapestry, Lift, JSP und Context. Bis auf die Ausnahme Ruby on Rails sind das alle Frameworks für die Java-Plattform. Der Autor betrachtet zwei Anwendungsfälle, das Anzeigen einer Liste von Produkten und das einer statischen Datei. Als Metrik nutzt der Autor die durchschnittliche Antwortzeit und die CPU-Auslastung, während er die Anzahl der gleichzeitigen Nutzer und die Anzahl der in der Testanwendung angezeigten Produkte variiert. Als Lastgenerator JMeter, welches auf der selben Hardware ausgeführt wird, weswegen es zu Beeinflussungen der Anwendungen gekommen ist.

> *„CPU utilization of JMeter sometimes reaches 50%. In these cases, my CPU utilization of the associated framework will be very inaccurate.“*

Für einige Frameworks testet der Autor mehrere Konfigurationen, das heißt verschiedene Versionen des Frameworks, mehrere Anwendungsserver oder eine andere Laufzeitumgebung. Auch er berichtet von dramatischen Unterschieden in den Ergebnissen durch Änderungen in der Konfiguration.

> *„Simply switching from template system in Play framework from the default to Japid, has a huge impact (in this test 27 times faster).“*

Der für den Benchmark genutzten Code ist auf Github veröffentlicht, wo Implementierungen für Grails und Tapestry begesteuert wurden. Auch wenn der Autor in seinem Blog nicht seinen vollen Namen angibt, lassen die über 100 Kommentare zu den Einträgen annehmen, dass es eine gewisse Reichweite besitzt. Beispielsweise merkt der Tapestry-Gründer Howard M. Lewis Ship an, dass die Implementierung für Tapestry einige untypische Konstrukte benutzt. Andere Kommentare bitten darum weitere Frameworks einzubeziehen, haben Nachfragen oder Vorschläge bezüglich der Konfi-

53 JT Dev, “Rails, Wicket, Grails, Play, Tapestry, Lift, JSP, Context.”

guration. Während die Code-Änderungen ausschließlich innerhalb der zwei Wochen nach Veröffentlichung des Benchmarks stattfinden, geht die Diskussion im Blog weiter. Über ein Jahr nach Veröffentlichung meldet sich Ship erneut und bittet um die Berücksichtigung einer soeben neu erschienenen Tapestry-Version mit Performance-Verbesserungen.

Understanding JSF 2 and Wicket: Performance Comparison[54]

Der Benchmark aus dem Jahr 2012 führt den knapp drei Jahre vorher von Peter Thomas durchgeführten Vergleich von Wicket und JSF-Implementierungen fort.[55] Der Autor Leonardo Uribe ist Mitglied des Apache-MyFaces-Projektes, der populärsten JSF-Implementierung und will die in den vorangegangenen Jahren vorgenommenen Performance-Verbesserungen präsentieren und erläutern.

Er weist auf einige seiner Meinung nach ernste Probleme in der bestehenden Implementierung hin die er korrigiert und auf Github veröffentlicht hat. Dort sind ebenfalls Konfigurationsdateien, weitere Angaben zur Konfiguration sowie die Ergebnisse der Benchmarks in Rohform zu finden. Als Metriken nutzt er den Median und das 90. Perzentil der Antwortzeit bei einer variablen Menge gleichzeitiger Anfragen, der genutzte Speicher innerhalb der Java-VM und die Größe einer serverseitig vorgehaltenen Nutzersession.

Bei einem Test der Frameworks unter Last (40 gleichzeitige Verbindungen) ist sehr auffällig zu sehen, wie unterschiedlich die Ergebnisse für verschiedene Operationen der Testanwendung sein können. Die beiden Wicket-Versionen sind bei fast allen Operationen vergleichbar schnell oder schneller als die beiden JSF-Implementierungen, während sie bei der Operation „post book hotel“ deutlich schlechter abschneiden. Auch Uribe hält die Ergebnisse nicht für allgemeingültig:

> *„Please note these graphs cannot be generalized, and they depend strongly on the test application, but they are the best indication available at the moment.“*

The Great Web Framework Shootout[56]

„The great web technology shootout“ ist eine Serie von Blog-Artikeln, in denen der Autor ein Benchmark von 13 Web-Frameworks der Plattformen Python, Ruby, PHP beschreibt. Als Metrik nutzt er Anfragen pro Sekunde, bei zehn gleichzeitigen Verbindung. Er hat für jedes Framework drei Anwendungsfälle implementiert, eine statische Seite mit dem Text „Hallo Welt“, eine Seite die Text in ein Template einsetzt, sowie eine Seite die Text aus einer SQLite-Datenbank liest und in ein Template schreibt. Die Lasttests wurden auf Servern von Amazons EC2 durchgeführt, die Lastgene-

54 Uribe, *Understanding JSF 2 and Wicket: Performance Comparison*.
55 „Perfbench“, S. 21
56 Davis, “The Great Web Framework Shootout.”

rierung geschah mit ApacheBench. Der Autor hat den Code auf Github veröffentlicht, externe Beteiligung gab es trotz Aufforderung nicht.

The performance test of 6 leading frameworks[57]

Der Benchmark aus dem Jahr 2007 gehört zu den älteren Benchmarks von Web-Frameworks. Er betrachtet die Frameworks CodeIgniter, Catalyst, Symfony, Pylons, Ruby on Rails, CakePHP und Turbogears hinsichtlich ihres Speicher- und CPU-Verbrauches sowie der durchschnittlichen Antwortzeit und des Durchsatzes bei unterschiedlichen Mengen gleichzeitigen Verbindungen. Die genutzte Anwendung liest einen Text („Hallo Welt“) aus einem Model und schreibt ihn in ein Template. Die Lasttest wurden mit ApacheBench, http_load und siege durchgeführt.

Framework Benchmarks[58]

Der Benchmark der amerikanischen Firma TechEmpower ist der jüngste der hier betrachteten Benchmarks. Die Testfälle sind sehr einfach gehalten und umfassen das Serialisieren eines JSON-Objektes bzw. das vorherige Auslesen von Datenbank-Records. Als Metrik nutzen sie die „Peak Performance“, wie sie das beste Ergebnis des bei unterschiedlich vielen gleichzeitigen Verbindungen gemessenen Durchsatzes bezeichnen. Bei späteren Ergebnissen geben sie außerdem die vom Benchmark-Werkzeug wrk ermittelte Latenz als Mittelwert zusammen mit der Standard-Abweichung und dem Maximum an. Die Benchmarks wurde auf Amazon EC2 und auf eigener Hardware durchgeführt. Nicht zuletzt der sehr einfachen Beispiel-Anwendung geschuldet finden sich deutliche Performance-Unterschiede in den Ergebnissen. Das auffällig gute Abschneiden des von TechEmpower entwickelten kommerziellen Frameworks lässt die Frage aufkommen, ob es sich bei dem Benchmark nicht eher um eine Marketing-Aktion handelt. Der Benchmark ist auf Github zu finden, zu rund zwanzig Implementierungen gab es nach Angaben der Autoren externe Beiträge. Das Projekt nicht noch aktiv, derzeit gibt es zehn sog. Pull-Requests und regelmäßige Updates des Repositorys.

Mojarra vs. MyFaces performance[59]

Der im Jahr 2011 veröffentlichte Blog-Eintrag unterscheidet sich von den bisher erwähnten Benchmarks. Der Autor möchte eine Behauptung bezüglich der Performance Apache MyFaces wiederlegen, da er zwei JSF-Implementierungen vergleicht, kann er Anwendungstests benutzen, um die bei-

57 Aster, “The Performance Test of 6 Leading Frameworks.”
58 TechEmpower, Inc., “Framework Benchmarks.”
59 Punz, “Mojarra Vs. MyFaces Performance.”

den Frameworks zu vergleichen. Die Messung der Geschwindigkeit nimmt er im Browser vor, da die Test AJAX beinhalten.

3 Durchführung

3.1 Erwartete Probleme

Zu Beginn der Arbeit stand die technische Umgebung nicht fest. Ein Web-Frameworks baut auf eine Vielzahl weiterer Komponenten auf, dazu gehören unter anderem ein Anwendungsserver, Datenbankserver, und Bibliotheken für OR-Mapping, Authentifizierung, Autorisierung. Bei einem Benchmark hat auch die Umgebung, also das Betriebssystem, die Java-Laufzeitumgebung oder die Hardware einen erheblichen Einfluss auf die Leistung. Ebenso gibt es eine erhebliche Anzahl von Konfigurationseinstellungen.

Es wurde angenommen, dass die Frameworks unterschiedliche Bibliotheken verwenden oder die Verwendung von unterschiedlichen Versionen empfehlen. Für den Benchmark ist dies dahingehend problematisch, da entweder gemessene Leistungsunterschiede nicht mehr ausschließlich dem Framework zuzurechnen sind oder das Framework in einer unüblichen Konfiguration genutzt wird. Die Frameworks haben außerdem unterschiedliche Zielgruppen und Nutzungsprofile.

Diskussionen bei bestehenden Benchmarks ließen erwarten, dass es bei einer Veröffentlichung der Ergebnisse zu Nachfragen und Anregungen bezüglich der Konfiguration und der Umgebung kommen würde.

> *„Few comments about rails: webrick is slow, for ruby 1.9 use thin [...] not sure what was your products model, but you definetely should have done this in controller (...)“*[60]

Dies ist insbesondere ernstzunehmen, wenn der Durchführende keine Erfahrung mit dem Einsatz der beteiligten Technologien hat. Die Validität der Ergebnisse hängt auch davon ab, ob Beteilige aus dem Umfeld der Frameworks den Benchmark und seine Umgebung als geeignet betrachtet. Dies beinhaltet Betriebssystem, Laufzeitumgebung, Anwendungsserver und das Frameworks, jeweils mit ihrer Konfiguration und in unterschiedlichen Versionen.

> *„Can we have a test for grails 1.4 trunk? […] There are several improvements like static invokations in GSP, static injections in controllers, static resources cache…“*[61]

Da sich die Frameworks bezüglich ihrer Architektur unterscheiden und ich keine Erfahrung mit ihnen hatte, war eine Planung des Entwicklungsprozesses des Benchmarks realistisch nicht möglich.

60 Aus den Kommentaren zu JT Dev, “Rails, Wicket, Grails, Play, Tapestry, Lift, JSP, Context.”
61 Ibid.

3.2 Iteratives Vorgehen

Vor dem Hintergrund der zu erwartenden Schwierigkeiten wurde bei dieser Arbeit iterativ vorgegangen. Iteratives Vorgehen bedeutet hier, möglichst früh einen funktionierenden Prototypen zu erstellen und diesen dann schrittweise weiterzuentwickeln.

In Bezug auf diese Arbeit bedeutet dies, dass bereits zwei Wochen nach dem Beginn in Kalenderwoche 47 ein Benchmark mit `ab` auf den Servern des Fachbereiches stattfand. In der darauf folgenden Woche wurde dann die Automatisierung des Vorgangs mit Puppet umgesetzt: Das Deployment einer Anwendung aus einem Git-Repository, das Durchführen des Benchmarks mit ApacheBench und eine rudimentäre Auswertung mit R. In den darauf folgenden Woche folgten Versuche mit dem Framework Spring MVC. In Woche 49 habe ich an der Automatisierung des Anlegens neuer Container gearbeitet, sowie Probleme mit der Netzwerkverbindung und Namensauflösung behoben. Darauf folgte eine Recherche zu Amazons EC2 IaaS-Plattform, die wegen ihrer universellen Verfügbarkeit eine interessante Infrastruktur für Benchmarks ist. In Woche 51 habe ich mich damit beschäftigt, wie das Lastmuster und Transaktionen in einem Benchmark zu berücksichtigen sind. In der ersten Hälfte den neuen Jahres habe ich mich mit verschiedenen Datenquellen beschäftigt. Anschließend begann eine längere Recherche zu bestehenden Benchmarks von Frameworks, der Architektur von Web-Anwendungen und Performance-Messungen, sowohl in wissenschaftlichen Veröffentlichungen, als auch im Internet. Im März folgte die Konzeptionen der Teilbenchmarks. Im April habe ich das Erstellen von Auswertungen ergänzt und an die vorliegende Arbeit vervollständigt.

An den längeren Zeiträumen in der Beschreibung des Vorgehens ist auch die Abkehr vom iterativen Vorgehen zu erkennen. Dies ist dem Versuch geschuldet, die Konstruktion der Teilbenchmarks weniger von einer technischen sondern mehr von einer konzeptionellen Perspektive zu betrachten.

3.3 Infrastruktur

Ein Benchmark ist grundsätzlich nur in Bezug auf die verwendete technische Infrastruktur aussagekräftig, wenn für diese kein eigenständiger Benchmark existiert. Durch die Popularität von IaaS-Diensten gibt es allerdings die Möglichkeit, den Benchmark auf einer Plattform durchzuführen, die auch für Dritte zugänglich ist. Marktführer bei IaaS-Plattformen ist derzeit Amazons EC2.[62] Ressourcen wie CPU drückt Amazon mit eigenen Einheit aus, den „EC2 Compute Units“.[63] Es gibt vie-

62 Cohen, “Gartner Announces 2012 Magic Quadrant for Cloud Infrastructure as a Service.”

63 „Eine EC2 Compute Unit bietet die entsprechende CPU-Kapazität eines 1,0-1,2 GHz Opteron oder Xeon Prozessors von 2007. Dies entspricht außerdem einem 1,7-GHz-Xeon-Prozessor von Anfang 2006, auf den in unserer Originaldokumentation verwiesen wird.“ Amazon Web Services, Inc, “Amazon EC2-Instancetypen.”

le Benchmarks von den einzelnen Angeboten auf der EC2-Plattform, was eine Vergleichbarkeit und Einordnung besser möglich macht.[64]

Aus Kostengründen fand der Entwicklungsprozess auf drei virtuellen Maschinen statt, die vom Fachbereich Informatik, der Freien Universität Berlin zur Verfügung gestellt wurden. Die Maschinen besaßen jeweils aus zwei CPU-Kerne eines Intel Xeon E5520 und 6 GB Arbeitsspeicher.

4 Ergebnisse

Die Ergebnisse dieser Arbeit bestehen in einer Reihe von Teilbenchmarks, die auf Betrachtungen der bestehenden Benchmarks und des Funktionsumfanges der Frameworks basieren. Ebenfalls wird eine Benchmark- und Auswertungsplattform vorgestellt, die versucht den Problemen, die bei bestehenden Benchmarks aufgetreten sind zu begegnen.

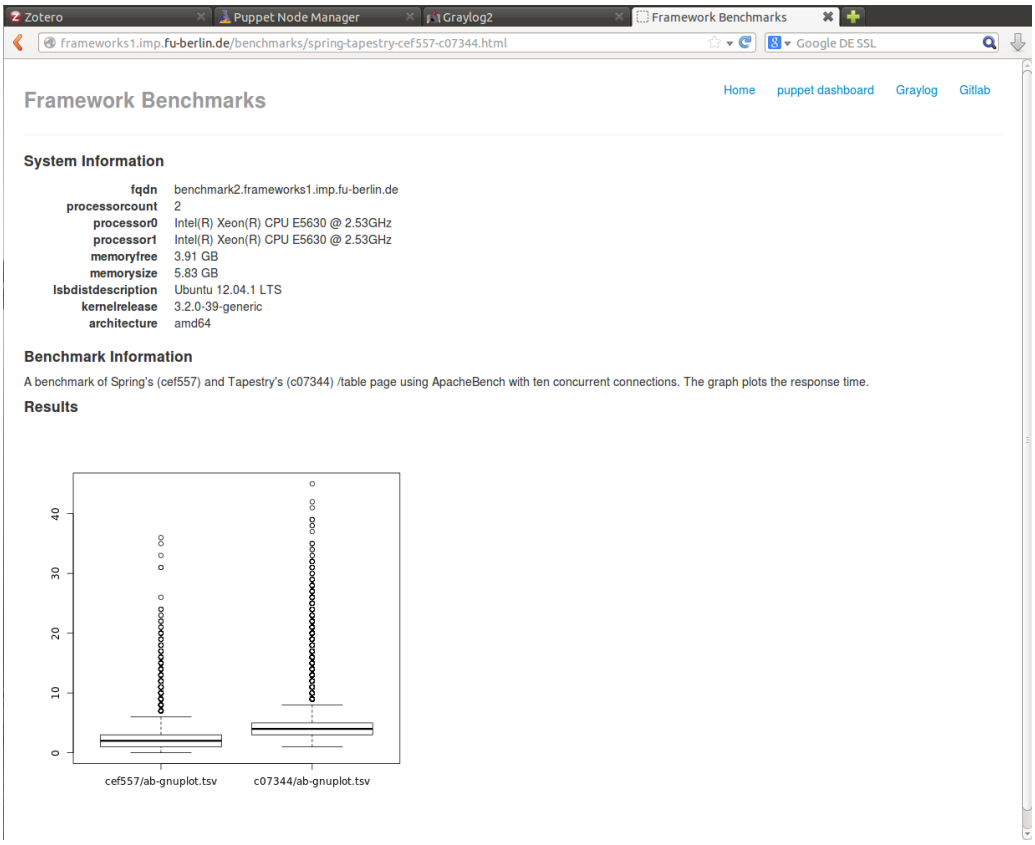

Abbildung 2: Beispiel einer Auswertung

Die im Rahmen dieser Arbeit umgesetzten Implementierungen reichen nicht, um Aussagen über die beiden betrachteten Frameworks zu machen. Die Implementierung eines Teilbenchmarks für Tapestry und Spring MVC dienen als Beispiel für den Funktionsumfang des im folgenden ebenfalls vorgestellen Plattform. Diese unterstützt verschiedene Anwendungsserver, Jetty und Tomcat, verschie-

64 Larabel, “Extensive Benchmarks Of Amazon’s EC2 Compute Cloud Performance.”

dene Versionen der Java-Laufzeitumgebung, die Lastgeneratoren ApacheBench und siege, eine Auswertung der Daten und Generierung von Plots mit R und eine abschließende Zusammenfassung als Web-Seite, wie sie in Abbildung 2 zu sehen ist.

4.1 Automatisierung des Prozesses

Wie zuvor beschrieben, soll der Benchmark zunächst auf Rechnern der Freien Universität und später bei Amazon durchgeführt werden. Eine erneute händische Installation der Benchmark-Umgebung ist allerdings mit großem Zeitaufwand verbunden. Bei jeder Replikation der Benchmarks besteht zudem die Gefahr, dass bereits durch kleine Änderungen der Konfiguration die Performance beeinflusst und damit die Ergebnisse verfälscht werden. Daher habe ich mich zu Beginn der Arbeit entschlossen, die Installation der Umgebungen innerhalb von Containern durchzuführen und zu automatisieren. Dies bietet zudem die Möglichkeit systematisch Veränderungen der Umgebung, d. h. verschiedene Anwendungsserver, Java-Laufzeitumgebungen oder Konfigurationseinstellungen betrachten zu können.

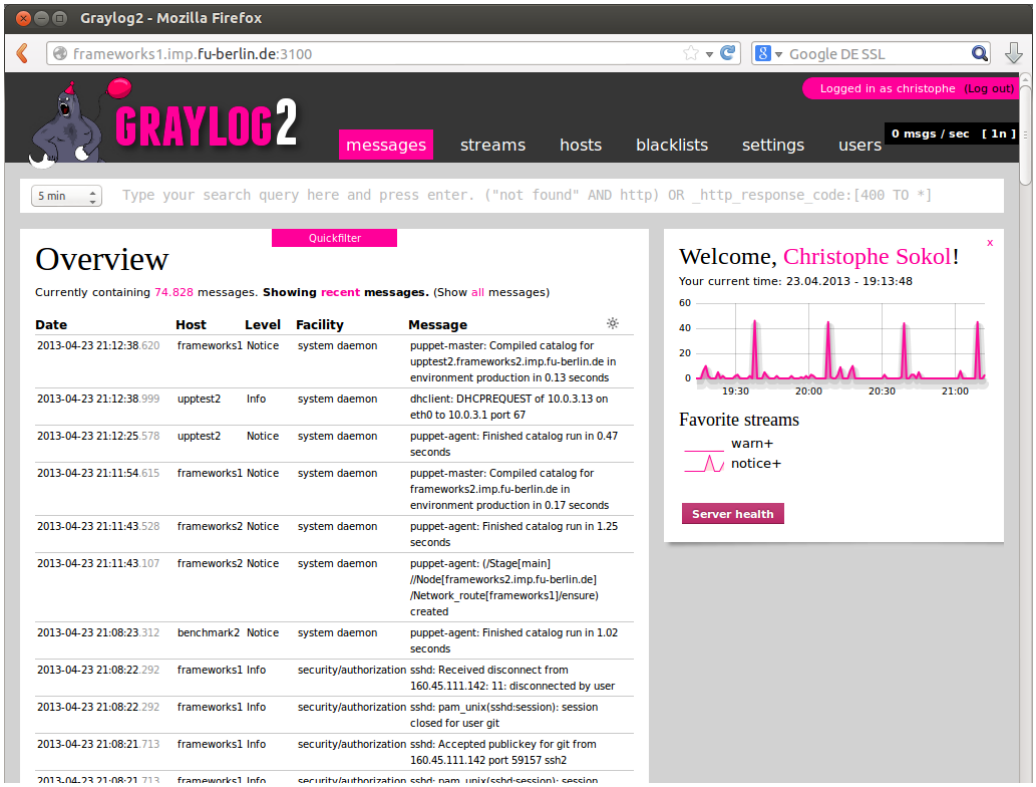

Abbildung 3: Graylog2

Ich nutze LXC[65] für die Kapselung der einzelnen Umgebungen, da das mir vertraute OpenVZ bei neueren Versionen von Debian und Ubuntu nicht mehr unterstützt wird.

65 http://lxc.sourceforge.net/

Für die Automatisierung standen nach einer ersten Recherche Chef[66] und Puppet[67] zur Auswahl. Ich habe mich für Puppet entschieden, da es als das populärere von beiden Werkzeugen eine größere Auswahl an Plugins eine breitere Unterstützung für weitere Komponenten bot, beispielsweise für die am Fachbereich genutzte vSphere-Virtualisierung. Um bei einer größeren Anzahl von virtuellen Maschinen nicht den Überblick zu verlieren, werden die Log-Einträge aller (virtuellen) Server zentral mit Graylog2[68] verwaltet. Das Auflösen und Installieren von Abhängigkeiten innerhalb der Anwendungen geschieht mit Maven[69].

Abschließend möchte ich noch zwei Werkzeuge erwähnen, die sich ebenfalls mit der Automatisierung von Benchmarks befassen: Für das CI-Tool Jenkins[70] gibt es ein Performance-Plugin[71], mit diesem man eine Performance-Messung mit JMeter in die Test-Automatisierung integrieren kann. Die Phoronix Test Suite[72] erstellt Performance-Messungen für eine Reihe von Open-Source-Programmen auf dem eigenen System, anschließend lassen sich diese auf einer zugehörigen Web-Plattform mit anderen Ergebnissen vergleichen.

4.2 Teilbenchmarks

Es gibt mehrere Möglichkeiten, einen Benchmark eines Web-Frameworks durchzuführen. Ein Mikrobenchmark würde die einzelnen Funktionen eines Benchmarks möglichst isoliert voneinander betrachten. Die Ergebnisse lassen sich allerdings nur miteinander vergleichen, wenn alle zu betrachtenden Frameworks die gleichen Funktionen bieten, beispielsweise alle Implementierungen des JSF-Standards. Eine weitere Möglichkeit ist die Implementierung einer Beispielanwendung mit allen Frameworks. Diese Beispielanwendung muss alle Eigenschaften besitzen, die mit den Frameworks möglicherweise umgesetzt werden. Dies ist aus mehreren Gründen schwierig: Wenn zwei Frameworks eine unterschiedliche Ausrichtung besitzen, z. B. sehr umfangreiche oder kleine Anwendungen, kann eine Beispielanwendung unmöglich beide erfüllen. Weiterhin ist es unklar, wie mit Funktionen der Frameworks verfahren werden soll, die nur von einem Teil der Frameworks umgesetzt werden. Werden sie ignoriert sind „schwergewichtige“ Frameworks im Nachteil. Ergänzt man sie innerhalb der Anwendung, wird diese untypisch für „schlanke“ Frameworks.

66 http://www.opscode.com/chef/
67 http://puppetlabs.com/puppet/what-is-puppet/
68 http://graylog2.org/
69 http://maven.apache.org/
70 http://jenkins-ci.org/
71 https://wiki.jenkins-ci.org/display/JEKINS/Performance+Plugin
72 http://www.phoronix-test-suite.com/

Daher besteht der im folgenden beschriebene Benchmark aus einer Reihe an Teilbenchmarks. Sie versuchen den von Web-Frameworks abgedeckten Funktionsumfang abzubilden und sind nach Funktionsbereich aufgeteilt.[73] Die starke Modularisierung der Frameworks macht dies möglich. Weiterhin gibt es Teilbenchmarks, die bei Web-Anwendungen häufig auftretende Anwendungs- und Problemfälle nachbilden.

Es wäre denkbar gewesen, die Spezifikationen der Teilbenchmarks formaler zu gestalten, beispielsweise durch die Angabe von Pfaden für einzelne Aktionen. Ich habe mich dagegen entschieden, da es den Spielraum für Entwickler unnötig einschränkt, ohne dass es bei der Durchführung nennenswerte Vorteile bringt. Wenn ein Framework eine Konvention bezüglich URLs hat, ist es nicht wünschenswert diese zu brechen.[74]

Falls nicht anders angegeben, werden bei jedem Teilbenchmark Lasttests mit einer sich erhöhenden Anzahl parallelen Verbindungen durchgeführt. Ich muss an dieser Stelle darauf verzichten, genaue Angaben über die Anzahl zu machen. Selbiges trifft auch auf die in den folgenden Teilbenchmarks genannten, zu variierenden Parameter zu. Mangels Erfahrung lassen sich diese nur experimentell bestimmen.

Bei jedem Teilbenchmark sind Variationen angeben, die ihn erweitern oder ein alternatives Vorgehen beschreiben. Ob eine solche Variation in der Praxis redundant ist oder aber sinnvoller ist als der Teilbenchmark selbst, wird sich ebenfalls zeigen müssen.

Hallo Welt

Mit einem Benchmark, der lediglich das Anzeigen einer einzelnen, statischen HTML-Seite betrachtet, kann man nur in den seltensten Fällen sinnvolle Aussagen über das genutzte Framework treffen. Dennoch ist er für eine Reihe von Dingen nützlich. Er liefert eine untere Schranke für die Performance des Frameworks, somit lassen sich bei komplexeren Testfällen die Veränderung beziffern. Mehrere bestehende Benchmarks nutzen einen vergleichbaren Anwendungsfall. Unter der Berücksichtigung der genutzten Hardware bietet es sich an die eigne Konfiguration zu verifizieren indem bestehende Ergebnisse reproduziert werden. Wie erheblich der Einfluss der Konfiguration auf die Ergebnisse ist, zeigen die bestehenden Benchmarks. Dieser Anwendungsfall stellt insbesondere hohe Anforderungen an den Lastgenerator.

73 Vgl. Kapitel 2.2

74 Falls sich zu einem späteren Zeitpunkt herausstellt, dass einheitliche Pfade ohne Einschränkung auf Seiten der Frameworks möglich sind, ist eine Standardisierung natürlich wünschenswert.

```
<!DOCTYPE html>
<html lang="no">
<head>
  <title>Hallo verd!</title>
</head>
<body>
  Hallo verd!
</body>
</html>
```

Text 4: Ein sehr einfaches HTML5-Dokument

Dieser Ansatz lässt sich als Variation auch auf weitere Komponenten übertragen. Bei Frameworks, die dem MVC-Muster folgen, steht der String „Hallo Welt“ in einem Anwendungsobjekt und wird von Controller in einer Variable des Templates geschrieben. In einer weiteren Variation, eher am Einsatzzweck einer REST-API orientiert, wird statt ein JSON-Objekt generiert und ausgegeben.[75]

Routing

Eine zentrale Aufgabe von Web-Frameworks ist das Routing, also das Verteilen von eingehenden HTTP-Anfragen. Der Benchmark simuliert eine sehr große Anwendung, indem er beispielsweise je nach Framework Seiten, Controller oder Views generiert, die in zufälliger Reihenfolge für verschieden viele erzeugte Route abgerufen werden.

Als Variation sind die Anfragen so zu gestalten, dass sie nur mit regulären Ausdrücken zu verarbeiten sind. Als eine weitere Variation ist eine verschieden große Anzahl von Parameter in den Routen zu betrachten.

Zufälliger Text

Dieser Benchmark simuliert die Ausgabe von Text und Daten in HTML-Seiten. Die Aufgabe ist die Ausgabe eines Templates, das mit zufällig generierten Zeichenketten gefüllt ist. Dabei muss die Anzahl der Zeichenketten und deren Länge variabel sein.

Die in Abbildung 2 gezeigte Auswertung basiert auf diesem Teilbenchmark mit fünfzig zufällig generierten Strings der Länge acht.

Eine Variation diese Benchmarks bindet jede Zeichenkette einzeln über eine Variable in das Template ein. Eine weitere Variation generiert zunächst Anwendungs-Objekte, die jeweils eine einzelne zufällige Zeichenkette enthalten und übergibt diese dann an das Template.

Seitenwechsel

Der Benchmark basiert auf Variation 2 des vorherigen Benchmark und und setzt voraus, dass das

75 Vgl. Rui, “PHP Framework MVC Benchmark - V20111201-4”; TechEmpower, Inc., “Framework Benchmarks.”

Framework die Funktionalität bietet, eine Menge von Anwendungsobjekten auf mehrere Seiten aufzuteilen und eine Navigation zu erzeugen. Jede dieser Seiten entspricht der Auflistung aus dem vorherigen Benchmark, um Links auf weitere Seiten ergänzt. Die Anzahl der Objekte und Seiten ist für jeden Benchmark festzulegen.

Als Variation dieses Benchmarks werden die Objekte aus einer Datenbank gelesen. Die Anzahl der Objekte sollte so gewählt sein, dass deren Größe den Speicher des Anwendungsservers deutlich übersteigt.

Binäre Daten

Der Benchmark besteht im Download einer binären Datei mit zufälligem Inhalt, beispielsweise aus `/dev/urandom`. Er simuliert eine auf dem Server generiertes PDF-Datei, beispielsweise eine Rechnung in einem Webshop. Die Größe der binären Datei und die Menge der gleichzeitigen Anfrage sind variabel.

Formular

Dieser Benchmark besteht aus mehreren HTTP-Anfragen. Zunächst ruft der Client ein Formular ab, das er anschließend ausgefüllt wieder an den Server sendet. Das Formular enthält alle verschiedenen Typen von Feldern, also Text-Felder, Auswahl-Felder, Ankreuz-Felder. Die Eingaben werden serverseitig validiert. Bei einem Text-Feld wird geprüft, ob die eingegebene Zeichenkette eine vorgegebene Länge nicht überschreitet, sich eine Zahl innerhalb eines definierten Bereiches befindet oder eine E-Mail-Adresse die korrekte Syntax hat. Bei einem Auswahl-Feld muss sichergestellt werden, dass die Auswahl aus den vorgegeben Wahlmöglichkeiten stammt. Bei Ankreuz-Feldern wird überprüft, ob sie angekreuzt wurden. Ist die Anfrage fehlerhaft, wird das Formular erneut angezeigt, wobei Felder die bereits eingetragenen Werte enthalten und die fehlerhaften Felder markiert sind. Bei einer erfolgreichen Validierung wird er auf eine Bestätigungsseite weitergeleitet, auf der die eingegebenen Werte erneut angezeigt werden.

Bei diesem Teilbenchmark variiert das Lastmuster zusätzlich zur der Anzahl der gleichzeitigen Zugriffe im Abstand zwischen dem Anzeigen und dem Abschicken des Formulars und beim Anteil der fehlerhaften Eingaben.

Variation eins dieses Benchmarks besteht in der Simulation von clientseitiger Vervollständigung und Validierung. Das Lastmuster umfasst dann zusätzlich unterschiedliche Annahmen über die Menge der Zugriffe die ein Nutzer beim Ausfüllen eines solchen Formulars generieren würde.

Bei Variation zwei generiert das Framework das Formular aus einem Anwendungs-Modell und erzeugt bei einer erfolgreichen Validierung entsprechende Objekte.

Weiterhin gibt es den sehr spannenden Anwendungsfall eines mehrseitigen Formulars, da er sowohl die Formularverarbeitung als auch die Sitzungsverwaltung eines Frameworks beansprucht.

Nutzersession

Erzeugt eine Cookie-basierte Session für den Nutzer, alle darauf folgenden Anfragen finden im Kontext dieser Nutzersession statt. Ein möglicher Anwendungsfall ist das eben erwähnte mehrseitige Formular.

Authentifizierung

Dieser Teilbenchmark betrachtet die An- und Abmeldung eines Nutzers an einer Anwendung. Er besteht aus mehreren Schritten. Zunächst wird ein Formular angezeigt, in dem der Client einen Benutzernamen und ein Passwort einträgt. Nach dem Abschicken des Formulars wird er auf einer Seite weitergeitet, die ihm die Anmeldung bestätigt oder eine Fehlermeldung anzeigt. Außerdem gibt es einen Link, der die Sitzung beendet und das erfolgreiche Abmelden bestätigt.

Das Lastmuster besteht in der Variation der Anzahl der gleichzeitig angemeldeten Nutzer und in der Länge der Sitzungen.

Eine Variation dieses Benchmarks ist eine zusätzliche Autorisierung. Nutzern sind unterschiedliche Rollen zugeordnet und bekommen auf einer Seite unterschiedliche Elemente zu sehen. Diese Elemente umfassen Links oder ganze Blöcke.

Datenbank

Dieser Teilbenchmark betrachtet sich mit Datenpersistenz und orientiert sich an den vier Standardoperationen auf relationalen Datenbanksystemen: Anlegen, Lesen, Aktualisieren und Löschen. Eine Implementierung umfasst das Anlegen von Objeten, das Aktualisieren von Objekten, das Auflisten von Objekten und das Löschen von Objekten. Ein Objekt enthält neben einer Nummer das es eindeutig identifiziert, eine Menge von Strings, deren Anzahl variiert wird.

Das Lastmuster variiert sowohl im Verhältnis der Lese- und Schreibzugriffe, der Menge der Objekte, sowie im Verhältnis der Operationen zu einander.

Nachrichten-Templates

Dieser Teilbenchmark besteht im Anzeigen einer Liste von Nachrichten mit jeweils unterschiedli-

chen Typen, die jeweils ein eigenes Template besitzen. Er ist an die Statusnachrichten von derzeit populären sogenannter sozialer Plattformen angelehnt.

Das Ziel dieses Teilbenchmarks ist die Geschwindigkeit der Template-Engine zu ermitteln. Das Lastmuster besteht in der Variation der gleichzeitig dargestellten Nachrichten, Nachrichtentypen sowie die Verschachtelung der Templates.

Globalisierung

Dieser Teilbenchmark besteht aus einer einzelnen Seite, die aus lokalisierten Texten und Datumsausgaben besteht. Die Anzahl der Elemente ist variabel. Das gewünschte Format wird als Parameter angegeben.

Das Lastmuster variiert in der Anzahl der verfügbaren Sprachen.

Eine Variation des Benchmarks benutzt den `Accept-Language`-Header der Anfrage zur Ermittlung des Darstellungsformates.

Nachrichten-API

Dieser Teilbenchmark ergänzt den bereits bestehenden Teilbenchmark Nachrichtenstrom um eine maschinenlesbaren Schnittstelle in Form einer RESTful API. Diese umfasst einfache Manipulationen wie das Anlegen, Ändern, Löschen und Auflisten von Nachrichten. Die Implementierungen sollen nach Möglichkeit sowohl XML, als auch JSON als Datenformat unterstützen.

Eine Variation dieses Teilbenchmarks ergänzt ihn um Authentifizierung mit Hilfe privater Schlüssel und signierter Request-Parameter.[76] Alle Zugriffe finden dann ausschließlich im Kontext des zugreifenden Nutzers statt.

4.3 Interaktion und Transparenz

Bei fast allen bestehenden Benchmarks lässt sich erkennen, dass es ein starkes Bedürfnis nach einer Ausweitung des Umfanges oder der Modifikation der Konfiguration gibt. Sehr häufig bleibt es allerdings bei Ankündigungen und die Beteiligung jenseits der Diskussion der Ergebnisse ist gering. Eine Ursache dafür könnte eine zu hohe Einstiegshürde sein, selbst eine kleine Änderung erfordert das vollständige Einrichten der Benchmark-Umgebung. Dazu kommt, dass bei den meisten Benchmarks die Konfiguration nicht vollständig und in einem strukturierten Format vorliegt, sondern lediglich durch die Angabe der genutzten Komponenten und deren Versionsnummern. Daher gibt es bei diesem Benchmark einige Maßnahmen, die die Mitarbeit externer Entwickler ermöglichen und

76 Reese, “Principles for Standardized REST Authentication.”

vereinfachen sollen.

Infrastruktur

Bei der Konzeption dieses Benchmarks gab es die Zielsetzung ihn auch auf einer weiteren Plattform durchführen zu können. Eine Möglichkeit dies umzusetzen ist die Nutzung vorkonfigurierter virtueller Maschinen, wie es auch einige Benchmarks tun. Ich habe mich für etwas anderes entschieden, das Konfigurationsmanagement-Werkzeug Puppet. Mit dem in Ruby geschriebenen Puppet lassen sich die Installation eines Systems, die Konfiguration von Diensten, das Deployment von Anwendungen, aber auch beliebige andere Aufgaben automatisieren. Die Spezifikation eines Systems mit Puppet erfolgt in einer eigenen, deklarativen Syntax, eine Datei die dieser geschrieben ist wird Manifest genannt. Die Stärke von Puppet liegt darin, für viele Systemkomponenten bereits vorgefertigte Module anzubieten, die Betriebssystem-spezifische Eigenschaften verbergen.[77] Die Entscheidung für Puppet geschah unter anderem, weil es die einfache Portierung auf ein anderes Betriebssystem versprach, sowie Unterstützung für das am Fachbereich genutzte Virtualisierungs-System vSphere von VMware anbot.[78]

Der Fachbereich Informatik hat für die Durchführung des Benchmarks drei virtuelle Server zur Verfügung gestellt. Auf diesen wurde wiederum eine Container-Virtualisierung eingerichtet. Dies geschah vor dem Hintergrund, mehrere unterschiedliche Konfigurationen betreiben zu können, ohne das diese sich gegenseitig beeinflussen. So ließe sich die Implementierung eines externen Entwicklers innerhalb eines eigenen Containers betreiben, ohne Wechselwirkungen oder Sicherheitsrisiken in Kauf nehmen zu müssen.[79] Container ermöglichen außerdem, experimentelle Änderungen separat zu testen und bei Bedarf auch wieder restlos entfernen zu können. Sowohl das Erzeugen, als auch das Einrichten eines neuen Containers mit LXC wurde für Puppet implementiert.

77 http://docs.puppetlabs.com/puppet/latest/reference/lang_summary.html; http://forge.puppetlabs.com/
78 http://forge.puppetlabs.com/puppetlabs/vcenter
79 Mit der Puppet-Konfiguration ist das nicht möglich, die Open-Source-Version von Puppet beinhaltet kein Berechtigungsmodell.

```
node 'frameworks1.imp.fu-berlin.de' {
  include fu-security
  # LXC
  include lxc-host
  container { 'benchmark2': }
}
node 'benchmark2.frameworks1.imp.fu-berlin.de' {
  include fu-security
  include r
  include benchmark
  Ab <<| |>>
  Siege <<| |>>
  R::Boxplot <<| |>>
}
```

Text 5: Puppet-Manifest mit einer virtuellen Maschine und einem Container (stark gekürzt)

Auf dem Server *frameworks1* befinden sich alle Komponenten, die für die Konfiguration oder die Durchführung des Benchmarks zuständig sind. Die Installationen der Frameworks und Teilbenchmarks in verschiedenen Konfigurationen geschieht auf dem Server *frameworks2*. Der dritte Server ist als Datenbank-Server vorgesehen, entsprechende Teilbenchmarks wurden aus Zeitmangel noch nicht implementiert.

Auch das Deployment von Teilbenchmarks sowie das Durchführen eines Benchmark-Laufes wurde mit Puppet realisiert. Auch wenn dies ein ungewöhnlicher Weg ist, habe ich mich aus zwei Gründen dafür entschieden. Die Einführung weiterer, spezialisierter Werkzeuge hätte zusätzlich Zeit gekostet und zu zusätzlicher Komplexität der Infrastruktur geführt. Die Vereinheitlichung bietet einen besseren Überblick über das Zusammenspiel der Konfiguration des Systems und der Anwendung. Auch die Konfiguration von Benchmarks, das Generieren von Statistiken und Auswertungen wird so automatisiert. In Text 6 ist die Konfiguration eines Containers mit Java 7, das Deployment einer Anwendung, ein Benchmark mit ApacheBench, das Generieren eines Graphen mit R und einer Zusammenfassung im HTML-Format zu sehen.

```
node 'java1.frameworks2.imp.fu-berlin.de' {
  include fu-security
  class { 'jdk':
    version => 7
  }
  vcsrepo {'/var/local/springbench':
    source => 'git@frameworks1.imp.fu-berlin.de:springbench.git'
  }
  @@ab { 'c00a78':
    url => 'http://java1.frameworks2.imp.fu-berlin.de:10002/table'
  }
  @@r::boxplot { 'table-spring-tapestry-c00a78-964805.png':
    datafiles => 'c00a78/ab-gnuplot.tsv 964805/ab-gnuplot.tsv'
  }
  @@page::result { 'spring-tapestry-c00a78-964805.html':
    image => 'table-spring-tapestry-c00a78-964805.png',
    description => 'Spring, Tapestry, ApacheBench, Tomcat, /table page'
  }
}
```

Text 6: Konfiguration einer Anwendung und Benchmark-Läufe (stark gekürzt)

Der Quellcode für die Teilbenchmarks und die Puppet-Konfiguration sind in git-Repositorys versioniert, so dass sich Änderungen an der Konfiguration, der Durchführung der Messungen und der Implementierungen der Teilbenchmarks nachvollziehen lassen.

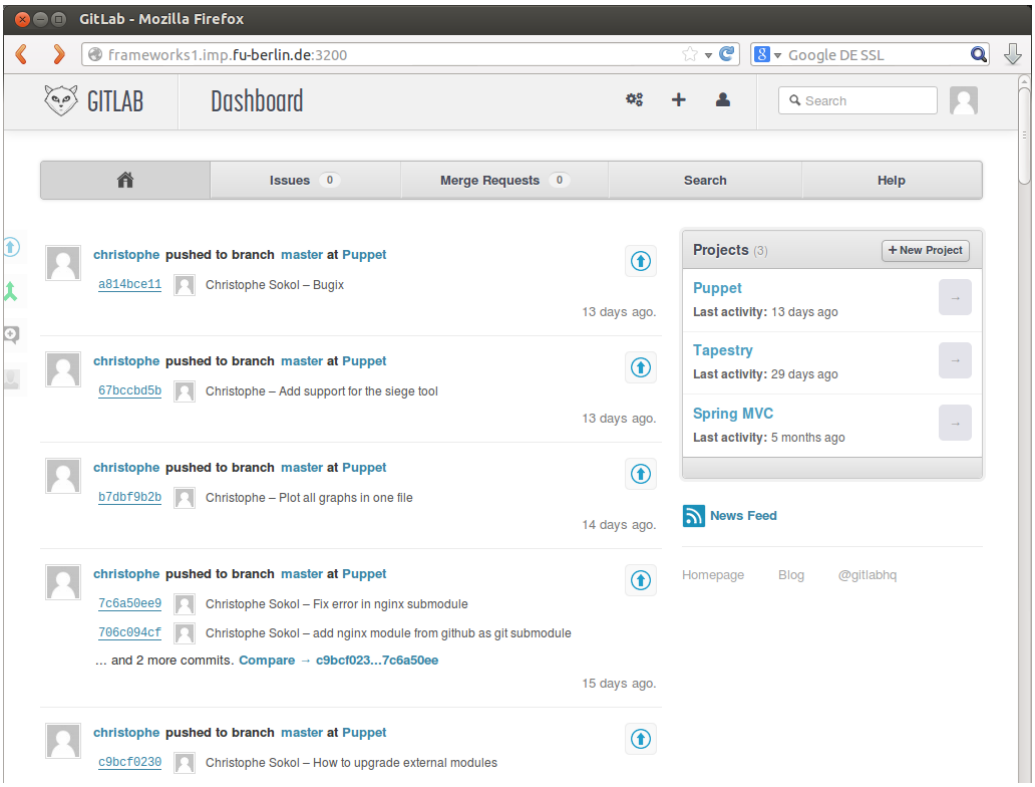

Abbildung 4: Gitlab mit einer Liste von Änderungen an der Plattform

Die Repositorys sind auch über eine Web-Oberfläche mit dem Namen Gitlab zugänglich. Diese ermöglicht nicht nur das Betrachten, der Dateien, sondern auch das direkte Ändern sowie Senden eines sog. Merge Request. Ein Entwickler kann in seiner Kopie des Repositorys Änderungen vorneh-

men und dann in Form des Merge Requests das Integrieren der Änderungen einfordern. Ebenfalls bietet es ein Wiki, dass für Dokumentation genutzt werden kann. Für eine bessere Sichtbarkeit des Projektes bietet es sich an, die Repositorys zukünftig auch auf Github zu veröffentlichen.

puppet doc

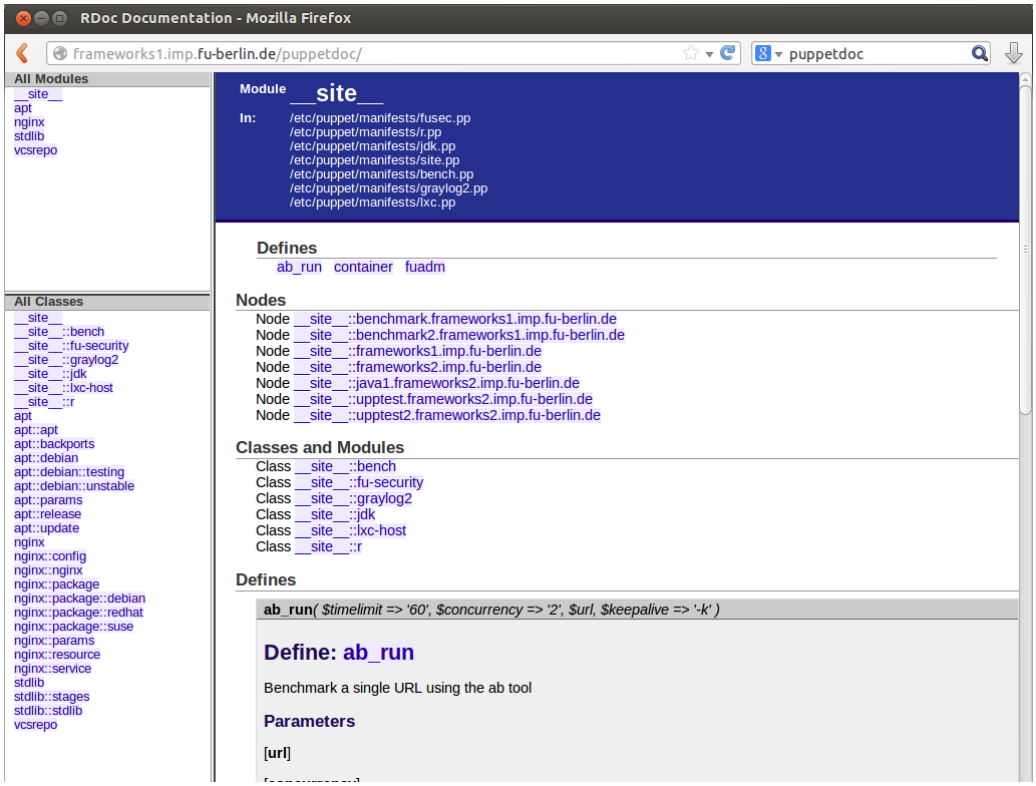

Abbildung 5: Puppet RDoc Documentation

Die in Puppet-Manifesten enthaltenen Kommentare lassen sich, vergleichbar mit der API-Dokumentation bei vielen Programmiersprachen, mit dem Befehl `puppet doc` in HTML konvertieren. Die resultierende Dokumentation umfasst alle Server und deren Konfiguration, die Schnittstellen der genutzten Puppet-Module, die eingerichteten Implementierungen, Benchmarks und Auswertungen. Sie liefert damit im Gegensatz zu einer reinen API-Dokumentation ein umfassendes Bild des Systems.

Puppet Node Manager

Der web-basierte Puppet Node Manager zeigt den Stand aller mit Puppet konfigurierten Systeme aus der Sicht des Puppet-Daemons. Abbildung 6 zeigt die Startseite des Werkzeugs. Im unteren Teil sind die fünf konfigurierten Systeme aufgeführt. Zwei davon sind die vom Fachbereich zu Verfügung gestellten virtuellen Server, drei weitere sind die darauf befindlichen Container.

Die Balken im oberen Teil zeigen den Gesamtstatus des Systems der vergangenen 30 Tage dar, auf

dem Screenshot werden sie nicht vollständig dargestellt. Die Höhe der Balken gibt die Anzahl der Aktualisierungen von Systemen an. Rot gibt an dass ein Fehler aufgetreten ist, Blau steht für eine Aktualisierung und bei Grün gab es keine Änderung. Änderungen sind beispielsweise das Hinzufügen einer Datei, aus Ausführen eines Befehls oder das Aktualisieren eines Git-Repositorys.

Auf dem Screenshot lässt sich erkennen, dass am 08.04.2013 Fehler korrigiert wurden, sowie ein weiteres System hinzugefügt wurde. Dadurch ist der rote Anteil in den Balken verschwunden und die Gesamtzahl der Aktualisierungen hat sich erhöht. Zur Diagnose von Fehlern lässt sich der Status eines jeden Systems ablesen, bei einer größerem Anzahl von konfigurierten Systemen lassen sie sich mit dem Menü auf der linken Seite nach Status filtern.

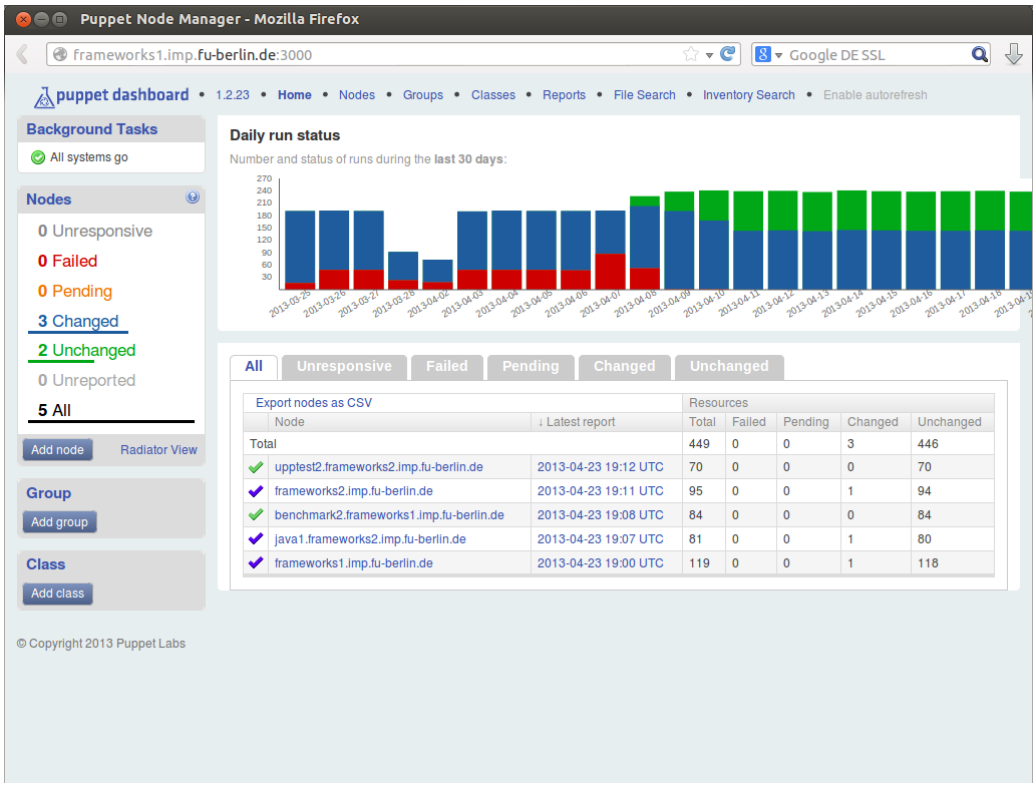

Illustration 6: Der Puppet Node Manager dient v. a. dem Monitoring

Durch Klicken auf den Hostnamen eines Systems gelangt man auf eine Detailansicht, die den derzeitigen Zustand aller mit Puppet konfigurierten Ressourcen angibt.

Graylog2

Graylog2 ist ein Dienst zur zentralen Verwaltung von System-Logs und kann diese über ein Web-Interface (s. Abb. 3) anzeigen, filtern und durchsuchbar machen. Er ist kein direkter Bestandteil der Benchmark-Infrastruktur, bietet sich aber an, um bei einer größeren Zahl an Systemen nicht den Überblick zu verlieren.

```
frameworks1 puppet-master[3199]: - puppet-master: Compiled catalog for
benchmark2.frameworks1.imp.fu-berlin.de in environment production in 0.23
seconds
benchmark2 puppet-agent[26575]: - puppet-agent: Finished catalog run in
0.92 seconds
```

Text 7: Ausgabe von Puppet-Client und -Server bei der Aktualisierung von "benchmark2"

5 Ausblick

Angesichts des Standes der Umsetzung fällt es nicht schwer Verbesserungsmöglichkeiten zu benennen. Neben technischen und konzeptionellen Arbeiten fehlen aber vor allem Menschen. Nur in Zusammenarbeit mit Web-Entwicklern und den Entwicklern der Frameworks lässt sich die Arbeit der Implementierung mit verschiedenen Frameworks bewerkstelligen und ohne eine offene Diskussion über die Repräsentativität von Teilbenchmarks werden die Ergebnisse niemanden überzeugen. Wie bei den bestehenden Benchmarks zu sehen ist, erhält die Veröffentlichung eines Benchmarks vergleichsweise einfach eine größere Aufmerksamkeit und es besteht die Bereitschaft Verbesserungen beizusteuern. Wenn diese Änderungen nicht unmittelbar berücksichtigt werden, beispielsweise in Form einer aktualisierten Auswertung, endet diese abrupt wieder. Es ist also von zentraler Bedeutung bei einer Veröffentlichung der Plattform sicherzustellen, dass die Hürden für Partizipation möglichst niedrig sind und jeder Beitrag automatisch in die Auswertungen übernommen wird.

Eine Möglichkeit die Leistungsfähigkeit der Plattform zu demonstrieren ist die Portierung der bei bestehenden Benchmarks genutzten Implementierungen. Ein Vergleich der Ergebnisse kann zusätzlich genutzt werden die korrekte Konfiguration der Plattform sicherzustellen. In diesem Zusammenhang ist eine Portierung zu einer bei bestehenden Benchmarks genutzten IaaS-Plattform wie Amazon EC2 sinnvoll.

Zusätzlich zu den derzeit betrachteten Frameworks Tapestry und Spring MVC bietet sich vor allem JSF und die Betrachtung der vielen verfügbaren Implementierungen an. Mit Untersützung der jeweiligen Communities ist auch eine Beteiligung anderer Plattformen wie Ruby oder PHP wünschenswert.

Als Werkzeuge zum Generieren von Last sollte das im Java-Umfeld populäre JMeter sowie das relativ junge und sehr leistungsfähige wrk[80] integriert werden. Die derzeit genutzten Werkzeuge validieren die von den Servern und Frameworks gesendeten Antworten nicht. Ein Teil der bestehenden Benchmarks hat damit Fehler entdecken können, die das Ergebnis sonst verzerren würden.

80 https://github.com/wg/wrk

Mit der hier präsentierten Integration von R und modernen Web-Technologien sind deutlich aufwendigere Auswertungen möglich. Außerdem gibt es noch viele Daten, die sich während der Durchführung eines Benchmarks erheben lassen, sowohl in der Laufzeitumgebung, als auch beim Betriebssystem.

6 Schluss

Das initiale Konzept dieser Arbeit sah vor, die Konzeption und Implementierung der Infrastruktur und der Teilbenchmarks in Form eines iterativen Vorgehens durchzuführen. Das Ziel war die frühzeitige Einbindung von Entwicklern der Frameworks zu ermöglichen und eine schwierige Kalkulation von Implementierungsaufwand zu vermeiden.

Dies ist nur in Teilen gelungen. Die Umsetzung der Infrastruktur, Implementierungen mit zwei Frameworks, die Herleitung von Teilbenchmarks und eine automatisierte Auswertung waren aufwändiger als angenommen. Der Mangel an interessanten und aufbereiteten Messergebnissen hat die Kontaktaufnahme zu Frameworks immer wieder verschoben. Die Frage nach charakteristischen Bestandteilen von Web-Anwendungen und -Frameworks war mit einem iterativen Vorgehen schwer zu beantworten.

Umgekehrt hat das iterative Vorgehen auch ermöglicht, die Aufgabe eines Framework-Benchmarks gleichzeitig von einer technischen, einer konzeptuellen und einer theoretischen Seite aus anzugehen.

7 Glossar

AJAX
: Leitet sich ab von „Asynchronous JavaScript and XML“ und bezeichnet die Technik bei der der Browser asynchron Daten vom Server lädt und auf einer bestehenden Webseite anzeigt.[81]

API
: Eine Web-API ist eine HTTP-Schnittstelle, mit der eine Web-Anwendung Daten anderen Anwendungen zur Verfügung stellt.

CDN
: Ein Content Delivery Network bezeichnet ein Netz meist geographisch verteilter Server, um Inhalte schnell an Nutzer übertragen zu können.

CI
: Continuous Integration ist eine Vorgehensweise der Softwaretechnik die verlangt die Bestandteile der Anwendung während der Entwicklung regelmäßig zusammenzuführen und das Resultat automatisiert zu testen.

81 Garrett, “Ajax: A New Approach to Web Applications.”

CRUD
: Leitet sich von den Grundoperationen Create, Read, Update und Delete relationaler Datenbanken ab, bezeichnet aber auch entsprechende HTTP-Methoden oder Bestandteile von Benutzeroberflächen.

CSS[82]
: Bezeichnet die Technologie mit der Web-Dokumenten grafische Elemente hinzugefügt werden.

CSRF
: Bezeichnet eine Gruppe von Sicherheitslücken, die auf unzureichend gesicherte Formulare o. ä. zugreifen, indem sie den Browser eines ordnungsgemäß autorisierten Nutzers dazu bringen auf diese zuzugreifen.

DOM[83]
: Das Document Object Model ist eine Schnittstelle für den Zugriff und die Manipulation von Dokumenten. Sie wird genutzt, um mit JavaScipt innerhalb eines Browsers auf HTML-und XML-Dokumente zuzugreifen.

HTML
: Ist die Beschreibungssprache für Dokumente im Web, die Bezeichnungen HTML4 und HTML5 verweisen auf die jeweilige Version des vom W3C verabschiedeten Standards. Die Abkürzung HTML wird sowohl als Überbegriff, als auch für die Weiterentwicklung von HTML5 durch die WHATWG unter dem Namen „HTML Living Standard“ verwendet.[84]

HTTP
: Das „Hypertext Transfer Protocol“ ist das im Web vorherrschende Protokoll zum Übertragen von Daten zwischen Client und Server. Als HTTPS erfolgt die Übertragung mit SSL oder TLS verschlüsselt und authentifiziert. Google setzt seinen Vorschlag SPDY als HTTP-Nachfolger bereits in den eigenen Produkten ein. Weiterhin steigt die Browserunterstützung für WebSockets sowie WebRTC.

I18n
: Numeronym des Begriffes „internationalization“, der bezeichnet ein Produkt so zu gestalten, dass es an Nutzer aus verschiedenen Sprachen, Ländern und Kulturen angepasst werden kann. Dies geschieht bei Webanwendungen durch das Unterstützen von Übersetzungen, unterschiedlichen Zeit- und Währungsangaben sowie -darstellungen, u. a.

L10n
: Ist die Kurzform des Begriffes „localization“ und bezeichnet den auf die Internationalisierung aufbauenden Prozess, die Anwendung an eine konkrete Nutzergruppe anzupassen, z. B. durch Einpflegen einer Übersetzung.

IaaS
: Bezeichnet das Geschäftsmodell, Hardware-Ressourcen, insbesondere Server oder Netzwerkkomponenten auf Mietbasis anzubieten. Charakteristisch für IaaS-Angebote sind ein fein granulierte Abrechnungsmodell, kurze Mietdauern sowie ein hoher Abstraktionsgrad bei der Spezifikation der gemieteten Komponenten.

82 http://www.w3.org/Style/CSS/
83 http://www.w3.org/DOM/
84 Hickson to whatwg@whatwg.org, “[whatwg] Administrivia: Update on the Relationship Between the WHATWG HTML Living Standard and the W3C HTML5 Specification.”

J2EE
: Die Java Platform, Enterprise Edition ist eine auf der Sprache Java basierenden Plattform für web-basierte und andere Anwendungen. Sie umfasst beispielsweise die Servlet API, die Persistence API, JavaServer Pages. Anwendungsserver wie Glassfish oder Tomcat implementieren die APIs je nach Anwendungsfall vollständig oder teilweise.

JS
: Ist die Abkürzung von JavaScript, als Ergänzung zu HTML entstanden und unter dem Namen ECMA-Script standardisiert.

JSF
: JavaServer Faces ist ein Standard für Web-Frameworks, der primär Benutzeroberflächen abdeckt. Die Referenzimplementierung heißt Mojarra, Apache MyFaces ist ebenfalls sehr populär.

JSP
: JavaServer Pages ist eine Template-Sprache zu Generierung von HTML- und XML-Ausgaben. Die Syntax lässt sich durch sog. Tag-Libraries erweitern. JSP gilt als veraltet.

JSTL
: Die Java Standard Tag Library ist eine Funktionsbibliothek und Syntaxerweiterung für JSP.

LXC
: Linux Containers ist ein Virtualisierungssystem auf Betriebssystem-Ebene, vergleichbar mit FreeBSD jails oder OpenVZ. Die Container nutzen den Kernel des Host-Systems sind aber ansonsten vollständig voneinander getrennt.

MVC
: Model-View-Controller ist ein Entwurfsmuster für Anwendungen mit graphischen Oberflächen. Es besteht in der Aufteilung der Anwendung die drei namensgebenden Bestandteile.

ORM
: Object-Relational-Mapping bezeichnet die Umwandlung, die nötig ist um Objekte in relationalen Datenbanken zu speichern.

REST
: Ist eine Software-Architektur für verteilte Anwendungen und wird häufig bei der Entwicklung von maschinenlesbaren Schnittstellen im Web eingesetzt.

SOAP
: Das Simple Object Access Protocol ist ein auf XML basierender Standard zum Austausch von Nachrichten und Daten. SOAP wurde vom weniger komplexen REST-Stil verdrängt.

URI
: Uniform Resource Identifiers sind standardisierte Bezeichner mit einer festgelegten Syntax.

URL
: Uniform Resource Locators sind eine Form von URIs und identifizieren vor allem Webseiten, aber auch Dateien im Dateisystem.

W3C
: Das World Wide Web Consortium das ein Standardisierungsgremium für Web-Technologien, beispielsweise HTML und CSS.

WHATWG

Die Web Hypertext Application Technology Working Group ist eine mit der W3C konkurrierende Arbeitsgruppe, die von Browserherstellern gegründet wurde, um den Standardisierungsprozess zu beschleunigen.

8 Verweise

Amazon Web Services, Inc. “Amazon EC2-Instancetypen,” 2012. http://aws.amazon.com/de/ec2/instance-types/.

Aster, Alex. “The Performance Test of 6 Leading Frameworks.” *Alrond’s Technoblog*, February 4, 2007. http://www.alrond.com/en/2007/jan/25/performance-test-of-6-leading-frameworks/.

Bergsma, Mark. “Wikimedia Architecture.” 2007. http://www.nedworks.org/~mark/presentations/san/Wikimedia%20architecture.pdf.

Bergsten, Hans. “Improving JSF by Dumping JSP.” *O’Reilly ONJava*, June 9, 2004. http://www.onjava.com/pub/a/onjava/2004/06/09/jsf.html.

Buchner, Björn, Axel Bottcher, and Christian Storch. “Evaluation of Java-based Open Source Web Frameworks with Ajax Support.” In *2012 14th IEEE International Symposium on Web Systems Evolution (WSE)*, 45–49. Trento, Italy, 2012. doi:10.1109/WSE.2012.6320531.

Burns, Edward, and Roger Kitain. “JSR 314: JavaServer Faces 2.1,” November 2010. http://www.jcp.org/en/jsr/detail?id=314.

Cohen, Reuven. “Gartner Announces 2012 Magic Quadrant for Cloud Infrastructure as a Service.” *Forbes*, October 22, 2012. http://www.forbes.com/sites/reuvencohen/2012/10/22/gartner-announces-2012-magic-quadrant-for-cloud-infrastructure-as-a-service/.

Davis, Seth. “The Great Web Framework Shootout.” Blog. *Curia: Technology Thoughts and Ideas*, 2011. http://blog.curiasolutions.com/the-great-web-framework-shootout/.

Donald, Keith. “Spring MVC 3 Showcase.” *SpringSource Team Blog*, July 22, 2010. http://blog.springsource.com/2010/07/22/spring-mvc-3-showcase/.

Eichinger, Christian. “Web Application Architectures.” In *Web Engineering: The Discipline of Systematic Development of Web Applications*, edited by Gerti Kappel, Birgit Proll, Siegfried Reich, and Werner Retschitzegger. Chichester: John Wiley & Sons, 2006.

Feitelson, Dror G. “Metric and Workload Effects on Computer Systems Evaluation.” *Computer* 36, no. 9 (September 2003): 18–25.

Fielding, Roy Thomas. “Architectural Styles and the Design of Network-based Software Architectures.” Doctoral dissertation, University of California, 2000.

Garrett, Jesse James. “Ajax: A New Approach to Web Applications.” *Adaptive Path*, February 18, 2005. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications.

Gousios, Georgios, Vassilios Karakoidas, and Diomidis Spinellis. “Tuning Java’s Memory Manager for High Performance Server Applications.” In *Proceedings of the 5th International System Administration and Network Engineering Conference SANE 06*, 69–83. edited by Alexios Zavras. NLUUG, Stichting SANE, 2006. http://www.dmst.aueb.gr/dds/pubs/conf/2006-05-SANE-JTune/html/JTune.pdf.

Hashemian, Raoufehsadat, Diwakar Krishnamurthy, and Martin Arlitt. “Web Workload Generation Challenges – an Empirical Investigation.” *Software: Practice and Experience* 42, no. 5 (2012): 629–647. doi:10.1002/spe.1093.

Hickson, Ian. Letter to whatwg@whatwg.org. “[whatwg] Administrivia: Update on the Relationship Between the WHATWG HTML Living Standard and the W3C HTML5 Specification,” July 19, 2012. http://lists.w3.org/Archives/Public/public-whatwg-archive/2012Jul/0119.html.

Hoeller, Juergen. “Next Stop: Spring Framework 4.0.” *SpringSource Team Blog*, January 16, 2013. http://blog.springsource.org/2013/01/16/next-stop-spring-framework-4-0/.

JT Dev. “Rails, Wicket, Grails, Play, Tapestry, Lift, JSP, Context.” Blog. *About Programming Stuff*, May 28,

2011. http://www.jtict.com/blog/rails-wicket-grails-play-lift-jsp/.
Larabel, Michael. “Extensive Benchmarks Of Amazon’s EC2 Compute Cloud Performance.” *Phoronix*, January 12, 2011. http://www.phoronix.com/scan.php?page=article&item=amazon_ec2_exhaustive.
Lev Walkin. “Stress-testing Httperf, Siege, Apache Benchmark, and Pronk.” *Lionet*, January 4, 2012. http://lionet.livejournal.com/99984.html.
Mason, Ross. “How REST Replaced SOAP on the Web: What It Means to You.” *InfoQ*, October 20, 2011. http://www.infoq.com/articles/rest-soap.
McCabe, James. “Webofcontacts: A Comparison of Web Application Frameworks.” Google Project Hosting, 2009. http://code.google.com/p/webofcontacts/.
Mordani, Rajiv, and Amy Roh. “JSR 315: Java Servlet 3.0 Specification,” 2009. http://download.oracle.com/otn-pub/jcp/servlet-3.0-mrel-eval-oth-JSpec/servlet-3_0-mrel-spec.pdf.
Nagpurkar, P., W. Horn, U. Gopalakrishnan, N. Dubey, J. Jann, and P. Pattnaik. “Workload Characterization of Selected JEE-based Web 2.0 Applications.” In *IEEE International Symposium on Workload Characterization, 2008. IISWC 2008*, 109–118, 2008. doi:10.1109/IISWC.2008.4636096.
Pereira, A., L. Silva, W. Meira, and W. Santos. “Assessing the Impact of Reactive Workloads on the Performance of Web Applications.” In *2006 IEEE International Symposium on Performance Analysis of Systems and Software*, 211–220, 2006. doi:10.1109/ISPASS.2006.1620805.
Perera, Srinath. “List of Known Scalable Architecture Templates.” *Srinath’s Blog: My Views of the World*, October 30, 2011. http://srinathsview.blogspot.com/2011/10/list-of-known-scalable-architecture.html.
Punz, Werner. “Mojarra Vs. MyFaces Performance.” *Werners Blog*, October 14, 2011. http://werpublogs.blogspot.de/2011/10/mojarra-vs-myfaces-performance.html.
Raible, Matt. “Comparing JVM Web Frameworks.” presented at the Spring I/O, Madrid, Spain, February 2012. http://de.slideshare.net/mraible/comparing-jvm-web-frameworks-spring-io-2012.
Reese, George. “Principles for Standardized REST Authentication.” Blog. *O’Reilly Broadcast*, December 26, 2009. http://broadcast.oreilly.com/2009/12/principles-for-standardized-rest-authentication.html.
Robinson, David, and Ken A. L. Coar. “RFC 3875: The Common Gateway Interface (CGI) Version 1.1,” October 2004. http://www.ietf.org/rfc/rfc3875.
Rui, Wang. “PHP Framework MVC Benchmark - V20111201-4.” Blog. *Ruilog*, December 1, 2011. http://www.ruilog.com/blog/view/b6f0e42cf705.html.
Sangeetha, S., K. L. Nitin, and S. Ananya. “JSF 2.0 Views: Hello Facelets, Goodbye JSP.” *Developer.com*, March 1, 2010. http://www.developer.com/java/web/article.php/3867851/.
Sankar, Rakesh. “PHP-Framework-Benchmark,” 2011. http://rakesh-sankar.github.com/PHP-Framework-Benchmark/.
Schwan, Ben. “Formatkrieg Beim Videochat: WebRTC Gegen CU-RTC.” *Heise Online*, June 3, 2013. http://heise.de/-1805581.
Sobel, Will, Shanti Subramanyam, Akara Sucharitakul, Jimmy Nguyen, Hubert Wong, Arthur Klepchukov, Sheetal Patil, O Fox, and David Patterson. *Cloudstone: Multi-platform, Multi-language Benchmark and Measurement Tools for Web 2.0*, 2008. http://cca08.org/papers/Paper33-Armando-Fox.pdf.
Sohn, Dirk M., and Papick Taboada. *OIO Kompass: Java Web-Frameworks*. Orientation in Objects GmbH, April 2012. http://www.oio.de/public/java/java-web-frameworks-vergleich/jsf-vs-gwt-studie.htm.
Taylor, Colleen. “Codecademy Partners With Twitter, Evernote, Box, And Others To Offer A Suite Of New API Lessons.” *TechCrunch*, February 19, 2013. http://techcrunch.com/2013/02/19/codecademy-partners-with-twitter-evernote-box-and-others-to-offer-a-suite-of-new-api-lessons/.
TechEmpower, Inc. “Framework Benchmarks.” *TechEmpower Blog*, March 28, 2013. http://www.techempower.com/blog/2013/03/28/framework-benchmarks/.
Thomas, Peter. “‘Perfbench’ Update: Tapestry 5 and Grails.” Blog. *Incremental Operations*, September 14, 2009. http://ptrthomas.wordpress.com/2009/09/14/perfbench-update-tapestry-5-and-grails/.
———. “Seam / JSF Vs Wicket: Performance Comparison.” Blog. *Incremental Operations*, January 14, 2009. http://ptrthomas.wordpress.com/2009/01/14/seam-jsf-vs-wicket-performance-comparison/.

Uribe, Leonardo. *Understanding JSF 2 and Wicket: Performance Comparison*, May 18, 2012. https://raw.github.com/lu4242/performance-comparison-java-web-frameworks/master/comparison-2012-APR/Performance%20Comparison-Leonardo%20Uribe-FINAL-Detailed.pdf.

Van den Enden, Stijn. "WWW: World Wide Wait. A Performance Comparison of Java Web Frameworks." presented at the Devoxx, Antwerp, Belgium, November 17, 2011. http://prezi.com/dr3on1qcajzw/www-world-wide-wait-devoxx-edition/.

Weier O'Phinney, Matthew. "On Microframeworks." Blog. *Phly, Boy, Phly*, August 17, 2012. http://www.mwop.net/blog/2012-08-17-on-microframeworks.html.

Wickesser, Craig. "Google Chose Jetty for App Engine." *InfoQ*, August 5, 2009. http://www.infoq.com/news/2009/08/google-chose-jetty.

Winterfeldt, David. "Spring Web Flow Subflow Webapp: JSP Example." *Spring by Example*. Accessed April 28, 2013. http://www.springbyexample.org/examples/spring-web-flow-subflow-webapp-jsp-example.html.

ZeroTurnaround. *Java EE Productivity Report 2011*, January 2011. http://zeroturnaround.com/java-ee-productivity-report-2011/.

Zinke, J., J. Habenschuss, and B. Schnor. "Servload: Generating Representative Workloads for Web Server Benchmarking." In *2012 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, 1–8, 2012.