Themengebiet

Websites werden immer umfangreicher, interaktiver und die Benutzerzahlen steigen stetig. Studien von großen Internetunternehmen wie Amazon und Google haben vor Jahren bereits gezeigt, dass dabei die Performance – Lade- und Reaktionszeiten – geschäftskritisch sind und einen direkten Einfluss auf Umsatz und Gewinn eines Unternehmens haben (Amazon 2006) (Amazon & Google 2009). In weiterer Folge bestimmt es auch die Reputation bei Benutzern über eine Website. Benutzer die einmalig ein schlechtes Erlebnis mit einer Website hatten, vergessen diese nicht so schnell und erzählen über diese Erfahrung gerne auch Familie und Bekannten (Schröter 2011) (Daspet 2012). Wie müsste jedoch eine Website im Jahr 2013 umgesetzt werden um allen Anforderungen gerecht zu werden? Global mit so wenig Verzögerung / Latenz zur Verfügung stehen wie nur möglich. Optimierung für verschiedene Arten von Netzwerken (von Glasfaser bis UMTS) und unzählige Geräte mit verschiedenen Displayarten und -auflösungen. Es sollen Techniken eruiert werden mit denen eine WWW-Software all diesen Anforderungen gerecht wird, die positiven Effekte sollen anhand von verfügbaren oder selbst erstellten Tools messbar sein. Die Erkenntnisse der Arbeit fließen direkt in das Masterprojekt MovLib ein, eine internationale und offene Filmdatenbank-Website, die mit dem „IMDb-Killer"-Anspruch entwickelt wird. Eine hohe Performance gepaart mit Kompatibilität ist somit unabdingbar.

Publikationen

Es existieren kaum bis gar keine wissenschaftlichen Abhandlungen zum Thema Performance im Web. Der Grund dafür dürfte die schnelle Entwicklung in diesem Sektor sein und der extrem hohe Praxisbezug der geleisteten Arbeiten. Im Kontrast zu den meisten Berichten die sich theoretisch mit einem Thema befassen.

Client-Side Performance Optimizations

Die Semesterarbeit „*Client-Side Performance Optimizations*" von Jakob Schröter (Schröter 2011) ist wahrscheinlich keine echte Publikation wie verlangt von der Aufgabenstellung, da es sich um die Arbeit eines einzelnen Studenten handelt und nicht dem klassischen Stil eines Berichtes folgt. Trotzdem habe ich mich dazu entschieden dieses Dokument hier mit aufzunehmen, da es als einzige Arbeit die ich ausfindig machen konnte das Thema der (Frontend-)Performance von Websites behandelt.

Der Autor erläutert zu Beginn weshalb die Performance einer Website kritisch für den Erfolg eines Internetprojektes ist und leitet direkt über in eine kurze Erläuterung welche Rolle dabei die Benutzerseite spielt. Auf Seite 5 zeigt er eindrucksvoll, dass über 80 % aller Berechnungen zur Darstellung einer Webseite lokal beim Benutzer geschieht (dies inkludiert das Herunterladen der einzelnen Ressourcen die die Webseite ausmachen). Im Dritten Kapitel erläutert er verfügbare Tools, die es Web-Entwicklern ermöglichen die eigenen Websites zu messen und somit zu verbessern. Kapitel 4 beschäftigt sich mit den „Best Parctices" die über die Jahre von Spezialisten und Unternehmen zusammengesammelt wurden. Abschließend wird von ihm noch darauf hingewiesen, dass die Optimierungen in den Entwicklungsprozess eingebettet sein müssen und die Entwicklung nicht erschweren dürfen, gefolgt von einer Zusammenfassung und einem Blick in die Zukunft.

Automated Construction of JavaScript Benchmarks

Im Bericht „*Automated Construction of JavaScript Benchmarks*" (Richards, et al. 2011) erläutern die Autoren zu Beginn kurz weshalb sogenannte synthetische Benchmarks zum Treffen von Aussagen über reale JavaScript-Anwendungen unangebracht sind. Sie stellen eine neue Software vor, die sie speziell zum benchmarking von realen JavaScript-Anwendungen erarbeitet haben vor, die Software hört auf den Namen „JSBench".[1] Das Erste Kapitel befasst sich damit was JavaScript eigentlich ist, wozu es eingesetzt wird und weshalb synthetische Benchmarks überhaupt existieren und weshalb sie unnütz für reale Anwendungen sind. Des Weiteren erklären sie ihren Ansatz und vorhergehende Arbeiten (von sich selbst und anderen Autoren). Im nächsten Kapitel werden sehr kurz andere Tools vorgestellt die demselben Prinzip folgen. Kapitel 3 widmet sich ganz der detaillierten Erläuterung des „Record/Replace"-Prinzips das im JSBench anwendung findet. Diesem Kapitel folgt eine Einführung in JSBench und wie damit Benchmarks erstellt werden können. Im Fünften Kapitel werden empirische Messungen

[1] http://code.google.com/p/jsbench/

mit JSBench und verschiedenen Browsern gesammelt, ausgewertet und genau erläutert. Das letzte Kapitel enthält eine Zusammenfassung des Berichtes.

JSMeter: Characterizing Real-World Behavior of JavaScript Programs

Im Bericht „*JSMeter: Characterizing Real-World Behavior of JavaScript Programs*“ (Ratanaworabhan, Livshits, et al., JSMeter: Characterizing Real-World Behavior of JavaScript Programs 2009) – der von Microsoft Research stammt – wird an das von mir zuvor ausgewählte Paper angeschlossen (bzw. chronologisch gesehen geht es dem vorherigen Bericht sogar voraus). Die Autoren untersuchen auch hier wie sich reale JavaScript-Anwendungen im Vergleich zu bis dahin bekannten und sehr gut untersuchten Programmen die mit Java, C# oder anderen bekannten Programmiersprachen verfasst wurden verhält. Erneut wird dabei auch auf synthetische Benchmarks eingegangen und der Fakt, dass diese ungeeignet sind um reale JavaScript-Anwendungen widerzuspiegeln wird in diesem Bericht zum ersten Mal klar festgemacht. In der Einführung wird der Stand der Dinge und ein paar wenige Fakten kurz zusammengefasst. Kapitel 2 dient dazu dem Leser Hintergrundwissen über die Art und Weise wie JavaScript funktioniert näher zu bringen. Hier gehen die Autoren auch darauf ein, wie sie ihre Messungen durchgeführt haben. Das nächste Kapitel schließt direkt daran an und beschreibt die Aufbauten der Experimente und was genau gemessen wurde. Das Vierte Kapitel enthält eine Zusammenfassung der Messergebnisse und Erläuterungen zu diesen. Im drittletzten Kapitel werden Implikationen die sich aus den Messergebnissen für reale JavaScript-Anwendungen ergeben unter die Lupe genommen, gefolgt von einer Auflistung ähnlicher Arbeiten in Kapitel 6 und einer finalen Zusammenfassung im Siebten Kapitel.

JSMeter: Measuring JavaScript Behavior in the Wild

Im Bericht „JSMeter: Measuring JavaScript Behavior in the Wild“ (Ratanaworabhan, Livshits, et al., JSMeter: Measuring JavaScript Behavior in the Wild 2010) – erneut von Microsoft Research und denselben Autoren wie der Bericht der zuvor vorgestellt wurde – wird an das zuvor vorgestellte Paper direkt angeschlossen und die Autoren spezialisieren sich viel stärker in dem was sie untersuchen. Es wurden 11 populäre Websites ausgewählt und deren JavaScript wurde mit speziellem Blick auf „Funktionen und Code“ sowie „Events und Handlers“ untersucht und gemessen. Die Funktionen werden von den Autoren des Weiteren im Hinblick auf „hot code“ (Codefragmente die ständig ausgeführt werden) und „cold code“ (Codefragmente die selten ausgeführt werden) unterteilt und näher untersucht.

JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications

Beim Bericht „*JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications*“ (Ratanaworabhan, Livshits and Zorn, JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications 2010) – erneut von Microsoft Research und denselben Autoren wir bei den beiden vorhergegangenen Berichten – handelt es sich eigentlich um ein Update des zuvor vorgestellten Papers. Ich werde hier nicht erneut auf den Inhalt eingehen, vielmehr darauf warum ich zweimal das mehr oder weniger selbe Paper hier mit hinein nehme. Diese neue Fassung wurde stark überarbeitet mit Hilfe von sehr vielen Rückmeldungen aus der Community (eine Auflistung der namhaftesten Helfer findest sich am Ende des Berichtes). Dieses Update zeigt, wie komplex die Messung von Performance sein kein, im Besonderen im Hinblick auf reale Anwendungen. Mit ein Grund weshalb der Großteil von Performance Diskussionen und Benchmarks in großen Communities besprochen und erforscht wird und nicht von Einzelpersonen.

You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions

Im Bericht „*You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions*“ (Nikiforakis, et al. 2012) geht es nicht direct um Performance, doch um ein Thema das im Zusammenhang mit Performance sehr wichtig ist, das inkludieren von JavaScript das auf anderen Domains gehostet wird. Am bekanntesten dürften hier die

„Google Hosted Libraries“[2] sein. Im Paper haben die Autoren 10.000 der bekanntesten Websites der Welt untersucht (laut Alexa). Die Autoren untersuchten zuerst wie üblich es ist JavaScript von externen Domains zu inkludieren, gefolgt von einer empirischen Evaluation der Qualität der Wartung der Bibliotheken die von Anbietern angeboten werden. Des Weiteren eruieren die Autoren Möglichkeiten wie die Inkludierung für Attacken ausgenutzt werden können.

Dynamic Object Offloading in Web Services

Im Bericht „*Dynamic Object Offloading in Web Services*“ (Zagarese, Canfora and Zimeo 2011) versuchen die Autoren mit speziellem Hinblick auf Serviceorientierte Architekturen (SOA) einen Weg zu finden wie die Performanceprobleme von Web Services durch eine intelligente, selbst lernende Middleware (Vermittlungssoftware). Dies geschieht anhand von Serialisierung von Objektattributen. Durch eine selbst lernende Technik (die hier noch auf Cache hits / misses basiert, jedoch in der Zukunft laut Autoren ausgebaut werden soll) entscheidet der Service welche Attribute gleich (*eager*) serialisiert werden können und welche später (*lazy*). Solche Techniken können auch auf herkömmliche Websites umgelegt werden.

SEDA: An Architecture for Well-Conditioned, Scalable Internet Services

Im Bericht „*SEDA: An Architecture for Well-Conditioned, Scalable Internet Services*” (Welsh, Culler and Brewer 2001) von Intel Research stellen die Autoren eine neue Architektur für hoch nebenläufige Internetservices vor. Die vorgestellte neue Architektur nennen die Autoren *staged event-driven architecture* (*SEDA*), die für massiv nebenläufige Anforderungen gedacht ist und die Erstellung von Services für diese Aufgabenstellung simpler gestalten soll. Jede Applikation innerhalb der SED-Architektur setzt sich aus einem Netzwerk von eventbasierten *Stages* die durch explizite *Queues* verbunden sind zusammen. Die Autoren implementieren einen hoch performanten HTTP-Server und einen Packet-Router für das Gnutella Peer-to-Peer-Netzwerk. Die Autoren wollen damit beweisen, dass ihre SED-Architektur performante und robuste nebenläufige Applikationsentwicklung ermöglicht.

Performance impact of deploying HTTPS

„*Performance impact of deploying HTTPS*“ (Kleppe 2011) ist ein weiterer Bericht der die zwei Schlüsselfaktoren Performance und Sicherheit zusammenbringt und untersucht. Die Frage die der Autor im Laufe seiner Arbeit untersucht ist simpel, welchen Einfluss hat der Einsatz von TLS/SSL auf die Performance von gängig eingesetzten Webservern (Apache, IIS, lighttpd und nginx). Im späteren Verlauf des Papers wird der IIS von Microsoft von den Tests ausgeschlossen, da er signifikant schlechter arbeitet als die anderen Server und der Autor somit weitere Messungen für irrelevant hält. Der Autor untersucht dann die Antwortzeiten der verschiedenen Modi (Klartext / HTTP, TLS / SSL / HTTPS und 0x33; letzteres wird von lighttpd nicht unterstützt), CPU-Belastung und Speichernutzung.

Performance analysis of VP8 image and video compression based on subjective evaluations

Im Konferenzbericht „*Performance analysis of VP8 image and video compression based on subjective evaluations*“ (De Simone, et al. 2011) wird der noch junge Kompressionsalgorithmus VP8 subjektiv unter die Lupe genommen. Der Algorithmus wurde von On2 Technologies entwickelt, das Unternehmen wurde später von Google aufgekauft (Zota 2010) und die Algorithmen wurden in Form von WebP und WebM als Webstandards von Google vorgeschlagen. Beide Formate (also WebP für Bilder und WebM für Videos) werden von den Autoren untersucht. Die neuen Formate von Google müssen dabei gegen etablierte (Teils sehr alte) Kompressionsverfahren antreten. Für die Auswertung wurden jeweils, für Bild und Video getrennt, 18 Probanden gefunden (unterschiedliche Geschlechterverteilung und beides Mal ein Durchschnittsalter von 24 Jahren). Die Autoren beschreiben genau wie die Tests durchgeführt werden und die Ergebnisse werden detailliert aufgeschlüsselt.

[2] https://developers.google.com/speed/libraries/

Using caching and optimization techniques to improve performance of the Ensembl website

„*Using caching and optimization techniques to improve performance of the Ensembl website*“ (Parker, et al. 2010) ist der letzte Bericht den ich hier inkludiere, somit komme ich auf 11 Papers insgesamt, nachdem der dritte JSMeter-Bericht jedoch nur ein Update für den Zweiten darstellt habe ich mich dazu entschieden ein weiteres Paper zu inkludieren. Die Autoren beschreiben detailliert wie sie den Ist-Stand der Ensembl Website untersuchten und anhand welcher Tools und Ressourcen sie die bestehende Website analysierten. Gefolgt von den umgesetzten Maßnahmen die unternommen wurden um die Performance zu steigern und den Systemhunger zu verringern. Persönlich gefällt mir dieses Paper sehr gut, da hier ein gesamter Optimierungsprozess eines bestehenden Webprojektes nachvollzogen werden kann.

Literaturverzeichnis

Amazon & Google. "The User and Business Impact of Server Delays, Additional Bytes, and HTTP Chunking in Web Search." *Präsentation.* 2009. http://velocityconf.com/velocity2009/public/schedule/detail/8523.

Amazon. "Make Data Useful." *Fallstudie.* 2006. http://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-28.ppt?attredirects=0.

Daspet, Éric. *Why bother with performance metrics when you can cheat?* Dezember 12, 2012. http://calendar.perfplanet.com/2012/why-bother-with-performance-metrics-when-you-can-cheat/.

Kleppe, Harald. *Performance impact of deploying HTTPS.* Amsterdam: University of Amsterdam, 2011.

Nikiforakis, Nick, et al. *You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions.* Raleigh: ACM, 2012.

Ratanaworabhan, Paruj, Benjamin Livshits, and Benjamin Zorn. *JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications.* Technical Report, Redmond: Microsoft Research, 2010.

Ratanaworabhan, Paruj, Benjamin Livshits, David Simmons, and Benjamin Zorn. *JSMeter: Characterizing Real-World Behavior of JavaScript Programs.* Technical Report, Redmond: Microsoft Research, 2009.

Ratanaworabhan, Paruj, Benjamin Livshits, David Simmons, and Benjamin Zorn. *JSMeter: Measuring JavaScript Behavior in the Wild.* Technical Report, Redmond: Microsoft Research, 2010.

Richards, Gregor, Andreas Gal, Brendan Eich, and Jan Vitek. *Automated Construction of JavaScript Benchmarks.* Artikel, New York: ACM (Purdue University & Mozilla Foundation), 2011.

Schröter, Jakob. *Client-Side Performance Optimizations.* Semesterarbeit, Stuttgart: Stuttgart Media University, 2011.

Welsh, Matt, David Culler, and Eric Brewer. *SEDA: An Architecture for Well-Conditioned, Scalable Internet Services.* Technical Report, Berkeley: Intel Research, 2001.

Zagarese, Quirino, Gerardo Canfora, and Eugenio Zimeo. *Dynamic Object Offloading in Web Services.* Benevento: University of Sannio, 2011.