HTW Dresden
Fachbereich Informatik

Bachelorarbeit

Leistungsuntersuchung von Webserveranwendungen und Optimierung auf Basis von Drupal 5 (am Beispiel von ITsax.de)

Bachelorarbeit

Autor:	Schneider, Martin
Seminargruppe	08/041/02
Betreuender Professor:	Prof. Vogt
Datum:	6. September 2011

Inhaltsverzeichnis

I. Einleitung 8

1. Motivation 8

1.1. pludoni GmbH 8

1.2. Ziel 9

2. Aufbau 9

II. Theoretische Grundlagen 10

3. Technische Voraussetzungen 10

4. Methoden der Web-Performance-Optimierung 11

5. Einordnung von Performanzproblemen 13

5.1. Clientseitige Probleme 13

5.1.1. Netzwerkverbindung 13

5.1.2. Browser 16

5.2. Serverseitige Probleme 18

5.2.1. Netzwerkverbindung 18

5.2.2. Datenbankabfragen 18

6. Wirtschaftliche Aspekte 19

6.1. Usability 19

6.2. Google Ranking 19

6.3. Serverlast 20

III. Praxisteil 22

7. Versuchsaufbau 22

8. Testverfahren 23

8.1. Servertest 23

8.2. Browsertest 23

8.3. Andere Analyseverfahren 24

9. Ausgangszustand **25**

10. Implementierung und Test der einzelnen Methoden **29**
10.1. Alternative PHP Cache 29
10.2. Drupal Cache 31
10.3. JS Aggregation und Minifizierung mit dem Javascript Aggregation Modul 32
10.4. CSS Aggregation und Komprimierung 32
10.5. Drupal Boost Modul 33
10.6. Bildkompression mit jpegoptim und OptiPNG 34
10.6.1. Verlustfreie Kompression 34
10.6.2. Verlustbehaftete Kompression 34
10.7. Drupal 5 Fehler bei umgefärbten Themes 35
10.8. Theme Bilder spriten 36
10.9. Module von Startseite entfernen 37
10.9.1. Partnerslideshow 37
10.9.2. Facebook Widget 38
10.9.3. Jobleiste deaktiviert 38
10.10Umprogrammierung der Module 38
10.10.1Partnerslideshow 39
10.10.2Facebook Widget 41
10.10.3Partneranzeigen 42

11. Endergebnis **45**

12. Vergleich **46**

13. Zusammenfassung und Ausblick **46**

Danksagungen

Der Autor möchte seinen Dank der Fakultät Informatik und Prof. Wiedemann für das zur Verfügung gestellte Büro, aussprechen.

Ich danke auch meiner Freundin, die mich während des Schreibens unterstützt hat und viele orthografische Optimierungen vornahm.

Diese Arbeit entstand in Zusammenarbeit mit der Firma pludoni GmbH und unter Aufsicht von Jörg Klukas. Ich danke ihm für die Möglichkeit mich mit diesem Themengebiet zu befassen und für das Vertrauen, dass er in mich gesteckt hat.

Besonderen Dank gilt meinem Betreuer, Prof. Vogt, der mein Thema unterstützte und sich mit meiner Arbeit beschäftigte und wertvolle Hinweise erteilte.

Abkürzungsverzeichnis

Ajax	Asynchronous JavaScript and XML
HTML	Hypertext Markup Language
PHP	Asynchronous JavaScript and XML
IMAP	Internet Message Access Protocol
CMS	Content-Management-System
CPU	Central Processing Unit
RAID	Central Processing Unit
SSD	Central Processing Unit
DOM	Document Object Model
DDoS	Distributed Denial of Service
MIME	Multipurpose Internet Mail Extensions
TtFB	Time to First Byte
TtsR	Time to start Render
LT	Load Time
APC	Alternative PHP Cache
PNG	Portable Network Graphics
URI	Uniform Resource Identifier
CDN	Content Distribution Network
MyISAM	My Indexed Sequential Access Method
InnoDB	Innobase Database

Glossar

Ajax	ist ein Konzept zur asynchronen Datenübetragung.
Community	Zusammenschluss vieler Teilnehmer, um ein gemeinsames Ziel zu erreichen.

CMS Ein Content-Management-System ermöglicht auch Personen ohne Programmierkenntnisse die Administration einer Webseite.

gzip Ein Programm für Komprimierungsaufgaben.

Apache Webserver Apache ist seit 1996 der populärste Web Server auf dem Markt. Eine sehr große Open-Source-Gemeinde sorgt für Updates und Zusatzmodule.

DOM Eine Schnittstelle für den Zugriff auf ein Dokument und dessen Änderung. `http://www.w3.org/DOM/#what`

MySQL ist ein relationales Datenbankverwaltungssystem.

Teil I.
Einleitung

> "Human beings don't like to wait. We don't like waiting in line at a store, we don't like waiting for our food at a restaurant, and we definitely don't like waiting for Web pages to load."
>
> Alberto Savoia

Jeder Internetnutzer hat eine gewisse Toleranz gegenüber langsamen Webseiten. Wenn er jedoch auf eine Webseite zu lange warten muss, geht er zur Konkurrenz[website optimization, 2008]. Um dies zu verhindern, hat das Thema Web-Performance in der letzten Zeit immer mehr an Bedeutung gewonnen. Aber was bedeutet Web-Performance eigentlich? Zum einen gibt es die subjektive Performance, die ein Besucher beim Browsen einer Webseite empfindet. Gibt es für den Nutzer bemerkbare Wartezeiten oder reagiert die Webseite schneller als die Wahrnehmungsschwelle? Solche Kriterien sind sehr wichtig bei interaktiven Webseiten und haben auch zum Erfolg von Ajax beigetragen. Zum anderen kann man auch eine objektive Aussage über die Geschwindigkeit, mit der eine Webseite generiert, ausgeliefert und beim Besucher im Browser angezeigt wird, treffen.

1. Motivation

Der Autor dieser Arbeit absolvierte sein Pflichtpraktikum in der pludoni GmbH und war bis zum heutigen Tag als Werksstudent tätig. Bei dieser Tätigkeit kamen immer öfter Performance Probleme zum Tragen. Um diesen zu begegnen, entstand diese Arbeit.

1.1. pludoni GmbH

"pludoni kommt aus dem Esperanto und bedeutet weitergeben." [Klukas, 2011] Das Unternehmen wurde 2008 von Dr. Jörg Klukas gegründet und hat als Ziel, den regionalen Mittelstand bei der Kommunikation und Bewerbersuche durch Communities zu

unterstützen. Die von pludoni gemanagten Communities heißen ITsax.de, ITmitte.de und MINTsax.de. Sie richten sich an Technologiefirmen aus Sachsen beziehungsweise Mitteldeutschland und versuchen über zahlreiche Dienstleistungen einen Mehrwert für die beteiligten Unternehmen zu schaffen. Dazu gehören im Allgemeinen:

- Aggregation, Veröffentlichung und Weitergabe von Stellenanzeigen,
- Organisation von regelmäßigen Community-Treffen zum Austausch der Unternehmen,
- Unterstützung und Beratung bei der Gestaltung Suchmaschinen-optimierter Inhalte und
- Infrastruktur zur gegenseitigen Empfehlung von Fachkräften und Lernenden.

1.2. Ziel

Alle Communities sind zum heutigen Zeitpunkt durch das Content-Management-System Drupal 5 realisiert. Drupal 5 ist mittlerweile schon vier Jahre alt und nicht mehr auf dem neuesten Entwicklungsstand der Web-Performance. Es gibt Drupal mittlerweile schon in der Version 7.7 und viele Verbesserungen in Sachen Geschwindigkeit wurden seither realisiert. Leider ist es nicht trivial ein Upgrade durchzuführen, da viele Strukturen sich stark verändert haben. Aus diesem Grund wurde der Autor damit beauftragt, die Möglichkeiten einer Drupal 5 Optimierung am Beispiel der ältesten Community ITsax.de zu ergründen und, wenn vorhanden, durchzuführen. Dabei soll ein Leitfaden entwickelt werden, anhand dessen Drupal 5 und allgemein Webseiten effizient optimiert werden können.

2. Aufbau

Die Arbeit ist in zwei Teile aufgeteilt. Der Theorieteil beschäftigt sich mit den Grundlagen der Web-Performance beziehungsweise der Web-Performance-Optimierung. Im Anschluss daran werden im Praxisteil Performance-Mängel der Webseite ITsax.de untersucht und, sofern vorhanden, behoben.

Teil II.

Theoretische Grundlagen

3. Technische Voraussetzungen

Im Rahmen dieser Bachelorarbeit werden verschiedenste Technologien verwendet und auf ihnen aufbauende Werkzeuge zu Hilfe genommen. Um zu verstehen, wo Probleme lokalisiert sind und wie solche Schwachstellen zu finden sind, muss man sich mit dem vorhandenem Technologiestack auseinandersetzen und ihn analysieren.

Server Ein Server ist ein Computer, der Informationen und Dienste für andere Computer zur Verfügung stellt.

Betriebssystem Die Software, die auf dem Server läuft. In der pludoni GmbH wird das Linux-Derivat Debian eingesetzt.

Datenbank Als Datenbank wird eine strukturierte Sammlung von Daten bezeichnet. Einer der häufigsten Vertreter, gerade im Zusammenhang mit PHP-Webanwendungen, ist MySQL.

Web Server Der Web Server ist für die zuverlässige Auslieferung von Webseiten zuständig. Er beantwortet die Anfragen, die die Nutzer mit Hilfe des Browsers stellen. In der pludoni GmbH wird für diesen Zweck Apache eingesetzt.

PHP ist eine dynamische, typisierte Skriptsprache. Sie ist speziell für die Webprogrammierung geeignet, da sie direkt in HTML eingebettet werden kann.[PHP, 2011a] In PHP kann prozedural und objektorientiert programmiert werden. Besonders gut unterstützt werden alle Arten von Datenbanken. Durch die große Entwicklergemeinde gibt es viele Erweiterungen für PHP. So kann Dank PHP auch sehr leicht mit anderen Services kommuniziert werden, beispielsweise IMAP für die E-Mail-Server Kommunikation.[PHP, 2011b]

Drupal ist ein CMS und Framework, welches in PHP geschrieben ist und viele direkt nutzbare Funktionen mitbringt. Dazu gehören unter anderem Administrationsoberflächen, News-Aggregatoren, Veröffentlichungsabläufe für Artikel und Blogbeiträge sowie Suchmaschinenoptimierungen. Außerdem ermöglicht Drupal die Installation weiterer, durch Nutzer entwickelte, Module. Dadurch wird eine fast

unbegrenzte Funktionsvielfalt ermöglicht. Im vorliegenden Fall wird Drupal in der Version 5 eingesetzt.

Client Der Browser des Nutzers erwartet in der Regel ein HTML Dokument. Dies wird durch Drupal generiert und vom Webserver ausgeliefert. Dabei werden Bilder verarbeitet, JavaScript Skripte ausgeführt und andere Elemente, wie Flash-Applikationen und Videos, berechnet.

In Abbildung 1 ist eine schematische Darstellung eines Webseitenaufrufs skizziert. Die orangenen Pfeile stellen eine Anfrage da und die grünen Pfeile die Antworten.

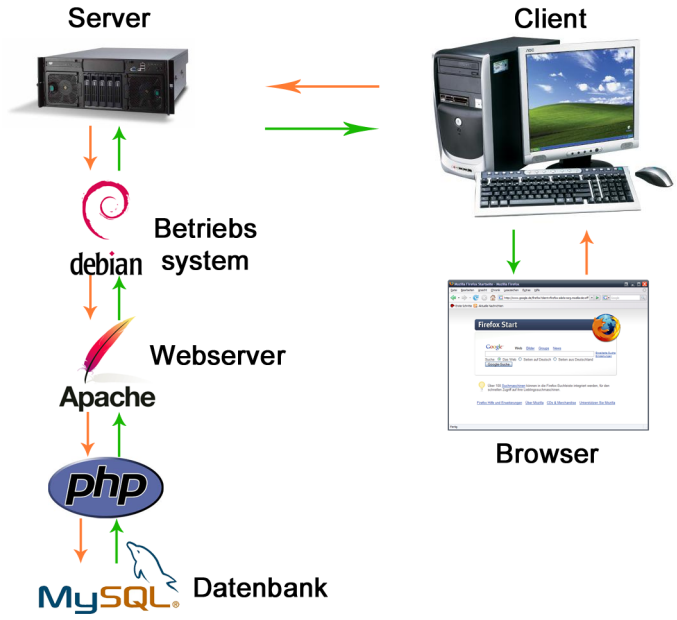

Abbildung 1: Grundsätzlicher Ablauf eines Webseitenaufrufs.

4. Methoden der Web-Performance-Optimierung

Die Methoden zur Leistungssteigerung können wie im Technologiestack untergliedert und jedes Element kann für sich betrachtet werden.

Serverhardware Der einzige Parameter, der am Server verbessert werden kann, ist seine Leistung, dass heißt Datendurchsatz und Rechengeschwindigkeit. Dies geschieht durch CPU Upgrades oder Arbeitsspeichererweiterung. Weitergehend

kann man die Festplattenzugriffsgeschwindigkeiten durch RAID-Verbünde und neue Technologien wie SSD-Speicher verbessern. Wenn ein einzelner Server nicht ausreicht, können auch zusätzliche Server zur Lastverteilung genutzt werden. Entweder es werden die verschiedenen Dienste, die der Server normalerweise anbietet, auf mehrere Server verteilt oder es wird der Server repliziert und die Last auf alle Server gleichmäßig verteilt.

Betriebssystem Der Ansatzpunkt für Verbesserungen auf Betriebssystemebene ist Memory-Mapping, dass heißt Speicherbereiche, die normalerweise auf der Festplatte liegen, werden in den Arbeitsspeicher umgelagert, um die Latenzen zu verringern. Dies wird genutzt, um Caches zu beschleunigen, die normalerweise von der Festplatte lesen.

Datenbank Auf Datenbankebene kann im Fall von MySQL nur beschränkt optimiert werden. Zum einen sind bei häufig genutzten Tabellen Indizes anzulegen und zum anderen kann man MyISAM statt InnoDB nutzen, um performanter zu sein. Datenbanken können auf einem eigenen Server betrieben werden. Falls sie sehr hohen Belastungen ausgesetzt werden, können durch Replikation mehrere Server die Datenbanklast tragen.

Web Server Optimierungen am Webserver sind sehr schwierig, da die Webserver-Performance hauptsächlich von der Serverleistung abhängt. Die Anzahl der gleichzeitigen Zugriffe wird maßgeblich durch den verfügbaren Arbeitsspeicher und die verfügbare Bandbreite eingeschränkt. Für Webseiten mit sehr hohem Traffic werden oft angepasste Versionen von besonders performanten Webservern wie nginx[1] oder lighttpd[2] genutzt. Diese Webserver sind auf Geschwindigkeit ausgelegt und haben eingebaute Lastverteilungssysteme.

PHP hat als Interpretersprache[3] mit Geschwindigkeitsproblemen zu kämpfen. Um diese zu eliminieren, gibt es mehrere Ansätze. Der einfachste ist bessere und effizientere Algorithmen zu programmieren. Außerdem kann durch Opcode-Caching die Zeit eingespart werden, die normalerweise benötigt wird, um den PHP Code zu kompilieren.

Drupal bietet viel Optimierungsspielraum, besonders da eine veraltete Version eingesetzt wird. Viele Inhalte können über Caches zwischengespeichert und somit

[1] Von Igor Sysoev für eine große russische Suchmaschine programmierter Webserver. `http://nginx.org/`

[2] Sehr gut skalierender Webserver von Jan Kneschke. `http://www.lighttpd.net/`

[3] Code wird bei jedem Aufruf neu kompiliert und ausgeführt.

stark beschleunigt werden. Weiterhin können unnötige Module deaktiviert werden. Das führt zu besseren Ausführungszeiten, da der PHP Interpreter weniger Code laden muss. Über Zusatzmodule können fehlende Funktionen wie CSS und Javascriptaggregation nachgerüstet werden.

Client Auf die Clientseite hat der Entwickler in der Regel wenig Einfluss. Verbesserungen können hier nur durch die Nutzung der aktuellsten Browserversionen erzielt werden. Außerdem können Einstellungen, wie die maximale Anzahl paralleler HTTP Verbindungen, im Browser geändert werden.

5. Einordnung von Performanzproblemen

Als Entwickler beziehungsweise Betreiber einer Web-Applikation können vielfältige Problematiken auftreten. Diese kann man in clientseitige sowie serverseitige Probleme unterteilen. Oft lassen sich diese mit Investitionen in die Infrastruktur beheben. Meistens gibt es aber auch Ansätze, die durch effizientere Techniken für Verbesserung sorgen. Dieser Abschnitt versucht die Problemstellungen und Ansätze für Leistungssteigerungen aufzuzeigen.

5.1. Clientseitige Probleme

Auf der Client-Seite hat der Applikations-Anbieter die wenigste Kontrolle. Dadurch dass die Applikation im Web angeboten wird, müssen die Begleitumstände berücksichtig werden. Des Weiteren muss klar sein, für welche Frontends programmiert und optimiert werden muss.

5.1.1. Netzwerkverbindung

Die Netzwerkverbindung, als Verbindungsglied zwischen Client und Server, kann an mehreren Stellen für Probleme sorgen. Betrachtet werden müssen Bandbreite und Latenzen. Latenzen entstehen durch die Wege, die die Informationen zurücklegen müssen sowie durch Verarbeitung in den Netzwerkknoten. Diese Tatsache hat besonders für international ausgerichtete Webseiten Folgen. Je weiter der Webseitenbesucher physisch entfernt ist, desto langsamer wird die Webseite geladen. Bandbreitenprobleme werden gerade in Deutschland noch durch den mangelhaften Breitbandausbau begünstigt. Da-

her müssen Webseitenbetreibern auf die Größe der Webseite, die sie an ihre Betrachter ausliefern, Wert legen. In Abbildung 2 wird deutlich, wie spärlich beispielsweise die neuen Bundesländer mit 16 Mbit/s oder mehr versorgt werden.[BMWi, 2011] Nur in den dunkelgrün markierten Gebieten werden die Haushalte zuverlässig mit schnellem Internet versorgt. In der Infografik in Abbildung 3, die aus dem Internet

Abbildung 2: Breitbandversorgung mit 16Mbit/s oder mehr

World Stats Broadbrand Penetration Report der IWTF von 2009 abgeleitet wurde, sieht man Deutschland im Vergleich zu den führenden Industrienationen.[Frucci, 2009] Der Unterschied zu unserem Nachbarland Frankreich ist sehr deutlich zu erkennen. Deutschland liegt auf dem 14. Platz mit ca 5Mbit/s und hat daher einen hohen Aufholbedarf. Um zu verdeutlichen, wie entscheidend die Geschwindigkeit des Internetzu-

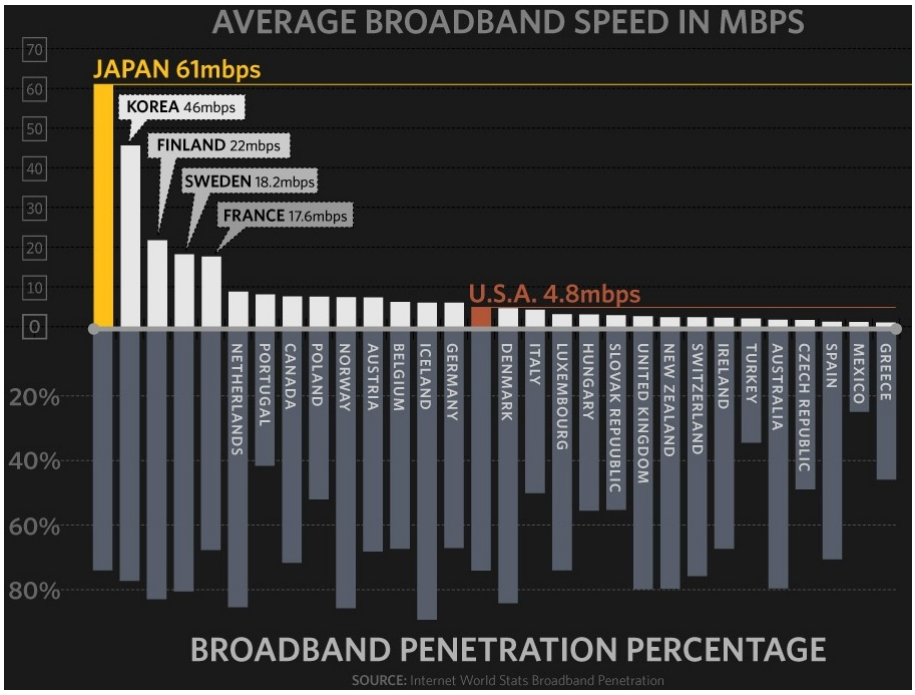

Abbildung 3: Ausschnitt aus einer Infografik des Internet World Stats Broadbrand Penetration Report der IWTF von 2009

gangs für die Ladezeiten einer Webseite ist, folgt ein erläuterndes Beispiel: Die durchschnittliche Webseitengröße beträgt 320 kB, wie im Abschnitt Ausgangszustand im praktischen Teil erläutert wird. Die üblichsten Anschlussgeschwindigkeiten betragen 1Mbit/s, 2Mbit/s, 6Mbit/s, 16Mbit/s, 25Mbit/s und 50Mbit/s. In der Tabelle wird die Zeit verglichen, die für die Übertragung von 320 kB mit den jeweiligen Bandbreiten, benötigt wird.

Mbit/s	1	2	6	16	25	50
s für 320 kB	2,6	1,3	0,44	0,16	0,1	0,05

Wenn die Ergebnisse in Relation zu den üblichen Ladezeiten im einstelligen Sekundenbereich gebracht werden, wird deutlich, wie groß der Einfluss der Bandbreite im ungünstigsten Fall werden kann.

5.1.2. Browser

Als Schnittstelle zwischen Nutzer und Webseite ist der Browser ein besonders kritischer Punkt und muss bei Web-Performanceanalysen besonders beachtet werden. Die meistgenutzten Browser sind der Internet Explorer, Mozilla Firefox, Google Chrome, Opera und Safari. In der Statistik in Abbildung 4 wird deutlich, dass der Internet Explorer, trotz der oft schlechteren Leistung, Marktführer ist.[StatCounter, 2011] Für Entwick-

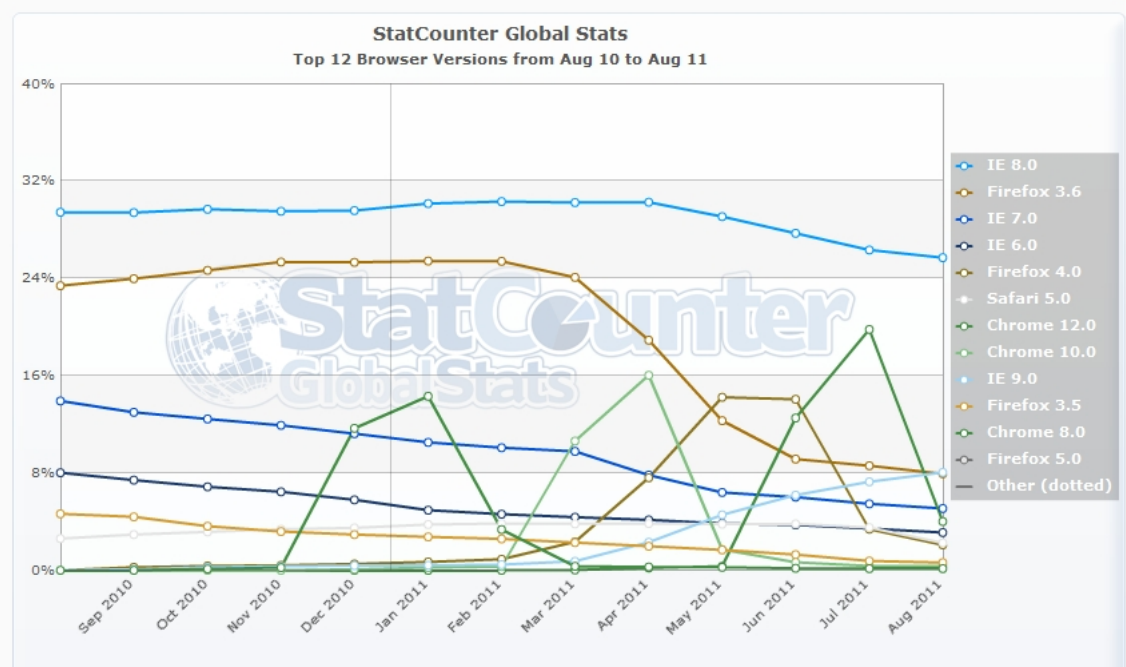

Abbildung 4: Marktanteile der Browser von August 2010 bis August 2011

ler bedeutet die Browser-Vielfalt zusätzlichen Entwicklungsaufwand. Unrühmlichstes Beispiel ist der Internet Explorer 6.[4] Seine Kompatibilitätsprobleme sind so schwerwie-

[4] Auszug aus Bugs des Internet Explorer 6: `http://www.dosonaro.com/6-der-haeufigsten-ie-bugs-und-wie-man-sie-behebt/`

gend, dass mittlerweile auch vom Hersteller Microsoft ein Update dringendst empfohlen wird.[5][Microsoft, 2011] Um die Standardkompatibilität zu messen, wurden die Acid[6] Tests entwickelt. Der Internet Explorer 6 schafft im Acid3[7]-Test nur 11 von 100 Punkten. Moderne Browser, wie Google Chrome und auch der Internet Explorer 9, schaffen Werte über 90 von 100 Punkten. Neben Kompatibilität haben noch andere Leistungsparameter Einfluss auf die Ladezeiten. Die Javascript Ausführungszeiten kann man unter anderem mit dem SunSpider JavaScript Benchmark[8] testen. Einen Vergleich aktueller Browser zeigt Abbildung 5. Es wird deutlich, dass die Browser-Hersteller versuchen ein einheitlich gutes Level zu erreichen. Ein weiterer limitierender Faktor ist

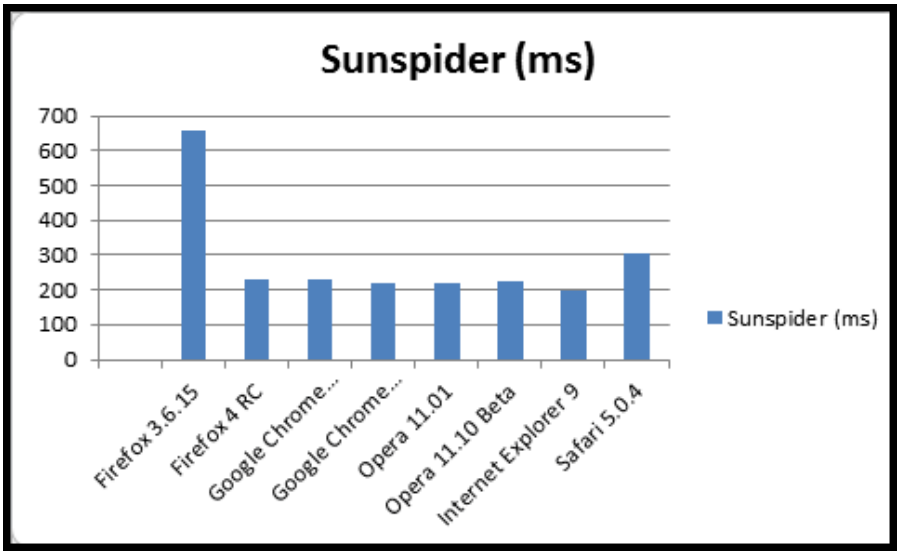

Abbildung 5: SunSpider Benchmark der letzten Browsergeneration

die künstliche Beschränkung der parallelen HTTP Zugriffe durch den HTTP/1.1 Standard. In diesem Standard wird unter anderem festgehalten, dass ein Client die Anzahl paralleler Verbindungen auf zwei beschränken soll. [Fielding, 1999] Das bedeutet, dass mehr Overhead zwischen den Übertragungen entsteht und die Inhalte nicht komplett parallel heruntergeladen werden können. Diese Beschränkungen wurden eingeführt, da die Prozessorlast mit der Anzahl der Verbindungen zunimmt. Außerdem benötigen die Server mehr Arbeitsspeicher, wenn sie zusätzliche Verbindungen aufbauen. Aus

[5] Internet Explorer 6 Countdown: `http://www.ie6countdown.com/`
[6] `http://www.acidtests.org/`
[7] Acid3 Test:`http://acid3.acidtests.org/`
[8] Der Test wird direkt unter folgender URL angeboten: `http://www.webkit.org/perf/sunspider/sunspider.html`

Gründen der Performance haben sich in den letzten Jahren die Browserhersteller von dieser Vorgabe entfernt. Aktuelle Browser ermöglichen sechs bis acht parallele Verbindungen, um die Bandbreite der Nutzer auslasten zu können. Zusätzliche Differenzen entstehen bei der DOM-Verarbeitung im Browser. Die Anzahl der enthaltenen Elemente steht grundsätzlich in negativer Korrelation zu der Geschwindigkeit mit der Operationen auf den Elementen durchgeführt werden können. Ursache ist an dieser Stelle die Notwendigkeit, die Elemente zu selektieren und aufgrund dieser Tatsache muss über alle Elemente iteriert werden. Die Zeit, die für diese Operationen benötigt wird, steigt linear mit der Anzahl an Elementen.[YahooDevNetwork, 2011] Die verschiedenen Browser können mit DOM-Elementen verschieden gut umgehen. Ähnlich wie bei der Javascript-Performance nähern sich die neueren Browser-Generationen einander an.

5.2. Serverseitige Probleme

Moderne, dynamische Webanwendungen sind komplexe Applikationen. In vielen verschiedenen Bereichen kann es zu Problemen kommen. Nicht immer wirken sich diese direkt auf die Web-Performance beim Endnutzer aus. Oftmals werden einfach mehr Ressourcen als nötig genutzt. Dies resultiert aus mangelhafter Konfiguration oder Programmierung.

5.2.1. Netzwerkverbindung

Wie auch beim Client spielt die Netzwerkverbindung des Servers eine entscheidende Rolle. Neben der beim Client erläuterten Latenz tritt beim Server die Bandbreite in den Vordergrund. Die Bandbreite ist das physische Limit an Auslieferungskapazität. Je größer die Webseite ist, die ausgeliefert wird, desto weniger dieser Anfragen kann der Server befriedigen. Natürlich unter der Bedingung, dass der Server in der Lage ist, die entsprechende Menge an Daten bereitzustellen. Wenn die auszuliefernden Daten statischer Natur sind oder es sich um gecachte dynamische Inhalte handelt, ist diese Menge normalerweise ausreichend, um die Netzwerkverbindung zu sättigen.

5.2.2. Datenbankabfragen

So gut wie jede Webseite nutzt mittlerweile Datenbanken zur Verwaltung ihrer Datenbestände. Oft sind diese Datenbanken ein Schwachpunkt für die Web-Performance,

da meistens Daten von der Festplatte gelesen werden müssen und komplexe Abfragen viel Zeit in Anspruch nehmen können. Bei Applikationen, die mehreren Nutzern parallel Zugriff ermöglichen müssen, kann es zu Problemen kommen, wenn die Datenbank Table-Locking verwendet. Dann wird die komplette Tabelle für Schreibzugriffe gesperrt, sobald ein Nutzer einen Datensatz manipuliert. Wenn solche Fälle oft auftreten, sollte zum Row-Locking gegriffen werden, welches nur zeilenweise den Zugriff sperrt. Weitere Verbesserungen können durch die geschickte Nutzung von Indizes erreicht werden. Indizes erreichen eine Beschleunigung von Abfragen die auf Spalten mit großen Feldgrößen zugreifen.

6. Wirtschaftliche Aspekte

Web-Performance ist sehr wichtig für den wirtschaftlichen Erfolg von Projekten, die hauptsächlich durch ihre Webseite repräsentiert werden.

6.1. Usability

Web-Performance hat einen großen Einfluss auf die Bedienbarkeit von Webseiten. Ohne die Innovationen im Gebiet der Web-Performance hätte Google mit GMail und ihren anderen Web-Applikationen keinen Erfolg erzielen können. Die Nutzer waren Desktop-Applikationen gewöhnt. Nur wenn eine Anwendung auch in einer akzeptablen Geschwindigkeit auf Benutzerinteraktionen reagieren kann, hat sie eine Chance auf dem Markt zu bestehen. Es gibt einige Studien, die sich mit Web-Usability auseinandersetzen. Zu den Ergebnissen gehörte unter anderem, dass langsamere Webseiten zu Vertrauensverlust sowie Nutzerfrustration führen. Als weitere Konsequenz ist der empfundene Qualitätsverlust zu nennen. Die genannten Punkte führen letztendlich zu niedrigeren Konversionsraten, das heißt, dass weniger Besucher zu Kunden werden. Zusätzlich steigt die Bail-out-Rate, welche beschreibt, wie viele Nutzer die Webseite während des Ladevorganges wieder verlasssen.[Toll, 2011]

6.2. Google Ranking

Für (Internet-)Firmen ist es von immenser Bedeutung auf den vorderen Plätzen der Google-Suche zu erscheinen. Ein Großteil der Internetnutzer sucht Angebote über die Google-Suche.[Hitwise, 2011] Google hat mit ihrer Initiative „Let's make the web

faster“[9] für viel Entwicklung und Aktivität im Bereich Web-Performance gesorgt und arbeitet zielgerichtet weiter in diese Richtung.[Google, 2011a] Dazu gehören seit 2008 Google AdWords[AdWords, 2008] und seit 2010 die Google-Suche selbst. Dies stellt viele Unternehmen vor die Aufgabe, ihre Webseite zu optimieren und zu beschleunigen, um im Google-Ranking konkurrenzfähig zu bleiben. Wie genau Google die Webseiten testet, ist nur zum Teil bekannt, da diese Informationen zu Googles Geschäftsgeheimnissen gehören. Es finden sich aber einige Fakten, die bei der Optimierung für die Google-Suche, im Bereich Web-Performance, helfen können. Bekannt ist, dass der Google-Suchmaschinen-Bot nichts mit der Geschwindigkeitsmessung zu tun hat und Google nur Daten von echten Nutzern, die die Google-Toolbar in ihrem Browser installiert haben, nutzt. Leider sind Daten nur für Internet Explorer ab Version 6 und Firefox ab Version 2 verfügbar. Als Kriterium für die Bewertung wird die Onload-Zeit gemessen. Dabei führt das verzögerte Nachladen von Inhalten zu einem besseren Ergebnis. Wenn eine Webseite für Google optimiert werden soll, muss also auch die Performance mit berücksichtigt werden. Performance ist aber nur ein Teil des Google-Rankings. Für Webseitenbetreiber gilt daher, dass aktuelle und gut strukturierte Inhalte nicht hinter die Performance gestellt werden dürfen.[Bixby, 2011]

6.3. Serverlast

Eine umfassende Verbesserung der Auslieferungszeiten von Webseiten hat direkten Einfluss auf die Serverperformance. Dies ist leicht in einem Experiment feststellbar, wie folgende Überlegung zeigt: Wenn eine Webseite nur noch die Hälfte der Zeit benötigt, um ausgeliefert zu werden, hat man doppelt soviel Auslieferungskapazität zur Verfügung. Damit können Kosten für Server und Traffic verringert werden. Außerdem ermöglichen Caching und Optimierungen einen Großteil an Prozessorlast einzusparen, der dann für andere Aufgaben zur Verfügung steht. Besonders wichtig ist die Performance in Momenten hoher Zugriffszahlen, beispielsweise wenn eine Webseite in einem prominenten News-Portal erwähnt wird. Oftmals bricht dann die Webseite zusammen, weil die Administratoren nicht mit solch einem Ansturm gerechnet haben. Erwähnenswert ist das Newsportal www.heise.de auf dem regelmäßig davon gesprochen wird, es wurde eine Webseite „geheist“[10]. Noch problematischer wird es, wenn ein Unternehmen Ziel krimineller Aktivitäten wird. Aktuell zu beobachten ist dieses Phänomen bei den Angriffen des Miner-Botnetzes auf mehrere deutsche Web-

[9] `http://code.google.com/intl/de/speed/`
[10] Eine Webseite wurde durch die hohe Anzahl der Zugriffe von Heise-Lesern überlastet.

seiten.[Kaspersky, 2011] Um sich vor solchen Distributed Denial of Service (DDoS) Angriffen schützen zu können, ist eine performante Webseite sehr wichtig, um genug Ressourcen für Firewalls und andere Gegenmaßnahmen zur Verfügung zu haben. Gerade als kleine oder mittelgroße Onlinefirma sollte man sich über die Bedrohungen durch Erpresser im Klaren sein.

Teil III.

Praxisteil

7. Versuchsaufbau

Der Optimierungsversuch kann leider nicht auf dem Livesystem durchgeführt werden, da dass die Stabilität beeinträchtigen würde. Deswegen wurde ein Entwicklungssystem genutzt, um die verschiedenen Methoden zu analysieren. Das Entwicklungssystem ist eine neuere Anschaffung der pludoni GmbH und dadurch, im Vergleich zum Livesystem, leistungsfähiger. Dadurch wird sich das Endergebnis, wenn es nach dem Abschluss der Arbeit auf das Livesystem übertragen wurde, in seinen Kenndaten unterscheiden. Trotz alledem werden die Verbesserungen relativ zum Testsystem proportional ausfallen.

- Testplattform: pludoni Server eq4
- Prozessor: Intel® Core™ i7-920
- Arbeitsspeicher: 8 GB DDR3 RAM
- Festplatten: 2 x 750 GB SATA-II HDD (Software-RAID 1)
- Netzwerkverbindung: 100Mbit
- Server Software: Apache/2.2.16
- PHP Version: PHP 5.2.6-1+lenny13 with Suhosin-Patch 0.9.6.2
- Mysql Version: 14.12 Distrib 5.0.51a
- Betriebssystem: Debian 4.1.2-25 mit Kernel 2.6.26-26lenny2
- Serverstandort: Rechenzentrumspark Falkenstein

8. Testverfahren

Das Testen von Web-Performance hat sich in den letzten Jahren stetig weiterentwickelt. Eine Zeit lang galten sogenannte Backbonetests[11] als das Non-Plus-Ultra. Nachdem aber immer deutlicher wurde, dass die tatsächlichen Ergebnisse sich stark von den Erwartungen unterscheiden, suchte man nach Alternativen. Der verlässlichste Test wird natürlich direkt beim Kunden beziehungsweise der Zielgruppe durchgeführt. Da es oft schwierig ist diese Tests durchzuführen, sollte zumindest mit ähnlichen Voraussetzungen getestet werden.

8.1. Servertest

Die rein serverseitigen Optimierungen wurden mit „ab“ getestet. „ab“, das für „apache bench“ steht, wurde entwickelt, um Apache Server zu testen. Es analysiert, wie lange ein Server braucht, um Webseiten auszuliefern und wieviele Anfragen er pro Sekunde befriedigen kann, ohne dass es zu Ausfällen kommt. „ab“ führt die gewünschte Anzahl an Anfragen hintereinander, mit der gewählten Anzahl an nebenläufigen Zugriffen (Concurrency) auf die Ressource, aus und berechnet dann anhand der Ergebnisse die durchschnittliche Antwortzeit. Der Kommandozeilenbefehl, der zum Testen genutzt wurde, war denkbar einfach:

Listing 1: „ab“ mit Parametern

```
1 ab -n 50 -c 1 http://itsax.it-jobs-und-stellen.de/
```

Dies führt 50 Anfragen mit einer Concurrency von 1 auf itsax.it-jobs-und-stellen.de.

8.2. Browsertest

Browsertests können durch die komplexen Voraussetzungen keine exakten Ergebnisse für alle möglichen Konfigurationen liefern. Ein Netbook zum Beispiel wird eine Webseite immer langsamer darstellen als ein Desktop-PC der neuesten Generation. Man kann natürlich am eigenen Computer Tests durchführen. Dafür gibt es aber auch Pro-

[11] Tests, die direkt am Backbone, also im Rechenzentrum durchgeführt werden. Führt zu besseren Ergebnissen, da der Client und der Testserver direkt verbunden sind. Dieser Test spiegelt aber nicht die Wirklichkeit wieder.

gramme, wie Firebug[12] und YSlow[13], die auch Verbesserungsvorschläge liefern können. Da aber möglichst objektiv getestet werden soll, ist es von Vorteil, eine standardisierte und automatisierte Umgebung zu nutzen. Für diesen Zweck bietet sich WebpageTest[14] an. Es wurde ursprünglich für den internen Gebrauch bei AOL entwickelt, steht aber seit 2008 unter der BSD-Lizenz und kann somit frei genutzt werden. Es ist eine umfassende Testsuite für Webseiten. Besonders die verschiedenen Analysen, die automatisch durchgeführt werden, sind positiv hervorzuheben. Das Programm ist in der Lage, die Webseite nach den „Google Page Speed"-Kriterien zu bewerten und ausführliche Inhaltsanalysen, aufgeschlüsselt nach Multipurpose Internet Mail Extensions (MIME)-Typen, zu liefern. Zusätzlich wird ein sehr hilfreiches Wasserfalldiagramm erzeugt und für jedes Element der Webseite ein Performance-Review erstellt. Darin werden unter anderem Kriterien wie Kompression und Caching betrachtet. Als besonderes Feature gibt es noch einen Videovergleich, in dem man Webseiten gegeneinander antreten lassen kann. Man kann WebpageTest sowohl selbst installieren als auch externe Anbieter nutzen. In diesem Fall wird www.webpagetest.org mit folgenden Parametern genutzt:

- 10 konsekutive Tests
- Standort des Clients: Frankfurt a. M.a
- Browser: IE9
- Connection: DSL 1,5 Mbps / 50ms RTT
- Only First View[15]

8.3. Andere Analyseverfahren

Profiling und Debugging auf Codeebene wurden nur sporadisch eingesetzt, da sie den Anwendungsfall Webseite nur eingeschränkt betreffen. Durch Caching kann die Codeausführungszeit größtenteils eliminiert werden. Bei Anwendungsfällen, die kein Caching erlauben, wie zum Beispiel eine Suche oder eine Applikation, ist es aber sehr hilfreich diese Tools zu verwenden. Sie ersparen viel Arbeit, da die Problemstellen schnell ausgemacht werden können. Von Hand diese Arbeiten durchzuführen, ist in kleinen Projekten sehr mühsam und in großen Projekten fast unmöglich. Jeder Ent-

[12] Entwicklertool für Firefox `https://addons.mozilla.org/de/firefox/addon/firebug/`
[13] Erweiterung für Firebug, die Webseiten bewerten kann. `https://addons.mozilla.org/de/firefox/addon/yslow/`
[14] Link: `https://sites.google.com/a/webpagetest.org/docs/`
[15] Nur ungecachte Zugriffe werden getestet.

wickler sollte solche Programme in seinem Repertoire haben. In Abbildung 6 ist eine Auswertung des Startvorganges mit Hilfe von KCachegrind zu sehen.

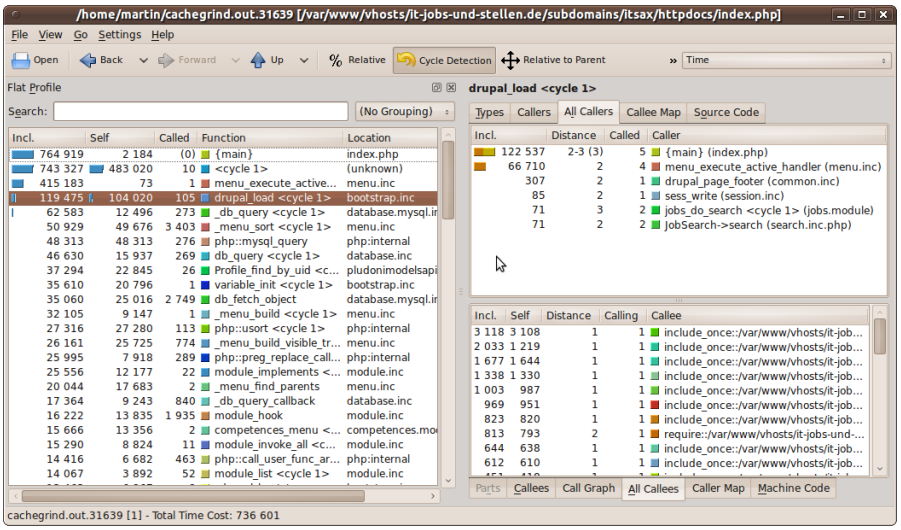

Abbildung 6: KCachegrind ist ein Profilingtool

9. Ausgangszustand

Sreeram Ramachandran, ein Software Ingenieur der Firma Google, hat eine Analyse über 4.2 Milliarden Seiten veröffentlicht. Diese ist im Rahmen der Initiative „Let's make the web faster" entstanden und zeigt häufige Fehlerquellen und ungenutztes Potential auf. Die durchschnittliche Webseite hat laut Ramachandran 320 kB Größe, 44 verschiedene Ressourcen und es werden nur 66% der komprimierbaren Inhalte tatsächlich komprimiert.[Google, 2011b] ITsax.de hat 106 Ressourcen und 444 kB an Daten. Schon anhand dieser zwei Zahlen lässt sich eine vergleichsweise schwache Leistung vorhersehen. Besonders die Anzahl an verschiedenen Ressourcen deutet auf Missstände hin, da Parallelisierung von Zugriffen nur bis zu einem bestimmten Grad möglich ist. Die Time to First Byte (TtFB) von 674ms bezeichnet die Zeit, die vergeht bis der Webbrowser die ersten Daten empfängt. Der Nutzer hat aber zu diesem Zeitpunkt noch keinen Inhalt präsentiert bekommen. Die Inhalte werden erst angezeigt, nachdem die Time to Start Render (TtSR) vergangen ist. Der Nutzer muss demnach ungefähr zwei Sekunden warten, bis die Webseite im Browser anfängt sich aufzubauen. Die Load Time

(LT) bezeichnet dann die Zeit die vergeht, bis die Seite komplett dargestellt wird und der Benutzer sie ohne Einschränkungen bedienen kann. Die Webseite ist in Abbildung 7 zu sehen. Es können aber auch nach der LT noch Inhalte nachgeladen werden, wie

Abbildung 7: Größe der Inhalte

zum Beispiel gestreamte Videos oder andere asynchrone Inhalte. Diese nachgeladenen Inhalte wirken sich aber nicht mehr negativ auf die User Experience aus, solange sie im Rahmen bleiben und nicht wichtige Teile der Webseite, wie zum Beispiel das Hauptmenü, noch per Flash geladen werden müssen. Die Anzahl der DOM Elemente bezeichnet alle vom Browser zu verarbeitenden Objekte und ist ein Indikator für die Komplexität der Webseite. Je mehr Elemente also vorhanden sind, desto länger muss der Browser die Positionierung und Darstellung berechnen. Mit 855 DOM Elementen gehört ITsax.de leider schon zu den komplexeren Seiten, was zu einem Großteil Drupal anzurechnen ist, welches auf Kosten der Performance andere Aspekte, wie Konfigurierbarkeit, in den Vordergrund stellt. Die Inhalte auf der Seite Itsax.de wurden mit dem Analysetool webpagetest.org ausgewertet. Die Ergebnisse sind in Abbildung 8 und 9 dargestellt, einmal in Bezug auf Größe und einmal aufgeschlüsselt nach der Anzahl der

benötigten Requests, um die Inhalte vom Server anzufordern.

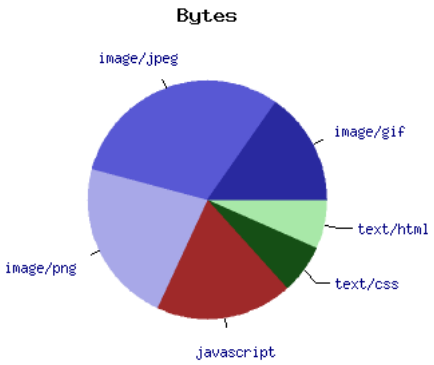

Abbildung 8: Größe der Inhalte

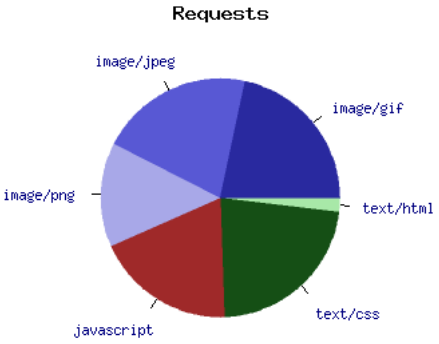

Abbildung 9: HTTP Anfragen

Die numerischen Werte sind in den entsprechenden Tabellen erfasst.

Leicht zu erkennen ist, dass der HTML Anteil, sowohl bei der Größe als auch bei Anzahl der HTTP Anfragen, eine untergeordnete Rolle spielt. Nicht vergessen werden darf aber die Zeit, die der Server benötigt, um den HTTP Code zu generieren. Dies kann man sehr gut im Wasserfalldiagramm in Abbildung 10 erkennen, in welchem der Start des Ladevorgangs hervorgehoben wurde. Bevor der Initiale GET Request abgeschlossen ist, weiß der Browser noch nicht, welche Ressourcen er laden muss und es können noch keine anderen Aktionen ausgeführt werden.

Tabelle 2: Übersicht der Datenmenge

MIME-Typ	Menge in Byte
image/jpeg	142251
image/png	101428
javascript	84758
image/gif	73378
text/css	30222
text/html	29917

Tabelle 3: Übersicht der HTTP-Requests

MIME-Typ	Menge in Byte
text/css	24
image/gif	23
image/jpeg	22
javascript	20
image/png	15
text/html	2

Listing 2: Startwerte gemessen mit ab

```
1  Server Software:        Apache/2.2.16
2  Server Hostname:        itsax.it-jobs-und-stellen.deDocument Path:
                 /
3  Document Length:        65218 bytes
4
5  Concurrency Level:      1
6  Time taken for tests:   18.182 seconds
7  Complete requests:      50
8
9  Write errors:           0
10 Total transferred:      3266726 bytes
11 HTML transferred:       3239226 bytes
12 Requests per second:    2.75 [#/sec] (mean)
13 Time per request:       363.640 [ms] (mean)
14 Time per request:       363.640 [ms] (mean, across all concurrent
       requests)
15 Transfer rate:          175.46 [kBytes/sec] received
```

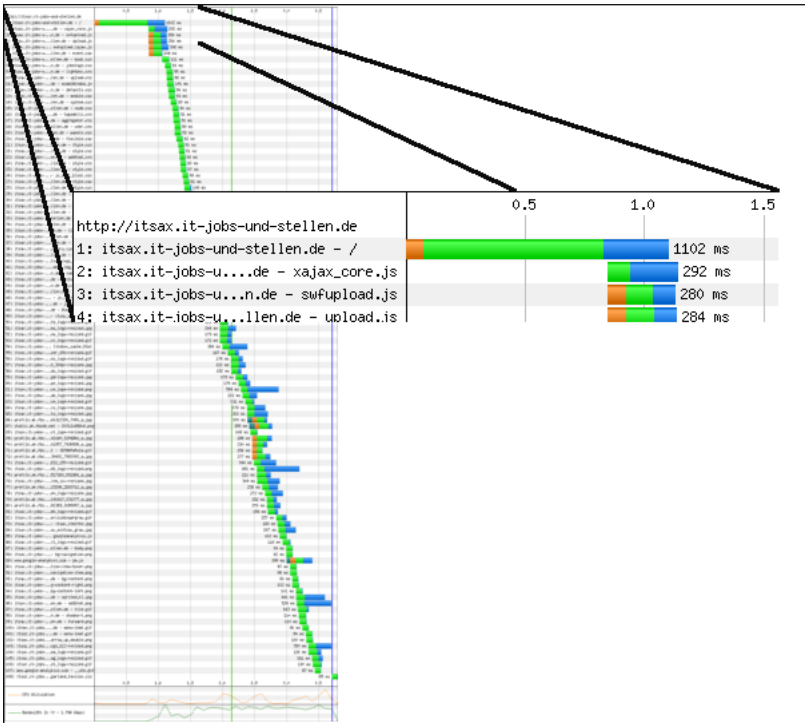

Abbildung 10: Wasserfalldiagramm: Ausgangszustand

10. Implementierung und Test der einzelnen Methoden

Im folgenden Abschnitt werden die eingesetzten Methoden dargestellt und ihre Auswirkungen auf die Webperformance der Seite ITsax.de aufgezeigt. Dabei werden alle Methoden, die vom Autor, nach der Analyse des Ausgangszustands, für möglicherweise sinnvoll befunden wurden, getestet. Die Methode wird jeweils mit dem Startwert verglichen und der Aufwand eingeschätzt.

10.1. Alternative PHP Cache

Als allererste Maßnahme wurde ein PHP-Opcode-Cache untersucht. Opcode-Caches speichern das Kompilat vorangegangener Operationen und können so beim nächsten Aufruf direkt die Berechnung ausführen. Solche Systeme sind immer zu empfehlen, da sie normalerweise keinerlei Probleme bereiten und alle PHP Anfragen beschleunigen. Aktuell gibt es mehrere dieser, auch PHP-Beschleuniger genannte, Opcode-Caches.

Unter ihnen nimmt aber der Alternative PHP Cache (APC)[16] eine führende Position ein. Hauptsächlich weil es der zuverlässigste unter den nicht-kommerziellen Caches ist. Ab PHP Version 5.4 wird er in PHP schon enthalten sein. Die Installation von APC ist auf einigermaßen aktuellen Linux Servern auch denkbar einfach. Das PHP-Pear-Framework, was oft schon installiert ist, erlaubt mit einigen wenigen Befehlen einen funktionierenden Opcode-Cache zu installieren.

Listing 3: Installation von APC

```
1 apt-get install php-pear
2 pecl install apc
```

Nach der Installation wurde die Antwortzeit des Apacheservers gemessen und es kam zu folgendem Ergebnis. Statt 363 ms wurden nur noch 233 ms benötigt. Das bedeutet eine Verbesserung um 36% und es können 4,28 Anfragen je Sekunde bearbeitet werden gegenüber vorher nur 2,75. Alles unter der Prämisse, dass es keine nebenläufigen Anfragen gibt.

Listing 4: APC gemessen mit ab

```
1  Server Software:        Apache/2.2.16
2  Server Hostname:        itsax.it-jobs-und-stellen.de
3  Server Port:            80
4
5  Document Path:          /
6  Document Length:        64842 bytes
7
8  Concurrency Level:      1
9  Time taken for tests:   11.681 seconds
10 Complete requests:      50
11
12 Write errors:           0
13 Total transferred:      3261730 bytes
14 HTML transferred:       3234230 bytes
15 Requests per second:    4.28 [\#/sec] (mean)
16 Time per request:       233.620 [ms] (mean)
17 Time per request:       233.620 [ms] (mean, across all concurrent
       requests)
18 Transfer rate:          272.69 [kBytes/sec] received
```

[16] http://pecl.php.net/package/APC

10.2. Drupal Cache

Der Drupal Cache wurde mit Version 5 eingeführt und kann über das Administrationsinterface aktiviert werden. Durch ihn wird ein Caching aller Seiten ausgelöst. Dabei werden die fertig berechneten Webseiten beim ersten Aufruf komplett in der Datenbank abgelegt. Wenn eine schon gecachte Seite aufgerufen wird, muss nur noch der fertige HTML-Code aus der Datenbank geholt werden. Diese Methode ist aber zu unflexibel, da nicht konfigurierbar ist, welche Seiten gecacht werden sollen. Für die Webseite ITsax.de müssen aber dynamische Inhalte ungecacht bleiben. Testweise wurde der Cache aktiviert und mit „ab“ getestet. Die Ergebnisse sind in folgender Ausgabe zu sehen.

Listing 5: Drupal Cache gemessen mit ab

```
1  Server Software:        Apache/2.2.16
2  Server Hostname:        itsax.it-jobs-und-stellen.de
3  Server Port:            80
4
5  Document Path:          /
6  Document Length:        65005 bytes
7
8  Concurrency Level:      1
9  Time taken for tests:   2.082 seconds
10 Complete requests:      50
11 Failed requests:        0
12 Write errors:           0
13 Total transferred:      3276900 bytes
14 HTML transferred:       3250250 bytes
15 Requests per second:    24.02 [\#/sec] (mean)
16 Time per request:       41.638 [ms] (mean)
17 Time per request:       41.638 [ms] (mean, across all concurrent requests
       )
18 Transfer rate:          1537.09 [kBytes/sec] received
```

Die Zeit pro Anfrage wird deutlich kleiner und es wird klar, dass Caching für den serverseitigen Teil die größten Verbesserungen bringt. Die PHP-Codeausführung kann auf ein notwendiges Minimum, dass heisst auf die Momente in denen der Cache neu geschrieben wird, reduziert werden. So kann zumindest der zu berechnende HTML-Code direkt ausgeliefert werden.

10.3. JS Aggregation und Minifizierung mit dem Javascript Aggregation Modul

Aggregation und Minifizierung sind Verfahren, die in der Web-Performance-Optimierung häufig eingesetzt werden. Für das Drupal 5 System gibt es fertige Module, die diese Aufgabe übernehmen. Das Modul interveniert innerhalb des Drupal Kerns. Dort werden die Javascripte ersetzt, die vorher direkt so ausgegeben wurden wie sie die Module lieferten. Stattdessen wird eine einzelne, das heisst aggregierte Version, die optional minifiziert werden kann, ausgeliefert. Diese Minifizierung wird natürlich genutzt und spart einige kB. Ermöglicht wird dies durch die Nutzung von JSmin[17]. JSmin ist ein Filter, der unter anderem Kommentare entfernt und mehrere Leerzeichen zu einem zusammenfasst. Durch die Nutzung dieser Methode konnten 11 HTTP Abfragen und 11 kB an Datenvolumen eingespart werden.

Tabelle 4: Ergebnis der JS Aggregation und Komprimierung

Load Time:	3.658s
Time to First Byte:	0.595s
Time to Start Render:	1.915s
#DOM Elements:	844
#Requests:	95
Bytes In:	432 kB

10.4. CSS Aggregation und Komprimierung

Analog zur Javascript Optimierung kann man auch die CSS Aggregation betrachten. Dabei werden durch Zusammenfassung der einzelnen CSS Dateien HTTP Abfragen eingespart. Wie man an der gesunkenen Anzahl der Requests sehen kann, wurden 21 Abfragen nur durch Zusammenfügen der einzelnen CSS Dateien zu einer einzigen eingespart. Durch die Bereinigung beziehungsweise Komprimierung werden dabei außerdem 17 kB an überflüssigen Zeichen entfernt.

[17] Bibliothek zur Minifizierung von JavaScript. `https://github.com/rgrove/jsmin-php/`

Tabelle 5: Ergebnis der CSS Aggregation und Komprimierung

Load Time:	3.577s
Time to First Byte:	0.649s
Time to Start Render:	1.577s
#DOM Elements:	834
#Requests:	85
Bytes In:	427 kB

10.5. Drupal Boost Modul

Das Boost Modul ist ein Cache für statische Seiten, der allerdings nur für nicht-angemeldete Gastnutzer funktioniert. Da ITsax.de fast ausschließlich von Gastnutzern besucht wird, bietet es sich an, dieses Modul zum Cachen kompletter Seiten zu nutzen. Dabei können HTML, XML, Ajax, CSS und Javascript Dokumente gecacht werden. Zusätzlich können diese durch gzipping komprimiert werden. Die genutzen Techniken sind sehr performant und bauen auf einem Dateisystemcache auf. Das bedeutet, jede Seite wird komplett auf der Festplatte abgelegt. Alle Arten von serverseitigen Prozessen werden so nach dem initialem Seitenaufruf, der das Caching auslöst, nicht mehr durchlaufen. Dies sieht man sehr gut an der Time to First Byte. Von den 172 ms werden ca. 50 ms für den DNS Lookup benötigt und 70 ms für die erste HTTP Verbindung. Der Server braucht demnach nur ca. 50 ms um die Seite auszuliefern. Dies hat natürlich einen sehr positiven Einfluss auf die Gesamtperformance und macht sich besonders beim Endnutzer bemerkbar, da die Seite spürbar schneller anfängt, sich im Browser aufzubauen.

Tabelle 6: Ergebnis der Aktivierung des Drupal Boost Moduls

Load Time:	3.233s
Time to First Byte:	0.172s
Time to Start Render:	1.473s
#DOM Elements:	856
#Requests:	106
Bytes In:	444 kB

10.6. Bildkompression mit jpegoptim und OptiPNG

Bilder und Grafiken bieten oft großen Optimierungsspielraum. Zum einen erfolgt dies durch die richtige Auswahl der Dateiformate und zum anderen durch Komprimierung der Bilder. Da es auf www.itsax.de nicht nur statische Inhalte gibt, sondern auch durch Communitymitglieder und Communitymanager eingestellte Inhalte verwaltet werden müssen, sollte eine nachträgliche Qualitätsoptimierung der hochgeladenen Bilder durchgeführt werden. Um dies umzusetzen, sind die Programme OptiPNG und jpegOptim zu empfehlen. Da es sich bei beiden Programmen um Kommandozeilenprogramme handelt, kann man ihre Anwendung leicht automatisieren. Mit dem Linuxbefehl „find“, der praktischerweise eine Möglichkeit, Befehle auszuführen, besitzt, kann man direkt die entsprechenden Dateien an die Optimierer übergeben. Diese Aktionen können dann über einen Cronjob periodisch jede Nacht ausgeführt werden.

10.6.1. Verlustfreie Kompression

Technisch wird eine verlustfreie Kompression durch Verfahren wie die Huffman-Codierung oder die Lauflängenkodierung umgesetzt. Im Fall des PNG-Formats wird die Komprimierungsmethode Deflate genutzt. Außerdem werden Vorfilter, in Form von prädikativer Kodierung, eingesetzt. Diese berechnen die wahrscheinlichen Farbwerte und es müssen nur die Abweichungen gespeichert werden.

Die Befehle sehen dann wie folgt aus:

Listing 6: verlustfreies Optimieren mit ”find”

```
1 find . -name "*.png" -exec optipng -o7 {} \;
2 find . -name "*.jpg" -exec jpegoptim {} \;
```

Auf jeden Fall sollte eine verlustfreie Komprimierung durchgeführt werden, da die Bilder in diesem Fall nur an Dateigröße verlieren und die Bildqualität unberührt bleibt.

10.6.2. Verlustbehaftete Kompression

Da im Web kein besonders hoher Detailgrad gewährleistet werden muss, ist auch eine Kompression erwünscht, die auf Kosten der Bildqualität die Bildgröße verringert.

Tabelle 7: Ergebnis der verlustfreien Kompression der Bilder

Load Time:	3.640s
Time to First Byte:	0.636s
Time to Start Render:	1.894s
#DOM Elements:	855
#Requests:	106
Bytes In:	429 kB

Listing 7: verlustbehaftetes Optimieren mit find

```
1 find . -name "*.jpg" -exec jpegoptim -m50 {} \;
```

Tabelle 8: Ergebnis der verlustbehafteten Kompression der Bilder

Load Time:	3.255s
Time to First Byte:	0.669s
Time to Start Render:	1.908s
#DOM Elements:	855
#Requests:	106
Bytes In:	371 kB

10.7. Drupal 5 Fehler bei umgefärbten Themes

Das Framework Drupal 5 benutzt Themes zur Gestaltung der Oberfläche. Um diese farblich anpassen zu können, wurde das Color Modul installiert, welches Themeveränderungen ermöglicht. Aufgrund der Tatsache, dass das Theme nur kopiert wird und im Anschluss die Farben geändert werden, entstehen bei diesem Vorgang unnötige Duplikate, die beim Laden der Seite mitgeschleppt werden. Um diese zu entfernen, wird einfach das Standardtheme durch das Modifizierte ersetzt. Dafür müssen nur noch einige Pfade in der style.css angepasst werden und man spart in dem Fall von itsax.de 4 kB, was immerhin ca 1% der übertragenen Datenmenge ausmacht. Umgesetzt wurde diese Maßnahme, indem die Duplikate manuell zu einer Datei zusammengeführt wurden. Oftmals müssen nur Farbangaben ersetzt werden, um die gewünschten Resultate zu erzielen. Danach wird das Theme durch die zusammengeführte Datei ersetzt. Das Color-Modul kann im Anschluss deaktiviert werden.

Tabelle 9: Ergebnis der Fehlerbereinigung des Theme Moduls

Load Time:	3.626s
Time to First Byte:	0.629s
Time to Start Render:	1.890s
#DOM Elements:	854
#Requests:	104
Bytes In:	440 kB

10.8. Theme Bilder spriten

Spriting wurde ursprünglich in der Videospielentwicklung verwendet, um Bilder in den Grafikspeicher zu laden. In der Webentwicklung ist es eine effektive Technik, um Bilder ohne mehrfachen Overhead zu laden. Beim Spriting wird aus vielen Einzelbildern eine Gesamtgrafik erstellt. Um auf die Ursprungsbilder dann einzeln zugreifen zu können, werden CSS Befehle genutzt, die es ermöglichen, die Größe und die Position eines Bildausschnittes anzuzeigen. Ein Ausschnitt der CSS-Datei ist in folgendem Code zu sehen.

Listing 8: CSS-Spriting

```
ul li, ul.menu li, .item-list ul li, li.leaf {
  margin: 0.15em 0 0.15em .5em;
  padding: 0 0 .2em 1.5em;
  list-style-type: none;
  list-style-image: none;
  background: transparent url(images/garland-sprites.png) no-repeat 1px
      .35em;
  background-position:0 -159px;
}

ul li.expanded {
  background-position:0 -119px;
  height: 10px;
}

ul li.collapsed {
  background-position:0 -79px;
}
```

Im ersten Block wird das gespritete Bild eingebunden. Mit Hilfe von background-position kann dann der Bildausschnitt verschoben werden. Die Höhenangabe bewirkt, dass nur der gewünschte Ausschnitt angezeigt wird.

Tabelle 10: Ergebnis des Spritings der Theme Bilder

Load Time:	3.707s
Time to First Byte:	0.669s
Time to Start Render:	1.968s
#DOM Elements:	855
#Requests:	103
Bytes In:	443 kB

10.9. Module von Startseite entfernen

Um zu überprüfen, welchen Einfluss verschiedene, im Netzwerkgraphen auffällige, Module auf die Gesamtperformance haben, werden sie testweise komplett deaktiviert. So kann man entscheiden, bei welchen Modulen zusätzlicher Programmieraufwand lohnenswert ist.

10.9.1. Partnerslideshow

Das Deaktivieren der Partnerslideshow hat gravierenden Einfluss auf die Gesamtperformance. Das Modul lädt im aktiven Zustand alle Bilder, die es benötigt und bremst damit den gesamten Ladevorgang aus. Die Ladezeit verringert sich um 27,7%. Der größte Teil des Geschwindigkeitszuwachses ist der Verringerung der Übertragungsmenge zuzuschreiben.

Tabelle 11: Ergebnis der Deaktivierung der Partnerslideshow

Load Time:	2.695s
Time to First Byte:	0.636s
Time to Start Render:	1.838s
#DOM Elements:	790
#Requests:	80
Bytes In:	276 kB

10.9.2. Facebook Widget

In diesem Fall ist der Einfluss auf die Performance geringer, aber mit einer Verbesserung von 6,7% auf jeden Fall vorhanden. Das Widget besteht aus neun kleinen Fotos und dem Facebookrahmen. Den größten Effekt hat hier die Verringerung der Anzahl der HTTP Anfragen. So können wichtigere Elemente schneller geladen werden.

Tabelle 12: Ergebnis der Deaktivierung des Facebook-Widgets

Load Time:	3.478s
Time to First Byte:	0.668s
Time to Start Render:	1.966s
#DOM Elements:	779
#Requests:	95
Bytes In:	406 kB

10.9.3. Jobleiste deaktiviert

Die Jobleiste hat einen geringfügig größeren Einfluss auf die Ladezeiten als das Facebook Widget. Die Ursache ist wahrscheinlich in den zusätzlichen Bibliotheken zu suchen, die zu diesem Modul gehören. Darunter sind Ajax Bibliotheken und anderer ThirdParty Code.

Tabelle 13: Ergebnis der Deaktivierung der Jobleiste

Load Time:	3.367s
Time to First Byte:	0.617s
Time to Start Render:	1.709s
#DOM Elements:	822
#Requests:	94
Bytes In:	411 kB

10.10. Umprogrammierung der Module

Um die langsamen Module weiterhin nutzen zu können, muss eine Lösung gefunden werden, die es ermöglicht, Inhalte nachzuladen, nachdem die Seite komplett geladen wurde. Um diese Asynchronität zu erreichen, wird mit Hilfe eines Timeout-Befehls,

das Laden der nicht priorisierten Inhalte verzögert. Programmiert wird dieses Verhalten mit Javascript, genauer JQuery. Besonders betrachtet werden muss dabei, ob eine vorhandene Dynamik erhalten bleiben soll. Dazu gehören zum Beispiel zufällige Elemente oder Elemente mit besonders häufigen Aktualisierungen.

10.10.1. Partnerslideshow

Abbildung 11: Partnerslideshow

Die Partnerslideshow ist eher ein kosmetisches Element der Startseite und für den Nutzer nicht notwendig. Daher kann es so programmiert werden, dass beim Laden nur ein leeres DIV übergeben wird. Dann wird ein Timeout gestartet und bei der Aktivierung des Timeoutevents wird das DIV mit den Slideshowelementen gefüllt sowie die Slideshow gestartet. Der Code wurde dahingehend angepasst, dass der HTML Code in eine Javascript Variable geschrieben wird, anstatt ihn direkt in der Seite einzufügen. Dadurch ist es möglich über eine DOM Manipulation den HTML Code später einzufügen. Das stellt sich dann wie im folgenden Codeaussschnitt dar.

Listing 9: Javascript - Slideshow

```
1  drupal_add_js('
2    if (Drupal.jsEnabled) {
3    $(document).ready(function() {
4       window.setTimeout(delayed_partnerbox,500);
5    });
6    function delayed_partnerbox(){
7      $("#projects").parent().html(partner_box_load);
8      $(".slideshow").cycle({
9          delay:  200 ,
10         height: "80px",
11         width: "180px",
12         containerResize: 0,
13         fit: 1,
14         fx: "fade"
15     });
16   }
17 }', 'inline')
```

An das window Objekt wird in Zeile 3 ein Timer gehängt. Der Timer löst nach 500 ms einen Callback aus, durch den die Funktion delayed_partnerbox ausgeführt wird. Durch diese Verlagerung des Inhalts braucht die Webseite nur noch 2.749s zum Laden. Bis die Webseite die nachgeladene Slideshow anzeigt, vergehen 3.794s. Die Webseite ist also eher benutzbar und später komplett geladen. In Testverfahren, wie zum Beispiel für das Google Ranking wird nur betrachtet, wann die Seite das Onload Event auslöst. Somit wurde die Seite für Google und den Nutzer schneller. Der Performancegewinn geht, wie in der vorherigen Analyse festgestellt, aus der verringerten Datenmenge hervor. Natürlich spielt auch die Reduzierung der HTTP Anfragen eine Rolle. Die Datenmenge wurde auf 284 kB verringert und es wurden statt 107 nur 82 HTTP Anfragen benötigt.

Tabelle 14: Ergebnis der Umprogrammierung der Partner-Slideshow

Load Time:	2.749s
Time to First Byte:	0.626s
Time to Start Render:	1.832s
#DOM Elements:	855
#Requests:	82
Bytes In:	284 kB

10.10.2. Facebook Widget

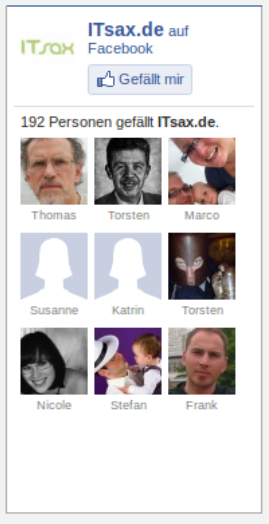

Abbildung 12: Das Facebook-Widget

Das Modul für das Facebook Widget wurde durch Entwickler der pludoni GmbH so programmiert, dass es das von Facebook zur Verfügung gestellte Widget cacht und dann auf der Webseite darstellt. Das Hauptproblem bei dem Widget ist, dass die Profilbilder, die angezeigt werden, auf den Servern von Facebook liegen und sich somit der internen Kontrolle entziehen. Um das Modul abzukoppeln wird ein Timer analog zum vorhergehenden Modul programmiert.

Listing 10: Facebook Widget

```
function facebook_show_likebox() {
  $local_path = drupal_get_path("module","facebook_pludoni");
  drupal_add_js('
  if (Drupal.jsEnabled) {
      $(document).ready(function() {
      window.setTimeout(delayed_facebookbox,400);
    });
    function delayed_facebookbox(){
      $("#facebook-likebox").html("<iframe frameborder=\"0\" scrolling=\"
          no\" src=\"/'.$local_path.'/caching/likebox_cache.html\" id=\"
          fbooklikebox\"></iframe>");
    }
  }', 'inline');
  return '<div id="facebook-likebox"></div>';
}
```

Wieder wird als initialer Rückgabewert nur ein leeres DIV-Element ausgegeben. Diesmal kann der HTML Code aber direkt eingefügt werden, ohne den Umweg über eine Javascriptvariable gehen zu müssen. Durch die Verringerung der HTTP Anfragen um 11 und der Webseitengröße um 38 kB kann eine Verbesserung der Ladezeit von 300 ms erzielt werden.

Tabelle 15: Ergebnis der Umprogrammierung des Facebook-Widgets

Load Time:	3.411s
Time to First Byte:	0.576s
Time to Start Render:	1.848s
#DOM Elements:	857
#Requests:	95
Bytes In:	406 kB

10.10.3. Partneranzeigen

Das Partneranzeigen-Modul musste umprogrammiert werden, da die Startseite durch das Boost Modul gecacht werden soll. Caching bedeutet normalerweise die Abwesenheit dynamischer oder zufälliger Elemente. Da aber bei jedem Besuch der Webseite ein zufälliger Communitypartner angezeigt werden soll, musste eine Lösung gefunden

Abbildung 13: Partneranzeigen Block

werden. Gelöst wurde das Problem durch Auslagerung des Codes in einen Callback, der unter einer anderen URI angesprochen werden kann. Das Boost Modul kann solche Inhalte auch cachen. Es ist aber auch über einen URI Filter konfigurierbar. So kann eine performante Startseite mit dynamischen Inhalten realisiert werden. Die Ladezeit wird durch diesen Umweg leider negativ beeinflusst, aber 100 ms stehen in keinem Verhältnis zu dem Gewinn, der später erzielt wird. Da in einer Drupal Umgebung gearbeitet wird, kann leicht über einen Menü-Hook[18] dem System die neue URI mitgeteilt werden. Wenn sie, nachdem sie bekannt gemacht wurde, aufgerufen wird, liefert sie den einzufügenden HTML Code aus.

[18] Hooks sind eine Programmierschnittstelle von Drupal.

Listing 11: Ajax - hook

```
function partneranzeigen_menu() {
  $items = array();
  $items[] = array(
    'path'          => "partanz/ajax",
    'callback'      => 'partneranzeigen_display_block_1',
    'type'          => MENU_CALLBACK,
    'access'        => true
  );
  return $items;
}
```

Eingefügt wird der HTML Code dann über eine Ajax GET Anfrage. Mit der schon bekannten Timeout Methode.

Listing 12: Ajax - GET

```
function delayed_partneranzeigen(){
  $.get('/partanz/ajax',function(data){
    $('#partneranzeigenbox').html(data);
});
```

Obwohl die Seite um 31 kB kleiner geworden ist und zwei HTTP Anfragen eingespart werden, wird sie, wie schon erwähnt, langsamer.

Tabelle 16: Ergebnis der Umprogrammierung des Partneranzeigen-Moduls

Load Time:	3.871s
Time to First Byte:	0.669s
Time to Start Render:	2.255s
#DOM Elements:	857
#Requests:	105
Bytes In:	417 kB

11. Endergebnis

Nach der Zusammenführung aller Optimierungen wurde die Webseite erneut getestet. Das Wasserfalldiagramm der optimierten Webseite ist in Abbildung 14 dargestellt. Nach nur einer Sekunde fängt der Browser an die Seite darzustellen. 2,913 Sekunden

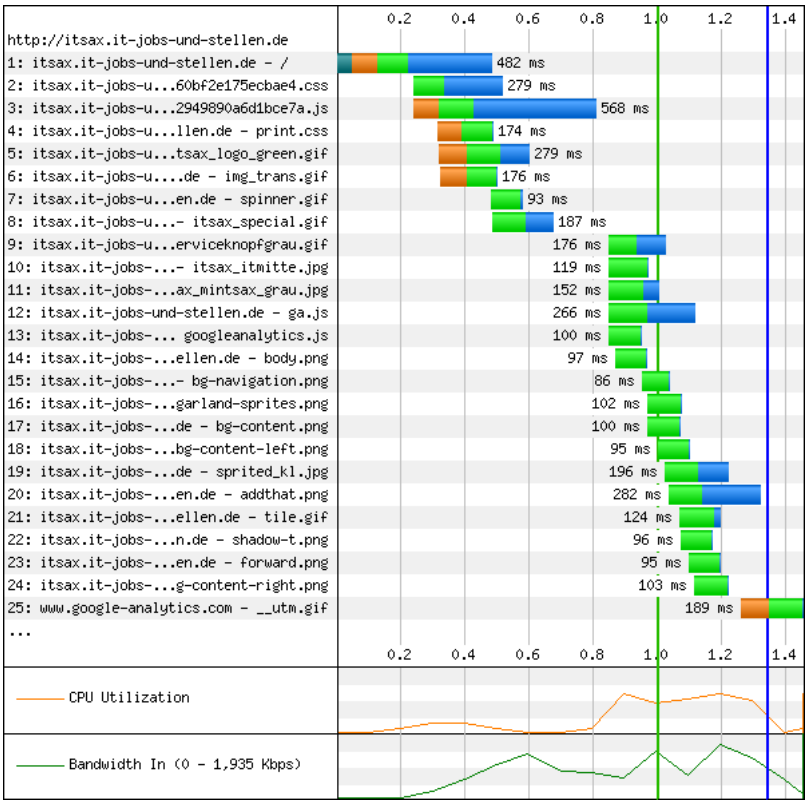

Abbildung 14: Wasserfalldiagramm: Optimierte Webseite

werden benötigt bis alle asynchronen Inhalte verfügbar sind. Die Webseitengröße wurde auf 138 kB komprimiert. Daher können auch langsame Internetverbindungen die Seite schnell laden. Von den 107 HTTP Anfragen sind nur noch 67 übrig geblieben, von denen aber lediglich 25 vor dem Onload-Event benötigt werden. Dank dieser Reduzierung auf das Wesentliche erwartet den Besucher nun eine subjektiv wesentlich angenehmere Webseite.

12. Vergleich

Ein abschließender Vergleich aller Methoden wurde tabellarisch zusammengetragen. In der ersten Spalte steht der Methodenname, darauf folgen die Gesamtladezeit und die Zeit, die der Nutzer warten muss bis der Browser anfängt Inhalte darzustellen. Zu jedem Ergebnis wird noch eine Einschätzung abgegeben. Abschliessend wurde das Gesamtergebnis in der letzten Zeile dargestellt.

13. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde beispielhaft an der Startseite der Community ITsax eine Web-Performanceanalyse durchgeführt und offensichtliche Fehler behoben. Zusätzlich wurden die Verbesserungen so dokumentiert, dass sie auf alle anderen Portale der pludoni GmbH leicht angewendet werden können. Auf dem Gebiet der Web-Performance gibt es ständig neue Entwicklungen, da sehr viele exzellente Programmierer in Firmen wie Google, Microsoft und Yahoo an dieser Entwicklung beteiligt sind. Oftmals ist es daher schwer die effektivsten Verbesserungen herauszufiltern und Prioritäten zu setzen. Die Arbeit hat gezeigt, dass ein Fokus auf die Bereiche der Verringerung der Anzahl der HTTP-Anfragen und der Webseitengröße zu sehr guten Ergebnissen führt. Weiterführende Maßnahmen könnten zum Beispiel die Nutzung von CDN Services sein. Auch eine aktuellere Drupal Version könnte in Betracht gezogen werden, dies ist aber mit einem erheblichen Zeitaufwand verbunden. Da der Fokus dieser Arbeit nur auf der Startseite lag, wurden andere kritische Bereiche, wie zum Beispiel die Suche, noch nicht betrachtet. Gerade bei dynamischen Elementen wäre eine weiterführende Beschäftigung mit den Möglichkeiten des Profiling und des Debugging anzuraten.

Tabelle 17: Auswertung der Ergebnisse

Methode	Load Time	TtSR	Kommentar
Ausgangszustand	3,728s (±0, 00%)	2,002s (±0, 00%)	Der Ausgangszustand ohne Optimierungen.
JS Aggregation	3,658s (−1, 87%)	1,915s (−4, 35%)	Bei geringen Aufwand können hier erste Verbesserungen erzielt werden.
CSS Aggregation	3,577s (−4, 05%)	1,577s (−21, 23%)	Diese Optimierung sollte auch auf jeden Fall durchgeführt werden.
Boost Modul	3,233s (−13, 22%)	1,473s (−26, 42%)	Caching ist Pflicht! Nur in seltenen Ausnahmen ist davon abzusehen.
Bildkompression verlustfrei	3,640s (−2, 31%)	1,894s (−5, 4%)	Durch Verringerung der Datenmenge können besonders Handynutzer profitieren.
Bildkompression verlustbehaftet	3,255s (−12, 68%)	1,908s (−4, 7%)	Zusätzliche verlustbehaftete Kompression sorgt oft für schnelleren Seitenaufbau, da große Mengen an Daten eingespart werden können.
Drupal 5 Theme Fehler	3,626s (−2, 73%)	1,890s (−5, 59%)	Durch diese sehr spezielle Optimierung können auch 100ms gewonnen werden.
Theme Bilder gesprited	3,707s (−0, 56%)	1,968s (−1, 69%)	Spriting ist eine gute Möglichkeit Requests zu sparen, macht sich aber leider nicht stark in der Ladezeit bemerkbar.
Partnerslideshow	2,749s (−26, 26%)	1,842s (−7, 99%)	Sehr deutlich wird die Geschwindigkeit durch das asynchrone Nachladen von Inhalten verbessern.
Facebook Widget	3,411s (−8, 50%)	1,848s (−7, 69%)	Besonders bei Inhalten von externen Quellen ist es nützlich die Ladezeiten abzukoppeln.
Partneranzeigen	3,871s (+3,83%)	2,225s (+11.13%)	An diesem Beispiel wird deutlich, dass nachladen nicht unbedingt Verbesserung sein muss. Trotzdem ist diese Umprogrammierung nötig, um das Caching der Startseite zu ermöglichen.
Gesamtergebnis	1.354s (−63, 68%)	0.952s (−52, 45%)	Wenn alle Verbesserungen auf einmal aktiviert werden, wird die Webseite dramatisch beschleunigt.

Abbildungsverzeichnis

1. Grundsätzlicher Ablauf eines Webseitenaufrufs. 11
2. Breitbandversorgung mit 16Mbit/s oder mehr 14
3. Ausschnitt aus einer Infografik des Internet World Stats Broadbrand Penetration Report der IWTF von 2009 15
4. Marktanteile der Browser von August 2010 bis August 2011 16
5. SunSpider Benchmark der letzten Browsergeneration 17
6. KCachegrind ist ein Profilingtool 25
7. Größe der Inhalte 26
8. Größe der Inhalte 27
9. HTTP Anfragen 27
10. Wasserfalldiagramm: Ausgangszustand 29
11. Partnerslideshow 39
12. Das Facebook-Widget 41
13. Partneranzeigen Block 43
14. Wasserfalldiagramm: Optimierte Webseite 45

Tabellenverzeichnis

2. Übersicht der Datenmenge 28
3. Übersicht der HTTP-Requests 28
4. Ergebnis der JS Aggregation und Komprimierung 32
5. Ergebnis der CSS Aggregation und Komprimierung 33
6. Ergebnis der Aktivierung des Drupal Boost Moduls 33
7. Ergebnis der verlustfreien Kompression der Bilder 35
8. Ergebnis der verlustbehafteten Kompression der Bilder 35
9. Ergebnis der Fehlerbereinigung des Theme Moduls 36
10. Ergebnis des Spritings der Theme Bilder 37
11. Ergebnis der Deaktivierung der Partnerslideshow 37
12. Ergebnis der Deaktivierung des Facebook-Widgets 38
13. Ergebnis der Deaktivierung der Jobleiste 38
14. Ergebnis der Umprogrammierung der Partner-Slideshow 40
15. Ergebnis der Umprogrammierung des Facebook-Widgets 42
16. Ergebnis der Umprogrammierung des Partneranzeigen-Moduls 44
17. Auswertung der Ergebnisse 47

Literatur

[AdWords 2008] AdWords, Inside: *Landing page load time will soon be incorporated into Quality Score - Inside AdWords.* http://adwords.blogspot.com/2008/03/landing-page-load-time-will-soon-be.html. 2008. – URL `http://adwords.blogspot.com/2008/03/landing-page-load-time-will-soon-be.html`

[Bixby 2011] Bixby, Joshua: *FAQs: The 12 most-asked questions about how Google factors page speed into its search rankings — Web Performance Today.* http://www.webperformancetoday.com/2011/08/05/faqs-google-seo-search-ranking-website-speed/. 2011. – URL `http://www.webperformancetoday.com/2011/08/05/faqs-google-seo-search-ranking-website-speed/`

[BMWi 2011] BMWi: *Breitbandportal des BMWi - Kartendownload.* http://www.zukunft-breitband.de/BBA/Navigation/Breitbandatlas/laenderkarten.html? 2011. – URL `http://www.zukunft-breitband.de/BBA/Navigation/Breitbandatlas/laenderkarten.html?`

[Fielding 1999] Fielding, R.: *Hypertext Transfer Protocol – HTTP/1.1.* http://www.ietf.org/rfc/rfc2616.txt. 1999. – URL `http://www.ietf.org/rfc/rfc2616.txt`

[Frucci 2009] Frucci, Adam: *Internet Speeds and Costs Around the World, Shown Visually.* http://gizmodo.com/5390014/internet-speeds-and-costs-around-the-world-shown-visually. 2009. – URL `http://gizmodo.com/5390014/internet-speeds-and-costs-around-the-world-shown-visually`

[Google 2011a] Google: *Let's make the web faster - Google Code.* http://code.google.com/intl/de/speed/. 2011. – URL `http://code.google.com/intl/de/speed/`

[Google 2011b] Google: *Let's make the web faster - Google Code.* http://code.google.com/intl/de/speed/articles/web-metrics.html. 2011. – URL `http://code.google.com/intl/de/speed/articles/web-metrics.html`

[Hitwise 2011] HITWISE: *Hitwise United States » - the power of competitive intelligence*. http://www.hitwise.com/us/datacenter/main/dashboard-23984.html. 2011. – URL `http://www.hitwise.com/us/datacenter/main/dashboard-23984.html`

[Kaspersky 2011] KASPERSKY: *„Das haben wir nicht bestellt“: Miner-Botnetz attackiert deutsche Pizza-Lieferdienste*. http://www.kaspersky.com/de/news?id=207566473. 8 2011. – URL `http://www.kaspersky.com/de/news?id=207566473`

[Klukas 2011] KLUKAS, Jörg: *Unternehmen | pludoni GmbH - the community experts*. http://www.pludoni.de/unternehmen. Juni 2011. – URL `http://www.pludoni.de/unternehmen`

[Microsoft 2011] MICROSOFT: *IE 6 Warning | Educate Others to Stop Using IE6 | IE6 Countdown*. http://www.ie6countdown.com/educate-others.aspx. 2011. – URL `http://www.ie6countdown.com/educate-others.aspx`

[Netcraft 2011] NETCRAFT: *May 2011 Web Server Survey | Netcraft*. http://news.netcraft.com/archives/2011/05/02/may-2011-web-server-survey.html. 5 2011. – URL `http://news.netcraft.com/archives/2011/05/02/may-2011-web-server-survey.html`

[website optimization 2008] OPTIMIZATION website: *The Psychology of Web Performance - how slow response times affect user psychology*. http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/. 5 2008. – URL `http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/`

[PHP 2011a] PHP: *PHP: Was ist PHP? - Manual*. http://www.php.net/manual/de/intro-whatis.php. 2011. – URL `http://www.php.net/manual/de/intro-whatis.php`

[PHP 2011b] PHP: *PHP: Was kann PHP? - Manual*. http://www.php.net/manual/de/intro-whatcando.php. 2011. – URL `http://www.php.net/manual/de/intro-whatcando.php`

[StatCounter 2011] STATCOUNTER: *Top 12 Browser Versions from Aug 10 to Aug 11 | StatCounter Global Stats*. http://gs.statcounter.com/#browser_version-ww-monthly-201008-201108. 8 2011. – URL `http://gs.statcounter.com/#browser_version-ww-monthly-201008-201108`

[Toll 2011] Toll, William: *Why Marketers Must Care About Site Speed*. http://searchenginewatch.com/article/2085970/Why-Marketers-Must-Care-About-Site-Speed. 7 2011. – URL `http://searchenginewatch.com/article/2085970/Why-Marketers-Must-Care-About-Site-Speed`

[w3c 2005] w3c: *W3C Document Object Model*. http://www.w3.org/DOM/#what. 1 2005. – URL `http://www.w3.org/DOM/#what`

[Witzmann 2011] Witzmann, Jan: *Existenzbedrohung DDoS-Attacke: Kleine Onlinehändler werden zur Zielscheibe für Erpressungsversuche | Onlinehändler-News*. http://www.onlinehaendler-news.de/2011/08/11/ddos-attacke-kleine-onlinehandler-werden-bedroht/. 8 2011. – URL `http://www.onlinehaendler-news.de/2011/08/11/ddos-attacke-kleine-onlinehandler-werden-bedroht/`

[YahooDevNetwork 2011] YahooDevNetwork: *Best Practices for Speeding Up Your Web Site*. http://developer.yahoo.com/performance/rules.html#min_dom. 8 2011. – URL `http://developer.yahoo.com/performance/rules.html#min_dom`

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig, unter Angabe aller Zitate und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Dresden, den

Martin Schneider, HTW Dresden