Overview

• The Relation Between Revenue and Web Performance
• Web Performance Metrics
• Why Sites Are Slow?
• Website Optimisation
• Web Performance - Analysis Tools
Relation Between Sale and Web Performance

• Businesses of all sizes realising that even modest changes in page loading times can have a significant effect on their profits.

• The move toward a faster web has been driven largely by Yahoo! and Google.
  – Google experimented with reducing the size of its Maps homepage (from 100 KB to 70–80 KB). Within a week, traffic had increased by 10 percent.
  – Google also found that a half-second increase in loading times for search results had led to a 20 percent drop in sales.
Relation Between Sale and Web Performance

- Amazon calculated that a page load slowdown of just one second could cost them $1.6 billion in sales each year.
- Almost 40% of online shoppers abandon a website that takes more than 3 seconds to load.
- The fact that there is a correlation between speed and sales perhaps isn’t too surprising, but the extent to which even a tiny difference in loading times can have such a noticeable impact on sales certainly is.
- Work at Stanford University suggests that slow websites are also considered less credible!
- In 2010, Google announced that loading times would play a role in how it ranked sites — faster sites will rank higher.
What Is “Too Slow”, and When Do Websites “Feel” Slow?

• The perception of speed is very subjective and very context specific.

  Satisfaction = Perception - Expectation

• Imagine a situation where you visit a page and a loading indicator slowly moves from 5% to 10%. You'll expect it to take a while to hit 100%.

• If the percentage unexpectedly begins to rise quickly to 95% and then 100%, you'll be satisfied and happy, because your perception exceeded your expectation.

  – In a nutshell, website visitors are satisfied when their perception exceeds their expectation, and dissatisfied when the opposite occurs.
Perception

- Perception refers to how fast the user *thinks* your website is, rather than how fast it *actually* is.

- Most of the time, that's almost more important than the actual speed of your website.

- Generally, the perception of something being slow carries negative associations.

  - Example: changing the passengers perceived waiting time to pick up their luggage by *extending the distance* from the arrival gate to the baggage claim point.
Expectations

• In the context of performance, when servicing customers, it's important to manage and care about their expectations.
  – Example: Disney line-ups that show you the expected waiting times, with a rather pessimistic estimate, so that customers get to the front of the line in a much shorter time than predicted.
  – As a result, the customer feels more positive.

• We should set clear expectations by keeping user informed about the progress of their task.

• Show them the content they want to see in the fastest possible way;

• If waiting is required, show progress bars or other indicators to reassure them the website is still responding, and that they'll receive the content that they requested.
Response Time

• Perceived web performance involves how we, as humans, experience and respond to the performance of a system.

- We feel **instant perception** around a **100ms** delay
- A **slight perceptible delay** occurs between **100ms and 300ms**
- We definitely feel a **perceptible delay** under **1000ms (1s)**
- After **1 second**, we feel that a **mental context switch** starts
- After **10 seconds** and more, the **abandon rate** goes up and the user tends to leave the site
Web Performance Optimization (WPO)

• PO drives traffic towards your website
• Improves user experience (UX)
• Increases revenue
• Reduces costs

If you want to achieve a fast experience for your users, you need to understand what aspects could harm such a goal.

• This is especially important when you have to deal with a lot of competitor products and websites.
  – Being faster than your competitors definitely puts you ahead of the curve when somebody is googling for a service that you offer.
User Experience and Performance

- A website's user experience depends on several different factors: design, accessibility, information architecture, usability, perception, and performance.

- Information architecture, design, and performance are all very closely connected.

- Performance should be incorporated at the early design stage, and not added as an afterthought.
UX Principles

• There are several principles you can follow to achieve good user experience.

• First of all, you want to make sure you know and understand the user's goal.

• For example, the user goals of a shopping site such as Amazon, and of a public transport website, are very different
  – Amazon tries to persuade users to hang out for hours on their site
  – The transport Web site’s goal should be - to provide users with the train schedule as quickly as possible.
Web Performance Metrics – Different Views

• Web Performance Metrics:
  – WWW user perception:
    • fast response time
    • no connection refused
  – Web administrators:
    • high throughput
    • high availability

• Need for quantitative measurements:
  – Response time
  – Throughput
Throughput

Example:

The Web site of a travel agency was monitored for 30 minutes and 9,000 HTTP requests were counted.

- We want to assess the server throughput.

3 types of Web objects

- HTML pages: 30% and avg. size of 11,200 bytes
- images: 65% and avg. size of 17,200 bytes
- video clips: 5% and avg. size of 439,000 bytes
Throughput

• in terms of requests:
  – (No. of requests)/(period of time) = 
  9,000/(30 \times 60) = 5 \text{ requests/sec}

• in terms of bits/sec per class:
  – (total requests \times \text{class \%} \times \text{avg. size}) / (period of time)
  – HTML throughput (Kbps)
    • 9,000 \times 0.30 \times (11,200 \times 8) / 1,800 = 131.25
  – Image throughput (Kbps)
    • 9,000 \times 0.65 \times (17,200 \times 8) / 1,800 = 436.72
  – Video throughput (Kbps)
    • 9,000 \times 0.05 \times (439,000 \times 8) / 1,800 = 857.42

– Total throughput
  • 131.25 + 436.72 + 857.42 = 1,425.39 \text{ Kbps}
Response Time

- Example: Virtual car dealership
  - users search and submit purchase requests for new and used cars on the Web.
  - 1,300 affiliated car dealers provide info about vehicles on their parking lots.
  - new agreements will be phased in gradually, increasing the Web server’s traffic by 10%, then 20%, and finally 30% w.r.t. to current load.
Response Time – cont.

- **Web server request types:**
  - retrieve document and images
  - search the DB according to make, model, price range, and distance of dealer from buyer.
  - purchase request.

- **Critical request:** **search**
  - 5% of searches generate a car sale
  - Average sale generates £12,000 in revenues.
  - Service Level Considerations:

<table>
<thead>
<tr>
<th>Response time (in sec)</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 &lt; r ≤ 6</td>
<td>60% lost</td>
</tr>
<tr>
<td>r &gt; 6</td>
<td>95% lost</td>
</tr>
</tbody>
</table>
**Response Time – cont.**

- **Management questions:**
  - will the Web server support the load increase while preserving the response time below 4 sec?
  - if not, at which point will its capacity be saturated?
  - how much money could be lost daily if the Web server saturates when the load increases?

<table>
<thead>
<tr>
<th></th>
<th>Current</th>
<th>Current + 10%</th>
<th>Current+20%</th>
<th>Current+30%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Searches per day</strong></td>
<td>92,448</td>
<td>101,693</td>
<td>110,938</td>
<td>120,182</td>
</tr>
<tr>
<td><strong>Response Time</strong></td>
<td>2.86</td>
<td>3.80</td>
<td>5.67</td>
<td>11.28</td>
</tr>
<tr>
<td><strong>Percent Sales Lost</strong></td>
<td>0</td>
<td>0</td>
<td>60%</td>
<td>95%</td>
</tr>
<tr>
<td><strong>Sales per day</strong></td>
<td>4622</td>
<td>5085</td>
<td>2219</td>
<td>300</td>
</tr>
<tr>
<td><strong>Daily revenue ($1K)</strong></td>
<td>$83,203</td>
<td>$91,524</td>
<td>$39,938</td>
<td>$5,408</td>
</tr>
<tr>
<td><strong>Potential daily revenue ($1K)</strong></td>
<td>$83,203</td>
<td>$91,524</td>
<td>$99,844</td>
<td>$108,164</td>
</tr>
<tr>
<td><strong>Lost daily revenue ($1K)</strong></td>
<td>$ -</td>
<td>$ 0</td>
<td>$ 59,906</td>
<td>$ 102,756</td>
</tr>
</tbody>
</table>
Why Sites Are Slow?

- They weren’t designed with speed in mind
- Priority is given to graphics design and attractive looking interface regardless of size.
- Performance is often low on the programmer’s agenda
  - server-side scripting language (such as PHP or Perl) and back-end database not designed for performance.
- Testing of a new website usually carried out on a development server under low load.
  - Not all clients have high-end hardware and latest browsers
Website Optimisation

• The best approach to optimisation is:
  – First to have a good overview of how the whole system (hardware, operating system, databases, web servers, web browsers and so on) will fit together.

• Then you will be in a better position to judge where the greatest gains can be made.

• Without **effective measurement**, it's hard to determine if something has improved or gotten worse.
## TCP/IP Architecture

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocols/Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Layer</td>
<td>FTP, HTTP, Telnet, SMTP, SSH</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>Transmission Control Protocol (TCP)</td>
</tr>
<tr>
<td>Network layer - Internet Protocol (IP)</td>
<td></td>
</tr>
<tr>
<td>Network Interface Layer</td>
<td></td>
</tr>
<tr>
<td>Physical Layer</td>
<td></td>
</tr>
</tbody>
</table>
Web Components

- Web-based systems consist of multiple components working together:
The Causes of Poor Performance - Backend

• **Backend** – involves DNS Lookup, Server connection, Server response

• The Server response performance – also known as **backend performance**.
  – This involves the server processing the request sent by the browser and in turn sending out the correct website content.

• The **hang-up**: Heavy website traffic, overloading servers, code bugs, inefficient resources.
The Causes of Poor Performance – Network

• **Middle Mile** - Content travels through networks to user

• This stage is known as “middle mile”, where the content travels from the datacentre to the visitor’s browser.

• The hang-up: When packets of information have to travel via inefficient routes and across great distances to arrive at the browser.
The Causes of Poor Performance – Front-end

- **Front-end** – aspects such as: Browser download and Browser rendering times.

- The final stage is “front-end performance”. This stage consists of the visitor’s browser rendering the page’s content once the HTML file has been delivered.

- The hang-up: Downloading and rendering complex website content: numerous assets, heavy assets, and third party assets all result in problems.
Useful Performance Tools

• There are many different tools available for measuring website performance,

• Commercial products like:
  – or Neustar [http://www.neustar.biz/](http://www.neustar.biz/)

• But the good news is that there are loads of free tools as well.
  – Worth trying open source tools such as your **Browser-based Developer Tools** (Hit Ctrl + Shift + I on Windows, and Cmd + Opt + I on Mac in your browser to bring up the developer tools)
  – We use a remote WebPagetest server: [http://www.webpagetest.org](http://www.webpagetest.org)
Web Performance - Analysis Tools

- The best online waterfall tool is probably WebPageTest.org (commonly known as WPT), developed by Google, AOL, and others.
- It offers dozens of locations around the world from which to perform tests and has an impressive list of browsers to test in.
- The downside to WPT is that it shows how the page loads on a remote machine, not your own device.
Analysis Tools - Resource Waterfall

- Optimizing performance is about knowing how and when to load assets and avoiding any blocking of content
- Typing a website address in your browser triggers a set of HTTP requests and responses that together make a website appear in your browser.
  - The easiest way to visually represent all these HTTP requests/responses is to plot them onto a so-called resource waterfall. These are graphs showing the order in which the browser is requesting resources, and the time that it takes each resource to download.
  - The reason why the resources are aligned in a waterfall is because the browser can't download all the resources at once. The browser follows rules for downloading each component, either in parallel or sequentially.
WebPagetest Illustration for sitepoint.com

1. Summary
2. Details
3. Performance Review
4. PageSpeed
5. Content Breakdown

<table>
<thead>
<tr>
<th>Document Complete</th>
<th>Fully Loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Time</td>
<td>DOM Elements</td>
</tr>
<tr>
<td>First Byte</td>
<td>Time</td>
</tr>
<tr>
<td>Start Render</td>
<td>Requests</td>
</tr>
<tr>
<td>Speed Index</td>
<td>Bytes In</td>
</tr>
<tr>
<td>DOM Elements</td>
<td>Time</td>
</tr>
<tr>
<td>Time</td>
<td>Requests</td>
</tr>
<tr>
<td>Requests</td>
<td>Bytes In</td>
</tr>
<tr>
<td>6.882s</td>
<td>34</td>
</tr>
<tr>
<td>0.577s</td>
<td>367 KB</td>
</tr>
<tr>
<td>3.071s</td>
<td>6.882s</td>
</tr>
<tr>
<td>595</td>
<td>367 KB</td>
</tr>
<tr>
<td>4031</td>
<td>6.882s</td>
</tr>
<tr>
<td>595</td>
<td>367 KB</td>
</tr>
<tr>
<td>5.374s</td>
<td>28</td>
</tr>
<tr>
<td>0.493s</td>
<td>317 KB</td>
</tr>
<tr>
<td>2.006s</td>
<td>5.374s</td>
</tr>
<tr>
<td>595</td>
<td>317 KB</td>
</tr>
<tr>
<td>3135</td>
<td>9.326s</td>
</tr>
<tr>
<td>595</td>
<td>493 KB</td>
</tr>
<tr>
<td>353 KB</td>
<td>12.41s</td>
</tr>
</tbody>
</table>

Waterfall | Screen Shot | Video

COMP6205 - Web Performance
WebPagetest Illustration

• In addition to the data available in most browser developer tools (total requests, total page size, and page load time), WPT provides several other important metrics:

1. **Load Time**: The time from the initial request until the browser load event. Also known as the document complete time.

2. **First Byte**: The First Byte time is measured as the time from the start of the initial navigation until the first byte of the base page is received by the browser.
3. **Start Render** time is the first point in time that content starts to appear in the browser. This could even just be the background color, or the logo at the top corner of the page.

   - The Start Render time is also plotted in the waterfall as a green line, which ideally should appear as early as possible.

4. **Document Complete** (same as **Page Load Time**) records when the initial page loading process is complete.

   - This time is also plotted in the waterfall as a blue line.
WebPagetest Illustration

5. **Fully Loaded** time occurs when all assets of the page have been loaded, including any activity that is triggered by JavaScript after the main content has loaded.
Reading a Waterfall

Waterfall View

<table>
<thead>
<tr>
<th></th>
<th>DNS Lookup</th>
<th>Initial Connection</th>
<th>SSL Negotiation</th>
<th>Time to First Byte</th>
<th>Content Download</th>
<th>3xx response</th>
<th>4xx+ response</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uma</strong></td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.5</td>
<td>0.6</td>
<td>0.7</td>
</tr>
<tr>
<td><strong>Initial</strong></td>
<td>0.8</td>
<td>1.0</td>
<td>1.2</td>
<td>1.3</td>
<td>1.4</td>
<td>1.5</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>Content</strong></td>
<td>1.7</td>
<td>1.8</td>
<td>1.9</td>
<td>2.0</td>
<td>2.1</td>
<td>2.2</td>
<td></td>
</tr>
</tbody>
</table>

1. www.google.co.uk - /
2. www.google.co.uk - /
3. www.google.co.uk - logo.png
4. www.google.co.uk - core.png
5. www.google.co.uk - nav.png
6. www.google.co.uk - color.png
7. www.google.co.uk - EHnpC000-6GKJH1Q.png
8. www.google.co.uk - EHnpC000-6GKJH1Q.png
9. google.com - i1_1967ca.png
10. google.com - i2_1967ca.png
11. google.com - i3_1967ca.png
12. google.com - i4_1967ca.png
13. google.com - i5_1967ca.png
14. google.com - i6_1967ca.png

CPU Utilization

Bandwidth (0 - 5000 Kbps)
A Break Down of HTTP Transaction

25 ms + 30 ms + 203 ms + 42 ms + 34 ms = 334 ms

https://platform.twitter.com/widgets.js
DNS Lookup

- **DNS** stands for Domain Name System. During the DNS lookup, the browser attempts to look up the domain of the asset it is trying to load. Each human-friendly URL has a computer-friendly IP address. You can use both in your browser. For example, **www.sitepoint.com** maps to **54.221.218.251**.

- There are many variables that can cause a delay during this event.
Initial Connection

• The **initial connection** occurs when the client and server perform a “handshake” to start communicating with each other over Transmission Control Protocol (TCP).

• If resolving the DNS of the server is like looking up a phone number, this is the step of dialing the number and waiting for someone to answer.

• The travel time to communicate from sender to receiver is governed by **speed of the network** and **distance** between the server and your device.
SSL Negotiation

- **Secure Socket Layer** (SSL) enables secure communication. It's a protocol for encrypting information over the Internet (using https:// instead of http://).
- HTTPS in the scheme instructs the browser to perform a secure handshake.
- This event is about negotiation between two parties (the browser and the server). Each SSL negotiation over HTTPS requires a new TCP connection.
Time to First Byte

- **Time to First Byte** (TTFB) occurs after the TCP handshake has successfully been executed, and after the browser and server have started talking to each other – when the first byte is actually sent to the browser.

- This metric also represents **latency**. The bigger the green bar, the higher the latency.

- Depending on your **location** and/or the device you're using, you will notice smaller or bigger latency.
Bandwidth vs Latency

Page Load Time as bandwidth increases

Page Load Time as latency decreases

Read more: http://chimera.labs.oreilly.com/books/12300000000545/ch10.html#LATENCY_BOTTLENECK
In early 2011, Huawei and Hibernia Atlantic began laying a new 3,000-mile fiber-optic link ("Hibernia Express") across the Atlantic Ocean to connect London to New York, with the sole goal of saving traders 5 milliseconds of latency by taking a shorter route between the cities, as compared with all other existing transatlantic links... cost over $400M
## Signal Latencies in Vacuum and Fiber

<table>
<thead>
<tr>
<th>Route</th>
<th>Distance</th>
<th>Time, light in vacuum</th>
<th>Time, light in fiber</th>
<th>Round-trip time (RTT) in fiber</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York to San Francisco</td>
<td>4,148 km</td>
<td>14 ms</td>
<td>21 ms</td>
<td>42 ms</td>
</tr>
<tr>
<td>New York to London</td>
<td>5,585 km</td>
<td>19 ms</td>
<td>28 ms</td>
<td>56 ms</td>
</tr>
<tr>
<td>New York to Sydney</td>
<td>15,993 km</td>
<td>53 ms</td>
<td>80 ms</td>
<td>160 ms</td>
</tr>
<tr>
<td>Equatorial circumference</td>
<td>40,075 km</td>
<td>133.7 ms</td>
<td>200 ms</td>
<td>200 ms</td>
</tr>
</tbody>
</table>
Content Download

- This event occurs when the asset that your browser requested actually starts to show up in your browser.
- You sometimes can guess by looking at the length of the blue bar how big the asset actually is. Images, for example, tend to have a bigger blue bar.
- The bigger the blue bar, the larger the actual asset, and hence the more time it takes to be sent over the wire.
- Minimizing the file size of the asset is key to optimizing the amount of time spent on Content Download.
Page-level Events

• Waterfalls are also decorated with bars and lines, marking the times at which page-level events. These are:

• **Start Render** (first paint): The time for the browser to display the first pixel of content (paint) on the screen.

• **DOM Content Loaded**: After the browser has received the HTML payload, it parses it into the Document Object Model (DOM), which is a virtual representation of the page structure. The browser will fire an event to let the page know that the DOM is ready so that the page can go on to interact with it.
Page-level Events

- **On Load**: The start and end time of the page’s load-event handler. The browser fires the load event when the DOM is ready and all images have loaded. Pages typically use this event handler to perform secondary tasks like loading content below the fold (outside of the user’s viewport).

- **Document Complete**: Effectively, the time that the browser fires the load event. This event name can be considered a misnomer because the document may not necessarily be complete. Around this time, the page’s script is hard at work in the load-event handler firing off more requests for secondary content. The incomplete nature of this metric is why Fully Loaded was added to the table of metrics from the previous section.
### Web Page Performance - Connection View

#### Connection View

<table>
<thead>
<tr>
<th>DNS Lookup</th>
<th>Initial Connection</th>
<th>SSL Negotiation</th>
<th>Start Render</th>
<th>DOM Content Loaded</th>
<th>On Load</th>
<th>Document Complete</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Domain</th>
<th>Duration (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>http://www.google.co.uk</code></td>
<td>2.1</td>
</tr>
<tr>
<td><code>www.google.co.uk</code></td>
<td>0.1</td>
</tr>
<tr>
<td><code>ssl.gstatic.com</code></td>
<td>0.2</td>
</tr>
<tr>
<td><code>www.gstatic.com</code></td>
<td>0.3</td>
</tr>
<tr>
<td><code>www.google.com</code></td>
<td>0.4</td>
</tr>
<tr>
<td><code>apis.googleapis.com</code></td>
<td>0.5</td>
</tr>
</tbody>
</table>

- **CPU Utilization**
- **Bandwidth In (0 - 5,000 Kbps)**

---

**html** | **js** | **css** | **image** | **flash** | **font** | **other**
---|---|---|---|---|---|---

**COMP6205 - Web Performance** 44
# Web Page Performance - Request Details

| #  | Resource                        | Content Type | Request Start | DNS Lookup | Initial Connection | SSL Negotiation | Time to First Byte | Content Download | Bytes Downloaded | Error/Status Code | IP             |
|----|---------------------------------|--------------|---------------|------------|---------------------|-----------------|--------------------|-------------------|------------------|------------------|----------------|                |
| 1  | http://www.google.co.uk/        | text/html     | 0.106 s       | 59 ms      | 46 ms               | -               | 71 ms              | -                 | 1.1 KB           | 302              | 216.58.208.67   |
| 2  | https://www.google.co.uk/?gws_rd=ssl | text/html     | 0.311 s       | -          | 47 ms               | 75 ms           | 120 ms             | 159 ms            | 55.8 KB          | 200              | 216.58.208.67   |
| 3  | https://www.google.com_/ico/googleg_logo.png | image/jpeg     | 0.59 s        | -          | -                   | -               | 59 ms              | 4 ms              | 1.6 KB           | 200              | 216.58.208.67   |
| 4  | https://www.google.com_/uct_core_48.png | image/png     | 0.610 s       | -          | -                   | -               | 58 ms              | 3 ms              | 2.1 KB           | 200              | 216.58.208.67   |
| 5  | https://www.google.com_/ages/av_logo242.png | image/png     | 0.619 s       | -          | -                   | -               | 60 ms              | 39 ms             | 21.4 KB          | 200              | 216.58.208.67   |
| 6  | https://www.google.com_/o_color_272.png | image/png     | 0.62 s        | -          | -                   | -               | 98 ms              | 23 ms             | 13.2 KB          | 200              | 216.68.208.67   |
| 7  | https://www.google.com_/C6FEAbpOrGQ-8GK/HJ/8Q | text/javascript | 0.65 s        | -          | -                   | -               | 93 ms              | 226 ms            | 129.3 KB         | 200              | 216.58.208.67   |
| 8  | https://www.google.com_/C6FEAbpOrGQ-8GK/HJ/8Q | text/javascript | 1.301 s       | -          | -                   | -               | 50 ms              | 53 ms             | 24.2 KB          | 200              | 216.58.208.67   |
| 9  | https://gstatic.com/ages/1_1997ca66.png | image/png     | 1.331 s       | 173 ms     | 120 ms              | 294 ms          | 73 ms              | 16 ms             | 8.1 KB           | 200              | 216.58.208.67   |
| 10 | https://gstatic.com/kK572_eeZpYEU22Pckw | text/javascript | 1.411 s       | 113 ms     | 49 ms               | 99 ms           | 65 ms              | 70 ms             | 47.9 KB          | 200              | 216.58.208.67   |
| 11 | https://www.google.com_/dAG8z=x=144856569406 | text/html     | 1.513 s       | -          | -                   | -               | 94 ms              | -                | 0.0 KB           | 204              | 216.58.208.67   |
| 12 | https://www.google.com_/putassistant/b.png | image/png     | 1.646 s       | 259 ms     | 47 ms               | 68 ms           | 48 ms              | 2 ms              | 0.7 KB           | 200              | 216.58.208.68   |
| 13 | https://apis.google.com/1QwCg=qapi.loaded_0 | text/javascript | 1.825 s       | 3 ms        | 49 ms               | 78 ms           | 51 ms              | 84 ms             | 48.0 KB          | 200              | 216.58.208.78   |
| 14 | https://www.google.com_/172.cstt.172.dl.625 | text/html     | 2.103 s       | -          | -                   | -               | 66 ms              | -                | 0.0 KB           | 204              | 216.58.208.67   |
Waterfall Chart

Time to First Byte

COMP6205 - Web Performance
Time Spent on Back-end/Frontend

Top 10

Google
Facebook
YouTube
Yahoo!
Baidu
Wikipedia
Twitter
Amazon
Tencent
LinkedIn

24% - backend
76% - frontend
Time Spent on Back-end/Frontend

- 8% - backend
- 92% - frontend

9,990+
This information shows the **Performance Golden Rule** and points to the motivation for focusing on frontend optimizations.

However if you’re worried about availability and scalability, focus on the backend. But if you’re worried about how long users are waiting for your website to load focusing on the frontend is your best bet.
Golden Rule

- 80-90% of the end-user response time is spent on the frontend

http://www.stevesouders.com/blog/2012/02/10/the-performance-golden-rule/ - Steve Souders

- The optimizations of frontend is the web developers job

- So, he suggests that you have to start there to improve performance.
Why Focus on Front-end Performance?

- The front-end (i.e. your HTML, CSS, JavaScript, and images) is the most **accessible part** of your website.
  - If you’re on a shared web hosting plan, you might not have root (or root-like) access to the server and therefore can’t tweak and adjust server settings.
  - And even if you *do* have the right permissions, web server and database engineering require specialized knowledge to give you any immediate benefits.
- It’s also **cheap**. Most of the front-end optimization discussed can be done at no other cost but your time.
- Not only is it inexpensive, but it’s the **best use of your time because** front-end performance is responsible for a very large part of a website’s response time.
Performance Matters

- Performance matters to the end users
  - Performance means money, big sites like eBay, Amazon, Bing, Google .... Have made experiments and measured statistic showing that
  - The smallest changes in perceived speed effects their revenue in big ways.
    

- Perceived speed vs. Actual Speed (Visual completeness)
- Many studies show user are more interested in perceived speed
- Fast page load time builds trust in your website.
Performance Metrics

- **Backend metrics** - Refers to how your site content goes to the visitor’s browser.
  - DNS Time
  - Connection Time
  - First Byte Time
  - Last Byte Time
  - Using CDN

- **Front-end metrics** refers to how the user’s browser executes your page’s content. Aspects such as:
  - Time To Title
  - Time To Start Render
  - Time To Display
  - Time To Interact

- **Content complexity** — Is a breakdown of the building blocks of your website.
  - Overall Asset Weight
    - JavaScript Weight
    - CSS Weight
    - Image Weight
  - Overall Asset Count
    - JavaScript Count
    - CSS Count
    - Image Count
  - Number of Domains
Web Page Performance – Backend

- **DNS Time:**
  - DNS Time is the time it takes for your DNS provider to execute its service. (A DNS processes the visitor’s URL request and returns the IP address.
  - That matches the URL by searching through enormous databases). You can find your DNS time with an online service like DNSStuff.com or WebsiteTest.com.
  - A typical DNS time is 60 milliseconds. If your DNS time is much slower than that, you may be best off finding a new DNS provider.

<table>
<thead>
<tr>
<th></th>
<th>50th percentile</th>
<th>80th percentile</th>
<th>95th percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS Time (msec)</td>
<td>60</td>
<td>149</td>
<td>413</td>
</tr>
</tbody>
</table>
Web Page Performance – Backend

• Connection Time:

  – Connection time is the time elapsed from the request to when the connection between the visitor’s browser and your origin server is established. Normal connection times are around 150 milliseconds.

  – Problems with connection time can be a bit more difficult to diagnose because they only crop up in certain circumstances.

  – If your server is overloaded with traffic – either because of heavy traffic on your site or because of bots – connection times will spike.

  – Geography can also play a role, with distant visitors experiencing longer connection time.
Web Page Performance – Backend

- **Connection Time:**
  - Load testing with tools like **LoadStorm** or **JMeter** will simulate increased traffic and reveal the increased connection time and/or 503 and 504 errors that would occur in times of heavy server usage.
  - If load testing reveals serious issues, the solutions are to upgrade your infrastructure or offload some of your assets onto a **content delivery network (CDN)** or caching server.

<table>
<thead>
<tr>
<th></th>
<th>50th percentile</th>
<th>80th percentile</th>
<th>95th percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect Time (msec)</td>
<td>144</td>
<td>188</td>
<td>227</td>
</tr>
</tbody>
</table>
Content Delivery Network (CDN)

- A **Content Delivery Network (CDN)** is an interconnected system of computers over the Internet that provides Web content **rapidly** to **numerous users** by **duplicating the content on multiple servers** and directing it to users from the nearest/most **optimal CDN location**. Example: Amazon CloudFront

- First step: the CDN receives your content
- Second step: your content is distributed over the CDN
- Third step: customer request
  - is served from the nearest CDN.
  - If there are high number of concurrent requests, **global load-balancing ensures that all requests are served by local resources very quickly.**
CDN - Technical benefits

• High availability
  – In case of a network outage the content is still served from another CDN location.

• Fast Delivery

• Highly scalable (no capacity issues)

• Distributed denial-of-service (DDoS) DDOS protection
  – DDoS attacks are not successful because the load they generate is efficiently shared over the CDN platform.

• You can SAVE:
  – Cost of Internet access
  – Server capacity
  – Colocation cost
Web Page Performance – Backend

- **First Byte Time:**
  - After the connection is established, the next step in the backend process is the first byte of your website to travel along the connection to the visitor’s browser. Once the first byte arrives, your **First Byte Time** is registered.
  - The **main impediment** in achieving a good First Byte Time is **poorly structured backend code**.
  - Every web page has content that is “static” and does not need to be **personalized** to the particular visitor.
  - The static content should be separated from the rest so that it can be sent to the user immediately, rather than waiting for personalized content to be processed and sent along with it.
  - Issues with First Byte Time can be found with a combination of **continuous performance monitoring** and **load testing**.
Web Page Performance – Backend

- **Last Byte Time:**
  - It is logged when every byte of your website has made it to the visitor’s browser.
  - An average last byte time is about 700 ms, although we’ve seen times well over three seconds in the data for the higher percentiles.
  - Factors in determining Last Byte Time are largely to do with the Backend code and database.
  - If your code is not optimized, your database queries are poorly written, or if you’re not taking advantage of memcache, your Last Byte Time could be suffering.
  - Load testing tools, as well as application performance management (APM) tools like Gomez, Keynote, and Yottaa can help.
Web Page Performance – Front-end

• Four stages in the front-end execution process are commonly noted:

• Time to Title:
  – Time To Title is the time elapsed from the moment a visitor’s browser requests your site to the moment that your site’s title appears in the tab of his or her browser.
  – This metric actually has little to do with the front end of your website.
  – Time To Title is largely dictated by your backend performance—that is, how fast your website’s content is delivered from your origin server to the visitor’s browser.
Performance – Front-end: Time to Title

- So why is it in the front-end user experience category rather than backend?
  - Because on the front-end, where user perception of load time is half the battle, Time To Title is the first indication to the visitor that your site exists and is on the way.
  - When users see the title appear promptly they are apt to be patient throughout the load process. This makes Time To Title the first of several user experience metrics.

<table>
<thead>
<tr>
<th></th>
<th>50th percentile</th>
<th>80th percentile</th>
<th>95th percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to Title (msec)</td>
<td>1,259</td>
<td>2,368</td>
<td>4,647</td>
</tr>
</tbody>
</table>
Web Page Performance – Front-end

• **Time to Start Render:**
  
  – Time to Start Render is the *time elapsed from the request to when the visitor sees actual website content appear on the page.*
  
  – Like the Time To Title, this moment is important since it *assures* the visitor that your *site is loading*.
  
  – Nobody likes staring at a blank page.
  
  – Assuring visitors that they are in the right place and will be soon seeing the content they expect will promote a good perception of your website.
Time to Start Render

Time to Render

Frequency (# of sites)

Time to Interact (Seconds)

Frequency
Cumulative

COMP6205 - Web Performance
Web Page Performance – Front-end

- **Time To Display:**
  
  - Time To Display is the time elapsed from the request to when the browser has finished parsing the HTML page, constructed the Document Object Model (DOM), and displayed the HTML document.
  
  - This all means that the page will look like a web page, but there may be some images, interactive elements, and other media that haven’t fully loaded.
Time To Display

<table>
<thead>
<tr>
<th>Time to Display (msec)</th>
<th>50&lt;sup&gt;th&lt;/sup&gt; percentile</th>
<th>80&lt;sup&gt;th&lt;/sup&gt; percentile</th>
<th>95&lt;sup&gt;th&lt;/sup&gt; percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5,168</td>
<td>8,780</td>
<td>15,254</td>
</tr>
</tbody>
</table>
Web Page Performance – Front-end

• **Time to Interact:**
  
  – Time To Interact is the time elapsed from the request to the moment the user can interact with the page. (By “interact” we mean the page will respond properly to the visitor clicking a link, scrolling, typing into a field, or activating an element like a hover effect).

  – This does **not** mean that the page is fully loaded, as there may be scripts, trackers, and other assets that continue to load in the background.

  – But it does mean that the visitor can use the web page, and that’s an important moment.

  – Many site owners choose Time To Interact as the principle index for overall web performance because of its relationship with user experience.
Time to Interact

<table>
<thead>
<tr>
<th></th>
<th>50th percentile</th>
<th>80th percentile</th>
<th>95th percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to Interact (msec)</td>
<td>6,263</td>
<td>10,643</td>
<td>18,494</td>
</tr>
</tbody>
</table>
Web Page Performance – Content Complexity

• In order to put frontend and backend metrics to use, you’ll need to know about the composition of your website.

• **Content complexity** statistics can reveal areas of your site with potential for improvement.

• Just as with timing metrics, you can use percentiles from sample of the Web to determine how the composition of the site stacks up against the rest.
Content Complexity

• **Total Asset Weight** (including JavaScript Weight, CSS Weight, Image Weight ..)

  – The overall weight -- that is, the number of bytes -- of your website factors into its speed, as does the weight of individual assets (one heavy and slow-loading asset can have rippling effect on the performance)

• Use weight metrics to identify categories of assets that are too heavy in aggregate, and then use a waterfall chart to zero in on specific assets within that category that can be fixed or cut.

• For instance, if the total weight of your JavaScript files is in a much higher percentile and this the cause of slow down, try to optimize it.
The site is slow -- in viewing surveys with a Time To Interact over 10 seconds. The content complexity helps tell us why.
## Asset Weight

<table>
<thead>
<tr>
<th></th>
<th>50th percentile</th>
<th>80th percentile</th>
<th>95th percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset Weight (Bytes)</td>
<td>690,779</td>
<td>1,557,812</td>
<td>3,393,192</td>
</tr>
<tr>
<td>JS Weight (Bytes)</td>
<td>119,378</td>
<td>346,852</td>
<td>586,442</td>
</tr>
<tr>
<td>CSS Weight (Bytes)</td>
<td>18,134</td>
<td>60,265</td>
<td>120,000</td>
</tr>
<tr>
<td>Image Weight (Bytes)</td>
<td>318,788</td>
<td>895,948</td>
<td>2,056,237</td>
</tr>
</tbody>
</table>
Effect of Asset Weight on Time to Load

- **Image Bytes vs Time to Load**
- **javascript Bytes vs Time to Load**
- **css Bytes vs Time to Load**
Web Page Performance – Overall Asset Count

- **Overall Asset Count** (include JavaScript Count, CSS Count, Image Count..)
  - More assets necessarily mean more weight -- that’s reason enough to keep track of asset count.
  - But in addition, each time a visitor’s browser makes the trip to your origin server to fetch an asset for your site, it adds time to the page load.
  - That means each asset slows down your site no matter how small or compressed it is.
Each time a browser makes the request to the server to fetch an asset for your site, it adds time to the page load. That means each asset slows down your site no matter how small or compressed it is.
Number of Domains

- Most of your site’s content is hosted on your domain (or “origin”) server.

- But if your site fetch content from a third party, such as widgets for social media interaction or an embedded YouTube video, then there are additional domains in the mix.

- Taking content from other domains would not be bad for performance in a perfect world

<table>
<thead>
<tr>
<th></th>
<th>50th percentile</th>
<th>80th percentile</th>
<th>95th percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domains</td>
<td>6</td>
<td>17</td>
<td>33</td>
</tr>
</tbody>
</table>
Web Page Performance – HTTP Parameters

• Parallel Connections and Downloading

• Support for Virtual Hosting

• Persistent Connections and Keep-Alive

• Caching
**HTTP History**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Organization</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP 1.0</td>
<td>W3C</td>
<td>Feb 1996</td>
</tr>
<tr>
<td>HTTP 1.1</td>
<td>W3C</td>
<td>Jun 1999</td>
</tr>
<tr>
<td>SPDY</td>
<td>Google</td>
<td>Nov 2011</td>
</tr>
<tr>
<td>HTTP/2</td>
<td>IETF</td>
<td>May 2015</td>
</tr>
</tbody>
</table>

- **SPDY** (pronounced speedy) is an open networking protocol developed primarily at Google for transporting web content. **SPDY** manipulates HTTP traffic, with particular goals of reducing web page load latency and improving web security.
HTTP/2

• As of February 2015, the HTTP/2 specification has been approved by the Internet Engineering Task Force (IETF).

• One of the main advantages of HTTP/2 over HTTP/1.1 is that it allows many concurrent HTTP requests to run across one single TCP connection.

• HTTP/2 also comes with default HTTP header compression, as opposed to HTTP/1.1, which uses optional compression.

• In addition, with HTTP/2, the server can proactively send the client the resource it will need, known as server push.
What is important?!

- Reduce Page Size (<500Kb)
- Reduce the number of roundtrips (<40 per Page...)
- Structure the page (to improve render & download)
  - CSS First
  - Javascript last
- Cache, Cache, Cache
Goole’s Rules

- Avoid bad requests
- Avoid CSS expressions
- Combine external CSS
- Combine external JavaScript
- Defer loading of JavaScript
- Enable compression
- Leverage browser caching
- Leverage proxy caching
- Minify CSS
- Minify HTML
- Minify JavaScript
- Minimize request size
- Minimize DNS lookups
- Minimize redirects
- Optimize images
- Optimize the order of styles and scripts
- Parallelize downloads across hostnames
- Put CSS in the document head
- Remove unused CSS
- Serve resources from a consistent URL
- Serve scaled images
- Serve static content from a cookieless domain
- Specify a character set early
- Specify image dimensions
- Use efficient CSS selectors
Yahoo’s Rules

- Minimize HTTP Requests
- Use a Content Delivery Network
- Add an Expires or a Cache-Control Header
- Gzip Components
- Put StyleSheets at the Top
- Put Scripts at the Bottom
- Avoid CSS Expressions
- Make JavaScript and CSS External
- Reduce DNS Lookups
- Minify JavaScript and CSS
- Avoid Redirects
- Remove Duplicate Scripts

- Configure ETags
- Make AJAX Cacheable
- Use GET for AJAX Requests
- Reduce the Number of DOM Elements
- No 404s
- Reduce Cookie Size
- Use Cookie-Free Domains for Components
- Avoid Filters
- Do Not Scale Images in HTML
- Make favicon.ico Small & Cacheable
Conclusions

• Performance tuning of Web-based systems is a multi-variable and iterative process.

• A systematic planning is an effective approach.

• The tuning process involves:
  
  ▪ Application review
  ▪ System review
  ▪ Test design
  ▪ Data collection and analysis
  ▪ System and application modification

• Test design is crucial to successful performance tuning.
1. Professional Website Performance: Optimizing the Front-End and Back-End by Peter Smith

2. High-Performance Browser Networking by Ilya Grigorik, chapter 9 - 15

3. High Performance Web Sites by Steve Souders


References

*Using WebPageTest*

*High Performance Web Sites*

*Even Faster Web Sites*

Rick Viscomi, Andy Davies & Marcel Duran
References

- Lean Websites
  - By Barbara Bermes
  - Because Web Performance Simply Matters

- Designing for Performance
  - By Lara Callender Hogan