JOHANNES KEPLER
UNIVERSITY LINZ | JKU

TNF

Faculty of Engineering
and Natural Sciences

Web Performance Monitoring and Tuning

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements
for the academic degree

Bachelor of Science

in the Bachelors Program

Computer Science

Submitted By:
Jürgen Ratzenböck

At the:
Institut für Telekooperation

Advisor:
Gabriele Anderst-Kotsis

Linz, 20. November, 2015

Affidavit

I hereby declare that the following dissertation "Web Performance Monitoring and Tuning" has been written only by the undersigned and without any assistance from third parties.

Furthermore, I confirm that no sources have been used in the preparation of this thesis other than those indicated in the thesis itself.

Linz, on January 4, 2016

Jürgen Ratzenböck

Acknowledgment

Summary

Summary ...

Abstract

Abstract ...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 1990 the first website was published to the Internet and became available to people worldwide who already had access to the Internet at these times. This amount of people was very limited in the early years and websites were just simple HTML files with static content maybe containing even some pictures and links. Over the years the Web has evolved dramatically and today it's nearly inevitable for successful businesses around the world to survive without it. Companies present themselves with websites, sell their products and provide services for their customers online.

A good web experience nowadays is crucial and the speed of a website often plays a key role whether an interested visitor becomes a customer or not. Therefore it's a factor a successful company should not neglect when developing and maintaining a website. This thesis documents the detailed investigation of the performance of a business website and outlines possibilities to improve the perceived web experience based on the Upper Austrian pet tracking company Tractive GmbH.

1.1 Motivation

Throughout all the innovative developments in telecommunication the web browser has become the most widespread deployment platform to developers. Industry growth projections predict an amount of 20 billion connected devices by 2020 and each device comes with a browser installed. In contrast to all different applications which are somehow tailored to fulfil some defined tasks for a given targeted audience, the browser does not have any limitations disregarded of the type of the device, used platform, manufacturer etc. Due to this importance the browser from the early days looks nearly nothing like the one we use nowadays on our devices. Concurrently with these technical innovations the variety and complexity of tasks which can be done via the Web has increased as well and the user expectations on web applications are very high. This

is why web performance definitely has to be considered a feature next to design for instance. Empirical studies showed that users tend to drop away from slow websites whereas faster websites lead to

- better user engagement
- better user retention
- higher conversion rates (indicates how many of the website visitors convert the paying customers)

Single page applications, responsive designs, cool CSS3 animations or a bunch of new Web UI components featured by HTML5 are only a few things which come along with modern web applications as they appear in our todays web browsers. All these innovations add complexity and therefore more data to be handled throughout the network. This introduces a further challenge for web engineers as all this design tweaks require more work to deliver high performance web applications to their users. Understanding how the web with it's different underlying protocols works is essential. Good developers have to have some deeper knowledge of how the bits within a HTTP request are transported from one point to another as well as how the browser on the client side renders HTML, loads media and processes JavaScript. [3]

1.2 What is Tractive?

Tractive GmbH produces hardware devices for pets as well as develops software applications which allow users to communicate with their pets wearing these devices. The company was founded in Pasching, Upper Austria in 2012 and offers today several different devices whereby the origin goal was to find and locate pets fitted with a GPS tracker using a Smartphone App. In addition to the hardware devices and mobile apps Tractive develops and maintains a Website providing all the necessary product information for potential customers as well as a web shop which allows customers to purchase products and a web application where registered users can manage their trackers and pets.

This thesis will focus on the company's main Website containing all de-facto static content as well as the mentioned Pet Manager. During this thesis these parts are redesigned as well as integrated and connected to the current backend containing the data and business services used by client applications. Next to a good "look and feel" it was important for the company to evaluate how well the current website performs

and try to find bottlenecks which should be considered and fixed when integrating the new Website.

Chapter 2

Web Performance Metrics

Before any measurement or even improvement regarding a website's performance can be made, it's important to understand what actually are the reasons for the delay until a user can see the rendered webpage in her browser. Since a website is not like an

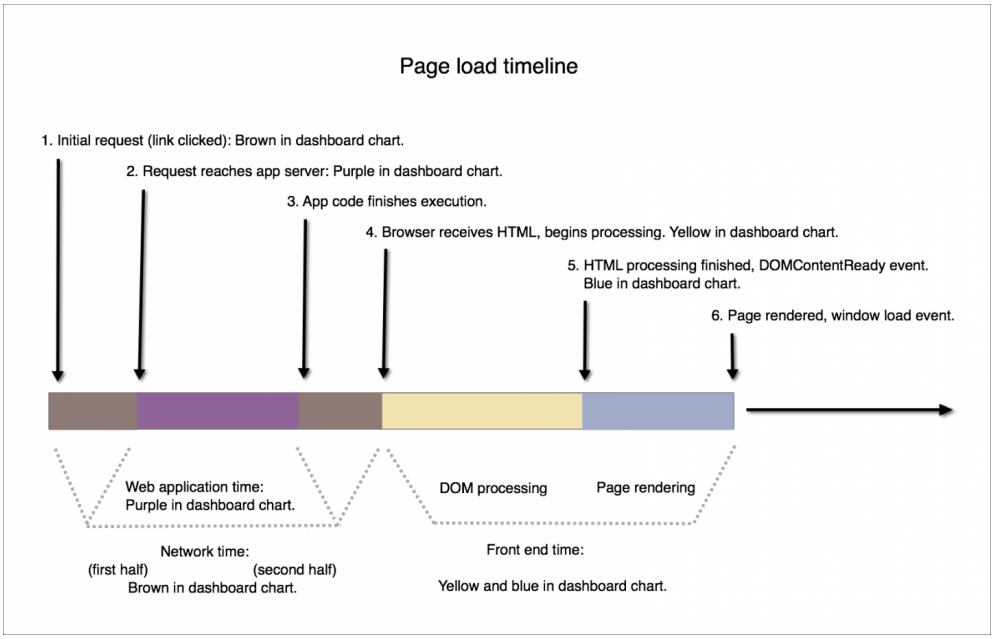

Figure 2.1: Timeline of a page load [5]

executable started at the computer of a client, there are several indicators affecting the overall performance of a website at which I want to look at in more detail now.

2.1 Latency

Latency basically is the time it takes a message to travel from it's point of origin to the destination. Although this sounds very simple, it's clear that not every source in the network is directly connected to a destination point. The global network infrastructure is much more complex and therefore several components contribute to the overall latency. In essence latency can be grouped into:

- Propagation delay
- Transmission delay
- Processing delay
- Queuing delay

Propagation delay is the time required for a network packet to travel from sender to receiver. It simply can be computed as a function of distance over speed and is heavily based on the speed of light, the maximum speed at which information can travel. Fortunately the speed of light is extremely fast and even including a small loss of speed due to refraction of the medium through which the message travels it only takes a few hundred milliseconds to cross the distance between two continents. There have been a lot of improvements regarding the medium connecting network components and with fiber-optic cables a speed of approximately 200.000m/s can be achieved nowadays. [3]

Route	Distance	Time (vaacum)	Time (fiber)
New York to San Francisco	4.148km	14ms	21ms
New York to London	5585km	19ms	28ms
New York to Sydney	15993km	53ms	80ms
Equatorial circumference	40075km	133.7ms	200ms

Table 2.1: Examples for propagation delays [3]

Transmission delay is the time it takes to put the packet to be sent onto the data link. [3] The bytes to be sent over the network are not automatically on the link and therefore there must be this delay. Due to the small size of request and response packets as they are used for websites, this amount of time usually stays within a few milliseconds. It can also be described as a function of packet length divided by data rate of the link.

The latencies presented the table above assume a direct connection between two locations. In practice this does not hold since a packet gets handled by several intermediate routers until it reaches the destination. Each router must accept the incoming packet first, look at the header information of it and decide where to send the packet next. The time required for these operations is called **Processing delay**. This delay can vary from packet to packet and strongly depends on the number of hops a packet does on it's way. [4] Nearly all of the operations are done in the hardware and cost less time. [3]

Due to everyday's enormous amount of traffic the network can be unpredictable under some circumstances. If the traffic load on routers is high some packets have to be put into an incoming buffer and will be queued before they can be processed. This is known as **Queuing delay** and can add some additional time to overall latency. The concerned routers on the way usually vary in latency and bandwidth as well as the traffic along the path which makes it difficult to predict occurrence of queuing.

The sum of these delays results in the total network latency and under normal circumstances does not exceed about 300ms, even when crossing oceans and continents. [3] Although these delays don't seem to introduce any kind of problem, especially when we consider such extraordinarily large distances like in the example, the parameter latency is critical and can introduce a performance bottleneck. This will get clearer when looking at the involved protocols by a request.

2.2 Throughput

Throughput is the number of items processed per unit time, such as operations per minute or instructions per second. When talking about network connections the term Bandwidth is usually used instead, which is equal to the throughput in bits per second. Regarding bandwidth there hast be differentiated between the core network and the edge points.

Bandwidth in core networks is the bit rate per seconds for links across long distances, like subsea connections between continents. Different cable mediums have been researched over years whereby with nowadays capabilities of fiber-optic connections, amazing rates of 70Tb/s for a single fiber-optic link could be reached.

On the edges of the network things look completely different. **Bandwidth at edge points** is the data rate we know from our ISP and depends heavily at the technology

used. No matter whether it is dial-up, DSL or fiber-optic to the home these numbers are very small compared to the core network bandwidth. [3]

Bandwidth is not directly connected to Latency, which means that a low latency not necessarily implicates high bandwidth of the data link. However if we take the bandwidth at the network edge, an increase of maximum throughput often leads to lower latency since more packets can be processed in the same time and collisions and congestion will more likely be avoided. If the problem occurs at some intermediate hops on the path, higher bandwidth does not have any positive effect on latency. [4]

2.3 Fundamental Protocols

We have covered two important metrics when it comes to deliver bytes from a client to the server and vice versa. To understand why these metrics can have great impact on the overall web performance, we take a closer look on the fundamental internet protocols involved when requesting a web page.

2.3.1 HTTP (Hypertext Transfer Protocol)

HTTP is part of layer 7 in the OSI model and comes into play when the browser talks to the server about it's capabilities and requirements whereas the server tells the browser how to treat content appropriately. As soon as address information and an underlying TCP connection is available, the browser sends an HTTP request using this connection by specifying the method and relevant information packed within a so called HTTP header and body. The basic request methods supported by every browser are GET and POST whereby GET allows to send custom parameters (data concerning the purpose of the application) within the header and POST within it's body. Furthermore the browser can add different properties to tell which version it supports, whether compression and caching can be used or TCP connections should be kept open. Adapting these properties in your web server's configuration regarding to your web page can help you to optimize performance remarkably. [1]

2.3.2 TCP (Transmission Control Protocol)

TCP is the mostly used transport protocol and the basis for nearly all of the HTTP traffic delivered on the Internet. Due to it's many features it supports out of the box it's

a rich protocol as well as an essential performance factor. Although TCP is pretty low-level and web developers most likely won't face it directly, several configurations done in the application layer, in essence in HTTP, influence the performance of TCP and the underlying network. To understand why TCP is very sensitive regarding latency we have to take a closer look on the basic techniques behind it.

TCP connections start with a three-way handshake between client and server because they must agree on connection specific variables. First the sender sends a SYN packet which is acknowledged by the receiver with a SYN ACK and only directly after the ACK packet from the client application data can be sent. This procedure obviously requires a full roundtrip between client and server before any data is sent. In it's simplest and initial form every HTTP request opens a new TCP connection which introduces time loss where no application data can be transferred.

Since TCP promises that packets will be delivered in-order, in-time and reliable it contains special algorithms, called flow control and congestion control respectively avoidance to handle bandwidth differences and network condition varieties appropriately. TCP regulates the speed at which packets are transferred according to the capabilities of sender and receiver. After an ACK each side advertises it's new receiving window indicating how many bytes it is able to receive at once by not being overwhelmed from the other side. With a mechanism called "Slow Start" TCP prevents both sides from overwhelming the underlying network. It therefore introduces a conservative congestion window which grows exponentially after each ACK packet to increase the amount of bytes over time until first packets get lost. Then the congestion avoidance intervenes and resets the window to a smaller size. The minimum between the receiving window and the congestion window indicates the amount of packet segments to be sent in one turn. Let's examine an example of how this slow start can look like in practice if we want to transfer 64KB with an initial congestion window of 4 segments between New York and Sydney (RTT = 160ms).

$$\frac{65535bytes}{1460bytes} = 45segments$$

$$160ms * \log_2(\frac{45}{4}) = 559ms$$

As we can see from the results it takes about 560ms to reach the throughput of 64KB. This is more than three roundtrips between New York and Sydney plus the time required for the three-way handshake the resulting latency is quite a lot. TCP internally negotiates and adjusts the size of the window according to available bandwidth, RTT and network conditions to optimize throughput and prevent gaps where one side has to wait for an ACK of unacknowledged data before it can send new data. All in all we

can see that having an eye on the internals of TCP can make a big difference as we have seen. [3]

2.4 Utilization and Efficiency

These two performance parameters give a more general indicator of the system's components performance. **Utilization** is the fraction of capacity of a component used. [4] This gives a good hint how busy a server component currently is and how much load it can handle by operating still normal. There are a lot of common utilization performance metrics which can be measured. The most popular ones are

- CPU
- Disk
- Memory

Although these metrics are not directly connected to the performance a user experiences when she waits for a web page to be visible in the browser, it helps to check whether infrastructure components operate as expected. The reason for bad performance is not always based on poor latency or throughput, sometimes whole components does not scale as desired and this can have negative impact on what happens on the server side.

Efficiency is defined as throughput divided by utilization. Efficiency cannot be measured but can reveal how much components can withstand. This metric can further have some influence in economic decisions when deciding about buying resources. In this case it is more likely known as cost-effectiveness. [4]

2.5 Processing Time on Server

This is the time between the arrival of the request on the server and the point where the server starts to send the response. For the performance of a web application it is an important metric as it gives information how fast the server fetches and builds the web resource requested. Since modern websites don't contain only pure HTML code usually much more work has to be done to generate the outgoing response file. This includes algorithms to be executed, working with databases in the backend, pre-processing CSS

and Javascript extensible languages, loading of images, generating HTML and so forth. Interactive web applications which work with bigger datasets can lack of performance pretty fast if you don't pay attention on using appropriate software technologies or doing cost-intensive work to fulfil interactive tasks of a user.

2.6 Browser Processing

When the first bytes have arrived at the client it's time for the browser to start it's work. Obviously it is important for good user experience that the time span between a request and the arrival of the first byte should be as minimal as possible because during this time nothing can happen inside the browser. Similar to the networking and processing time there is potential to save time and improve user experience when making the website visible.

2.6.1 DOM Processing

The first step the browser has to do is to identify the fetched resource from the server, where in it's simplest form it is HTML content. The received characters are disassembled into tokens and the DOM (Document Object Model) is constructed. The DOM is required to render any element on the screen and can be blocked by some other resources and therefore delayed. The same happens with CSS applied on HTML elements. For this purpose a so called CSSOM (CSS Object Model) gets constructed and afterwards combined with the DOM to a render tree ready for painting. Due to the reason that a website is effectively unusable without any style, CSS is basically treated as a render blocking resource and interrupts processing of other content. Another blocking resource is Javascript no matter if placed inline or within an external file. DOM construction will be interrupted if Javascript has to be executed. Processing the final DOM for first page view early shortens the time until elements appear on the screen and therefore this time point is a crucial one regarding any performance optimizations during web development. [2]

2.6.2 Page Rendering

After the DOM and CSSOM have been constructed and composed to the render tree the elements can be painted onto the screen. Although much of the work is done automatically by the browser, there are some performance tweaks possible to significantly

improve user experience. As metric which can be directly measured it is usually referred to the final onLoad-Event indicating the completeness of a page load when every element of the webpage is visible on the screen. However, most often it is more important to improve the visual appearing of single elements like text, images, multi-media content etc. There is a huge difference for the user whether one essential page element is visible after 0.3 seconds and the following after 0.6 seconds or both elements appear after 0.6 seconds. Page rendering is an important metric but has to be observed differently since here perceived experience is critical.

Chapter 3

Architecture of the system under test

Due to steady evolvement and increasing requirements the architecture of the system under test has become quite challenging. Physical resources and several software services have been added over the last months. Used software technologies and integration of Tractive's main website and dynamic web application into the whole environment as well as other server components and software tools influence web performance. The following paragraphs give an overview about the infrastructure concerning the web applications and other relevant services.

3.1 Infrastructure

Due to high availability and scalability requirements the main backend services and databases as well as the web application run on dedicated root server instances provided by Hetzner Online GmbH. The hosts are classified into three data-hosts containing our databases (MongoDB, Redis) and three multi-purpose application hosts. Regarding basic performance measurements of the web application the multi-puropse application hosts are relevant. They are optimized for user and hardware traffic and have following technical specifications:

- tractive02 and tractive03: Intel i7-2600 Quad-Core processor, 32GB DDR3-RAM, 2x 3TB HDD (Software RAID1)

- tractive04: Intel Core i7-920 Quad- Core processor, 48GB DDR3-RAM, 2x 2TB HDD (Software RAID1)

For the backend services as well as the current web application we use the full-stack web framework Ruby on Rails which provides many nice features to build a RESTful web application. Therefore the multi-pupose application host is mainly a Rails Application Server. Besides we have other software components which support us to provide a highly available and scalable web application illustrated in the picture below.

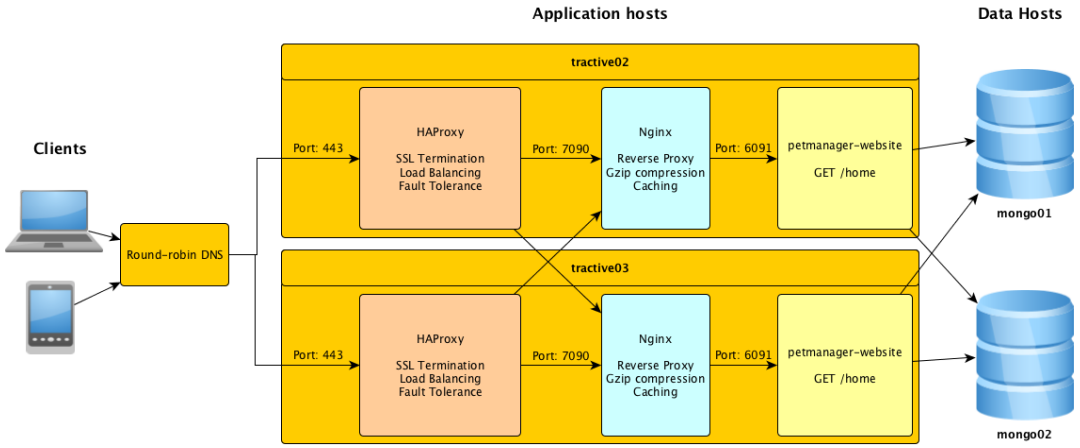

Figure 3.1: Illustration of infrastructure and platform (simplified with only two application and data hosts)

Since DNS can hold multiple IP address records for the same domain name it is possible to redirect incoming requests using a Round-robin DNS technique. This algorithm selects the IP addresses of Tractive's three application in an alternating way. A core component of the infrastructure is the HAProxy (High-Availability Proxy) which is the first receiver after an IP-address selection. Incoming requests independent of the type of the service they concern get distributed equally throughout the available servers. In addition to load balancing the HAProxy does health checks and prevents forwarded requests to unavailable services on an instance. Instead such requests are forwarded to a healthy instance. Furthermore HTTPS requests are terminated appropriately and custom ports (protected from outside) are used from this point on. Behind there is a Nginx web server as a powerful component mainly used for caching content, serving assets, providing compression and forwarding requests to the correct service acting as reverse proxy. Especially for static content it allows to return from here directly back when a cache hit was successful. Otherwise the requests gets proxied by and the according service, which for the web application is the petmanager-website service does it's work. In case user-related data is necessary backend algorithms might be triggered and probably the data hosts are required. The petmanager-website collects the necessary information and generates the requested web page dynamically. So far a quick overview about the overall infrastructure and it's components before I give some insights into the relevant services.

3.2 Relevant Services

Chapter 4

Measurements

Overview... Architecture... General Components

4.1 Comparing HTTP and HTTPS requests

4.2 Measuring with Apache Benchmark

4.3 Measure Browser Processing

Chapter 5

Suggested Improvements

Overview... Architecture... General Components

5.1 Content Delivery Networks (CDN)

5.2 Loading of Assets and Caching

5.3 Compression

5.4 Language and Redirection

5.5 Connection Reusing

Chapter 6

Conclusions and Future Work

Bibliography

[1] Patrick Chang. Fundamentals of http.

[2] Ilya Grigorik.

[3] Ilya Grigorik. *High-Performance Browser Networking*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, Sep 2013.

[4] Patrick Killelea. *Web Performance Tuning*, volume 2. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, Mar 2002.

[5] New Relic.