radware

STATE OF THE UNION

Ecommerce Page Speed & Web Performance

Summer 2014

SHARE THIS WHITEPAPER

Table of Contents

Executive Summary

In recent years, faster web page speed has been correlated with every business metric site owners care about:

- **Search results and page views** – At Smartfurniture.com, faster pages resulted in a 20% increase in organic search traffic and 14% more page views.[1]
- **Conversions** – Walmart.com found that for every one second of performance improvement, the site experiences up to a 2% conversion increase.[2]
- **Cart size and revenue** – AutoAnything.com cut load times in half and experienced an 11% increase in average cart size and a 13% revenue increase.[3]

Despite the fact that just a few seconds – and sometimes even fractions of a second – can make the difference between online success and failure, it can be difficult for site owners to gain a true understanding of their web site's performance. If you are a accessing your own company's website from the office, it's very likely you're enjoying the responsiveness of your corporate LAN, thereby guaranteeing speedy load times for your own site.

Since 2010, Radware has measured and analyzed the performance – from a real-world perspective – of the top 500 online retailers.[4] The purpose of this research is to gain ongoing visibility into how leading ecommerce sites perform for visitors who use the internet under normal browsing conditions.

This report answers the following questions:

- How do pages actually render in real-world scenarios?
- Given the assumption that page speed is an urgent issue for online retailers, has this urgency translated into faster pages over time?
- How quickly are retailers acting to adopt performance best practices that could give them a significant competitive advantage?

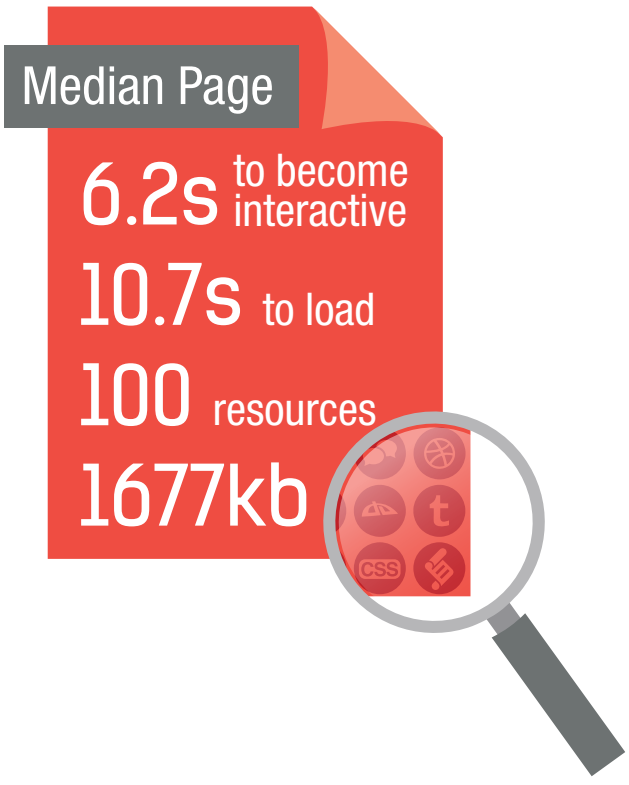
Except where otherwise noted, the results discussed in this report are for pages tested in Chrome 35. At the time of conducting this research, Chrome was the most widely used browser in the United States, with a market share of 36%.[5]

Key Findings

1. **Most retail websites are not meeting user expectations.**
   In an ideal world, web pages would render in three seconds or less. Yet we found that the median top 100 ecommerce home page takes 6.2 seconds to render its primary content and 10.7 seconds to fully load. Only 14% of the top 100 retail sites were able to deliver an optimal user experience. 17% took ten or more seconds just to be become interactive.

2. **Websites are getting slower…fast.**
   In just one year, median time to interact (TTI) has slowed down by 27% (from 4.9 seconds to 6.2 seconds), and median load time has suffered a 49% increase (from 7.2 seconds to 10.7 seconds).

3. **Page size and complexity are major factors in this performance slowdown.**
   The median page has grown by 67% in just one year – from 1007 KB in Summer 2013 to 1677 KB now. In 2013, the median page contained 82 resource requests. Today, the median page contains



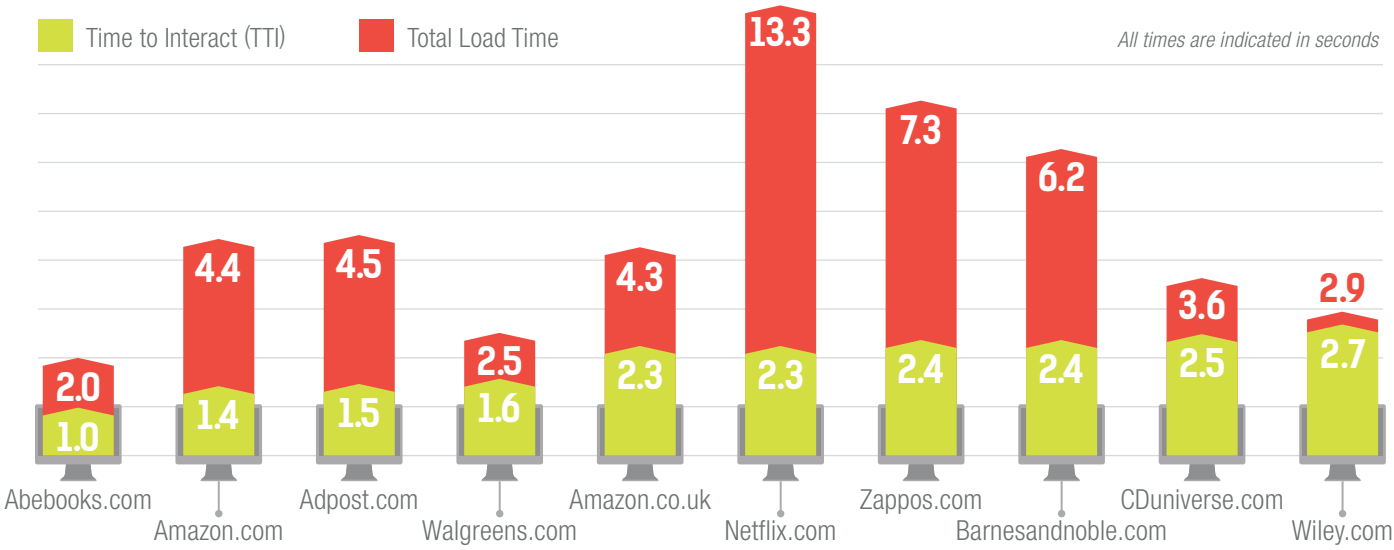
100 requests. Much of this growth in size and complexity is due to the proliferation of poorly optimized images and third-party scripts (e.g. page analytics, tracking beacons, and social buttons).

4. **Site owners are missing clear opportunities to better optimize their pages.**
   Most sites have implemented fundamental performance practices, but many are failing to leverage more advanced techniques and missing out on valuable opportunities to accelerate their pages. While 96% of sites enable "keep-alives" (a technique that allows TCP connections to remain open longer, thereby reducing the time spent re-opening connections) and 78% use a content delivery network to cache page resources closer to end users (thereby shortening server round trips and speeding up rendering time), most sites failed to properly implement image optimization techniques, such as compression and progressive JPEGs.

5. **Many sites are making the same three mistakes, which ultimately hurt the user experience.**
   A surprising number of sites experience the same recurring performance/usability problems, including delayed rendering and pop-ups that interrupt page render.

Who Was Fastest?

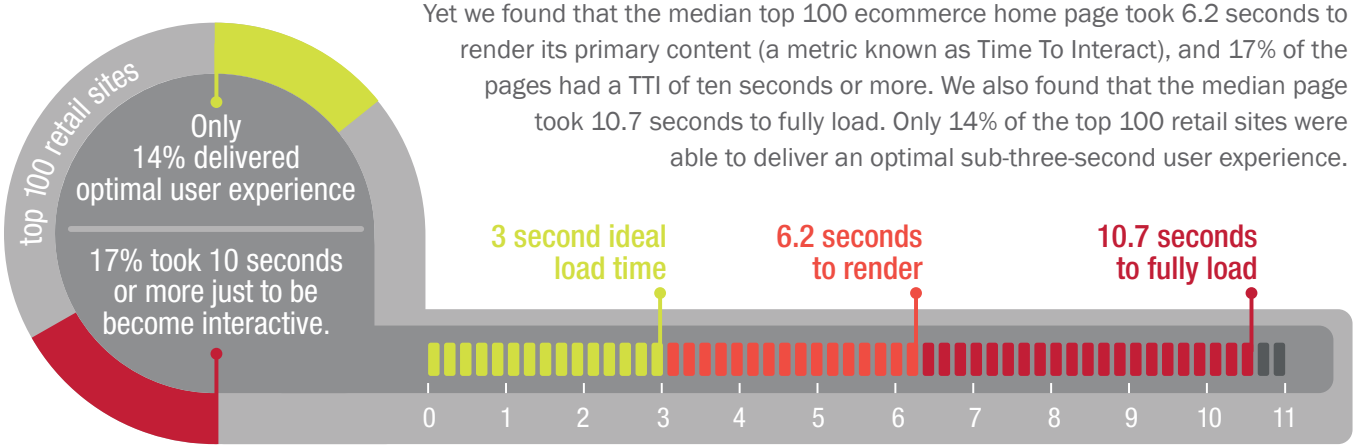
Among the top 100 sites, these were the fastest in terms of their ability to display meaningful, interactive content (e.g., feature banners with functional call-to-action buttons). This metric is known as "time to interact" (TTI) and is distinct from the better-known "load time" metric. (Load time indicates when all of a page's resources – from images to third-party-party scripts – have downloaded and rendered.) From a user experience perspective, TTI is a more meaningful performance metric than load time, as it indicates when a page begins to be usable.



We have provided the time to interact alongside each page's full load time in order to give perspective into the distinction between the two metrics, and to illustrate that **load time is not always the most meaningful measure of a site's performance**. For example, while Netflix has a load time of 13.3 seconds, it has a time to interact of 2.3 seconds; the TTI indicates that this site delivers a satisfactory user experience.

Finding #1: The Median Web Page Takes 6.2 Seconds to Render Primary Content and 10.7 Seconds to Load

The majority of online shoppers report that they will abandon a page that takes more than three seconds to load,[6] and ten seconds is the maximum threshold for most visitors' patience.[7]

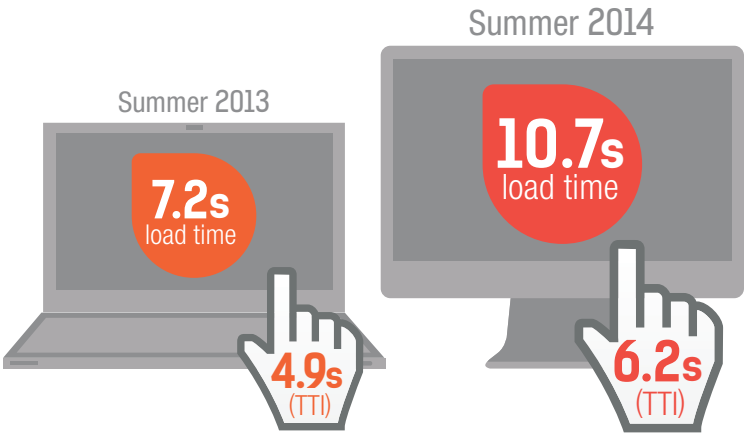


Yet we found that the median top 100 ecommerce home page took 6.2 seconds to render its primary content (a metric known as Time To Interact), and 17% of the pages had a TTI of ten seconds or more. We also found that the median page took 10.7 seconds to fully load. Only 14% of the top 100 retail sites were able to deliver an optimal sub-three-second user experience.

Finding #2: Websites are Getting Slower… Fast

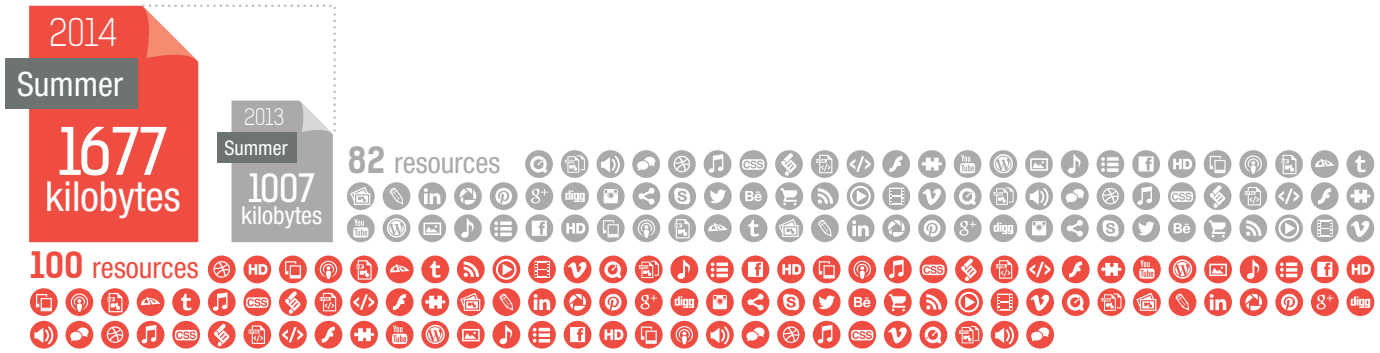
In just one year, the median Time to Interact (TTI) has slowed down by 27% (from 4.9 seconds to 6.2 seconds), and median load time has experienced a 49% slowdown (from 7.2 seconds to 10.7 seconds).

To understand why performance has suffered so dramatically over just twelve months, it is necessary to understand the impact of page size and complexity on front-end web performance. See the next section of this report for a detailed discussion of this issue.



Finding #3: Page Size and Complexity Present Major Performance Challenges

Bigger, more complex pages are often slower pages. The median page contains 100 resources and is 1677 KB in size. This represents 67% page growth in just one year.



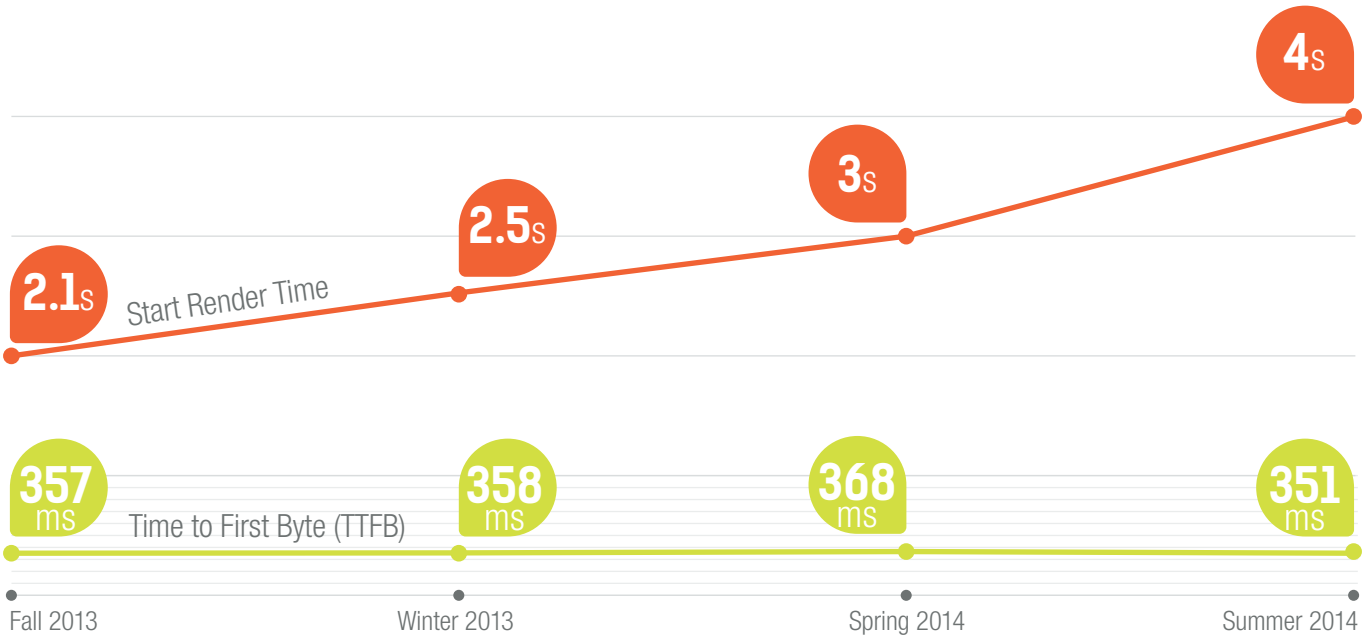
Much of the increase in page weight can be attributed to images. According to the HTTP Archive, images comprise 49% of the average top 100 page's total weight.[8] But too often these images are in the wrong format, uncompressed, or un-optimized – all of which add up to a serious performance drain.

For most sites, one of the greatest performance drains is the need to complete dozens of network round trips to retrieve page resources such as style sheets, scripts, and images. Each of these resources must make an individual round trip from the user's browser, which requests the file from the host server, which in turn delivers the file to the browser. Each of those requests experiences 75-140 milliseconds of latency, even for sites that use a content delivery network to cache resources closer to end users. These latency numbers add up quickly when the median page now contains 100 resources.

Page Complexity and How It Affects Time to Interact

To better understand why TTI has suffered in recent months, we looked to two additional metrics for more insight:

- Time to First Byte (TTFB) – This is the window of time between when the browser asks the server for content and when it starts to get the first bit back.
- Start Render Time – This metric indicates when content begins to display in the user's browser. Start Render Time can be delayed by slow TTFB, as well as a number of other factors.



Slow Time to First Byte is usually a sign that the site has a latency problem; in other words, your content is too far away from your visitors. This problem can be addressed by using a content delivery network (CDN) to cache page resources closer to end users. Most (78%) of the sites we tested currently use a CDN, and this number has not changed appreciably in recent years. Therefore it's not a surprise that TTFB has not improved significantly.

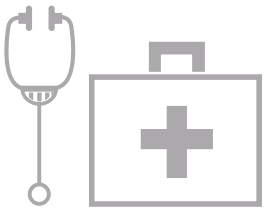
While TTFB has plateaued somewhat, Start Render Time has suffered – almost doubling in the past year, from 2.1 seconds to 4 seconds. Increased page complexity is the likely culprit. Modern web pages are more complex than ever – in the form of poorly placed style sheets, badly executed JavaScript, and third-party scripts (such as tracking beacons and page analytics) that block the rest of the page from rendering – and this complexity can incur a hefty performance penalty.

Finding #4: Site Owners are Missing Clear Opportunities to improve Performance

While the majority of sites we tested have implemented fundamental performance practices, many are failing to leverage more advanced techniques and missing out on valuable opportunities to accelerate their pages.

Almost all (96%) of sites enable keep-alives and 78% use a content delivery network to cache page resources closer to end users (as a result shortening server round trips and speeding up rendering time).

However most sites failed to properly implement image optimization techniques, such as image compression and progressive image rendering. According to the HTTP Archive, images comprise 49% of the average top 100 page's total weight, making them an excellent candidate for improving performance.[9]



KEEP-ALIVES

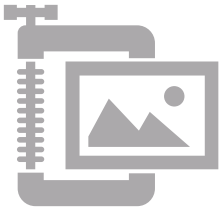
A 96%

B – 3% F – 1%

USE A CDN

no – 22%

yes 78%



COMPRESS IMAGES

A – 8% B – 9%
C – 20% D – 20%

F 43%

PROGRESSIVE JPEGs

A – 5% B – 3%
D – 1%
n/a – 15%

F 66%

Image Compression

Image compression is a performance technique that minimizes the size (in bytes) of a graphics file without degrading the quality of the image to an unacceptable level. Reducing an image's file size has two benefits:

- Reducing the amount of time required for images to be sent over the internet or downloaded.
- Increasing the number of images that can be stored in the browser cache, thereby improving page render time on repeat visits to the same page.

Despite the benefits of image compression, we found that 43% of pages failed to implement this technique, while only 8% scored an A.

Progressive JPEGs

Progressive image rendering is a performance best practice that can improve both real and perceived page speed. In one study of the top 2,000 retail sites, progressive JPEGs improved median load time by 15%.[10] Despite the performance benefits of using progressive images, only one out of three of the pages we tested contained progressive JPEGs, and only one out of twenty pages scored an A for implementation.

Google™

Google Page Speed assigns pages a score out of 100 for each best practice it measures.

A: 90–100 B: 80–89 C: 70–79 D: 60–69 F: 0–59

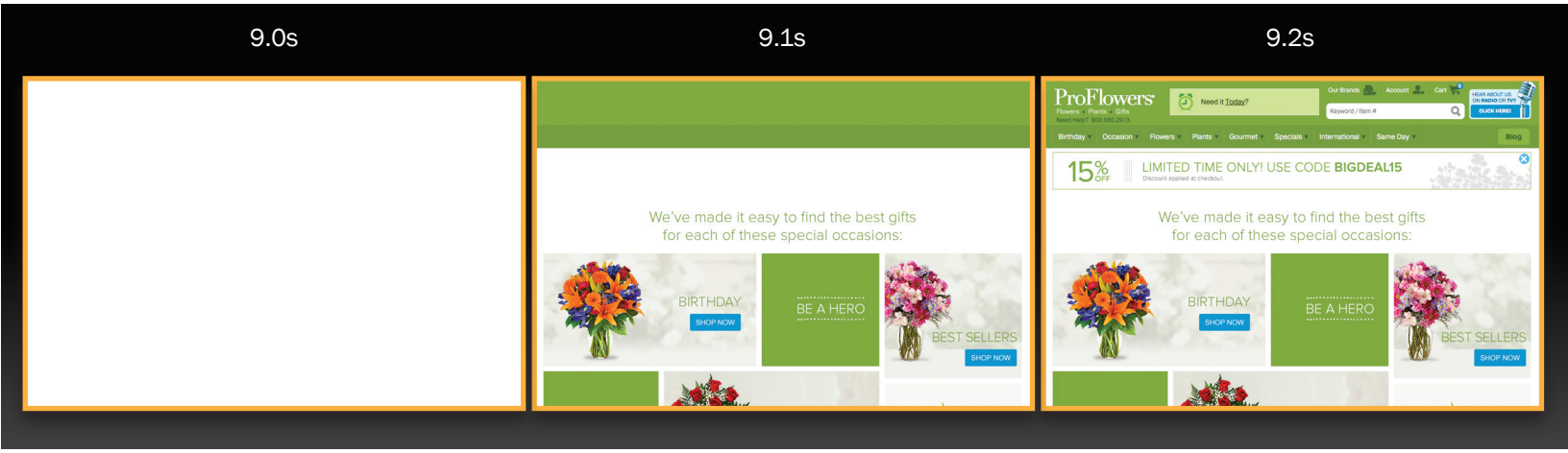
Finding #5: Many Sites Make the Same Three Usability Mistakes

We analyzed filmstrip views depicting the frame-by-frame rendering for the top 100 home pages. This exercise revealed three recurring performance and usability problems.

In each of these examples, we have identified common performance culprits and solutions. See "15 Things You Can Do to Cure Your Web Site's Performance Pains" in the next section of this report for a more detailed discussion of solutions.

1. Pages that are blank in the browser for several seconds, then suddenly populate.

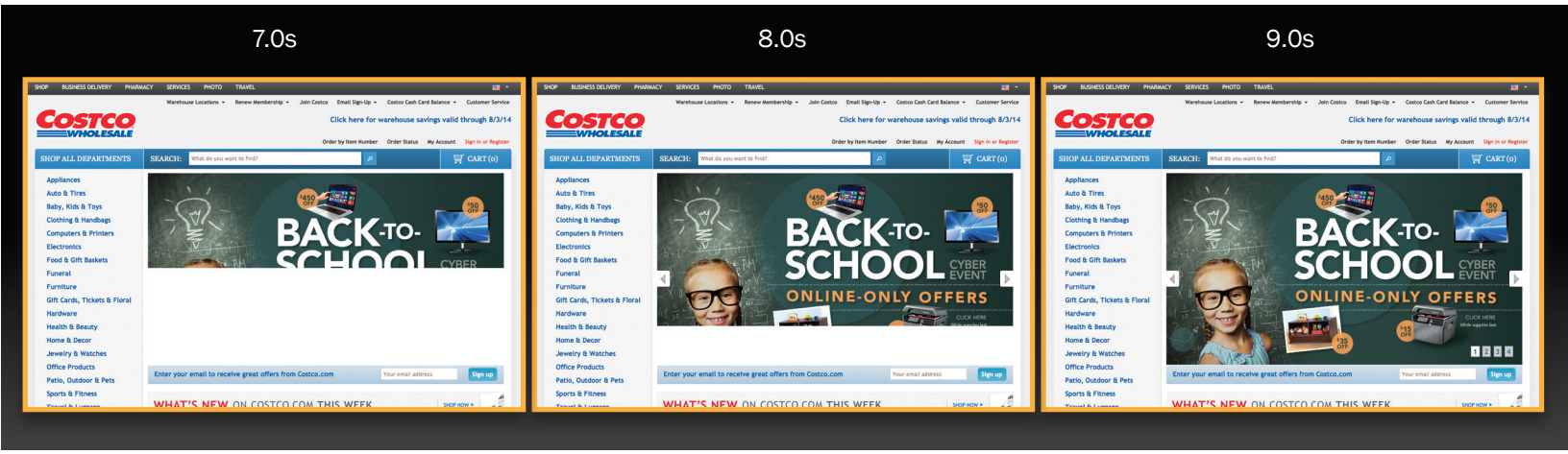
There is a measurable usability consequence of delaying the rendering of feature content: in one eye tracking study, it was found that **a user who endures an eight-second download delay spends only 1% of their total viewing time looking at the featured space on a page**. In contrast, a user who receives instantaneous page rendering spends 20% of their viewing time within the feature area of a page.[11]



Solution: Poorly executed styles sheets, JavaScript, and third-party scripts can block the rest of the page from rendering. Ensure that these resources are optimized and correctly implemented so as not to block the critical rendering path.

2. Pages in which the call to action is the last thing to render.

Placing the call to action (CTA) at the bottom of feature banners has become such a widely practiced design convention that most of us do not think twice about it; however, it frequently incurs a significant performance penalty, particularly in pages that use baseline images, which load line by line or in chunks. In many of the pages we studied, it was noted that the CTA – arguably the most critical page element – was often the last visible element to render.

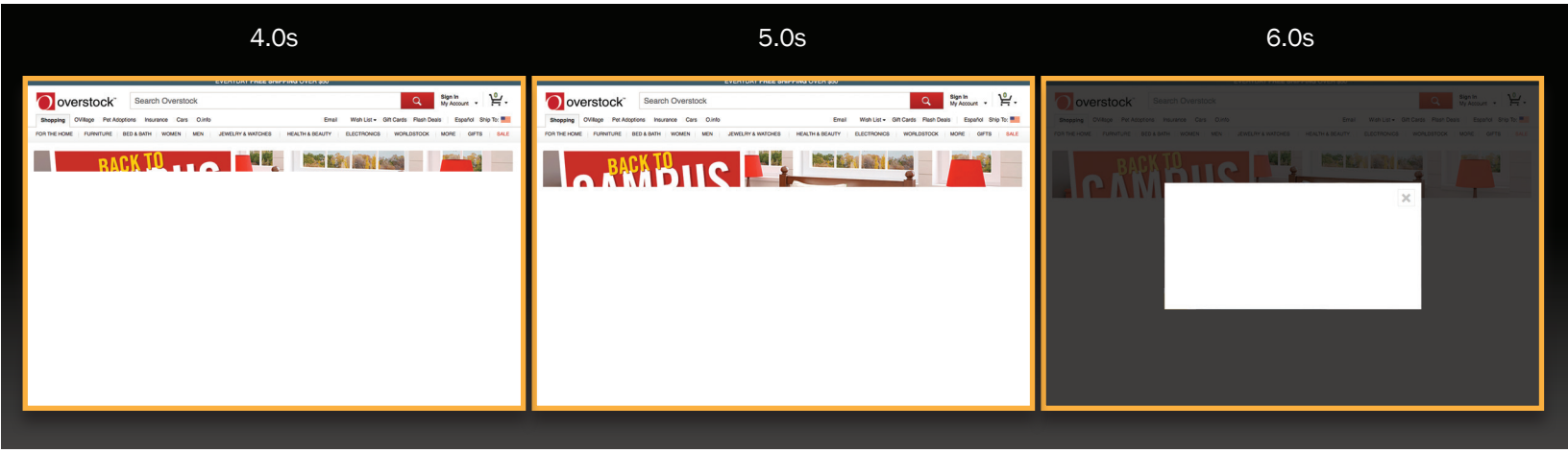


Solution(s): Consider using progressive JPEGs, which render in layers of increasingly high resolutions. By serving visual content to the page more quickly, progressive JPEGs can improve both actual and perceived page speed.

3. Pages in which a popup blocks the main page before it finishes rendering.

This was a recurring usability issue on many sites: within seconds of arriving at the home page, users are served with a pop-up before the rest of the page has rendered. Not only do these pop-ups act as a usability irritant, but they can also slow down or block the rendering of the main page content.

There are some use cases that support on-arrival pop-ups, such as requiring a user to identify their location in order to serve accurate item and shipping costs; however, in many cases the pop-ups noted in our study were for newsletters, surveys, and other opt-in marketing campaigns.



Solution: Delay the pop-up for at least ten seconds, to give visitors an opportunity to view the page.

15 Things You Can Do to Cure Your Web Site's Performance Pains

There are a number of best practices site owners can implement in order to improve both the real and perceived user experience for online shoppers. Some of these techniques can be implemented manually or via an automated solution, while others can only be performed by automated solutions.

1. Consolidate JavaScript and CSS

Consolidating JavaScript code and CSS styles into common files that can be shared across multiple pages should be a common practice. This technique simplifies code maintenance and improves the efficiency of client-side caching. In JavaScript files, be sure that the same script isn't downloaded multiple times for one page. Redundant script downloads are especially likely when large teams or multiple teams collaborate on page development.

2. Minify Code

Minification, which is usually applied to scripts and style sheets, eliminates non-essential characters such as spaces, newline characters, and comments. A correctly minified resource is used on the client without any special processing, and file-size reductions average about 20%. Script and style blocks within HTML pages can also be minified.

There are many good libraries available to perform minification, often along with services to combine multiple files into one, which additionally reduces requests.

3. Enable Keep-Alives

Enabling keep-alives is one of the easiest "low hanging fruit" on the performance optimization tree, yet a significant number of sites fail to do this. TCP connection is the process by which both the user and the server send and receive acknowledgment that a connection has been made and data can begin to be transferred. Too many TCP connections

will slow down your site. It's not easy to speed up TCP connection, but you can control how many times the connection takes place. To enable keep-alives, make sure you have the proper configuration on your servers and load balancer.

4. Compress Text

Compression technologies such as gzip reduce payloads at the slight cost of adding processing steps to compress on the server and decompress in the browser. These operations are highly optimized, however, and tests show that the overall effect is a net improvement in performance. Text-based responses, including HTML, XML, JSON (JavaScript Object Notation), JavaScript, and CSS, can all be reduced in size by as much as 70%.

5. Sprite Images

Spriting is a CSS technique for consolidating images. Sprites are simply multiple images combined into a rectilinear grid in one large image. The page fetches the large image all at once as a single CSS background image and then uses CSS background positioning to display the individual component images as needed on the page. This reduces multiple requests to only one, significantly improving performance.

6. Compress Images

Image compression is a performance technique that minimizes the size (in bytes) of a graphics file without degrading the quality of the image to an unacceptable level. Reducing an image's file size has two benefits:

- lessening the amount of time required for images to be sent over the internet or downloaded, and
- increasing the number of images that can be stored in the browser cache, thereby improving page render time on repeat visits to the same page.

7. Use Progressive Images

Progressive JPEGs are not a new innovation. They were widely used in the 1990s, but fell out of favor due to performance issues caused by slow connection speeds and crudely rendered JPEGs; watching a progressive image load pixel by pixel was a painful experience. Now that connection speeds have improved and progressive JPEGs are more sophisticated, this technique is feasible again and is returning as a newly heralded performance best practice. In one study of the top 2,000 retail sites, progressive JPEGs improved median load time by 15%.[12]

(Note: While all popular browsers will render progressive images, Safari, Mobile Safari, Opera and Internet Explorer 8 render them only as baseline JPEGs, meaning there is no performance benefit.)

8. Reformat Images

Inappropriate image formatting is an extremely common performance culprit. An image that is saved to the wrong format can be several times larger than it would be if saved to the optimal format. Images with unnecessarily high resolution waste bandwidth, processing time, and cache space.

As a general rule of thumb, these are the optimal formats for common image types:

- **Photos** – JPEG, PNG-24
- **Low complexity (few colors)** – GIF, PNG-8
- **Low complexity with transparency** – GIF, PNG-8
- **High complexity with transparency** – PNG-24
- **Line art** – SVG

9. Ensure That Feature Images Are Optimized to Load Early and Quickly

As discussed earlier in this report, site owners should be aware of the usability consequence of delaying the rendering of feature content: a user who experiences instantaneous page rendering spends 20% of their viewing

time within the feature area of a page, whereas a user who endures an eight-second download delay spends only 1% of their total viewing time looking at the featured space on a page.

10. Rethink the Design and Location of Call-to-Action Links in Feature Graphics

While the accepted design convention has been to position CTA buttons at the bottom of feature banners, this convention does not always serve the best interests of users or site owners, as shoppers must wait for the image to fully render before taking their next action on the page. Alternative solutions include repositioning the CTA, or using progressive images, which display the CTA earlier. (More on this practice below.)

11. Defer Rendering "Below the Fold" Content

Ensure that the user sees the page quicker by delaying the loading and rendering of any content that is below the initially visible area, sometimes called "below the fold." To eliminate the need to reflow content after the remainder of the page is loaded, replace images initially with placeholder <img> tags that specify the correct height and width.

12. Defer Loading and Executing Non-Essential Scripts

Many script libraries aren't needed until after a page has finished rendering. Downloading and parsing these scripts can safely be deferred until after the onload event. For example, scripts that support interactive user behavior, such as "drag and drop," can't possibly be called before the user has even seen the page. The same logic applies to script execution. Defer as much as possible until after onload instead of needlessly holding up the initial rendering of the important visible content on the page.

The script to defer could be your own or, often more importantly, scripts from third parties. Poorly optimized scripts for advertisements, social media widgets, or analytics support can block a page from rendering, sometimes adding precious seconds to load times.

13. Use AJAX for Progressive Enhancement

AJAX (Asynchronous JavaScript and XML) is a technique for using the XHR (XMLHttpRequest) object to fetch data from a web server without refreshing the page where the code is running. AJAX enables a page to display updated data in a section of a page without reconstructing the entire page. This is often used to respond to user interaction, but it can also enable your application to load a bare-bones version of a page quickly, and then to fill in more detailed content while the user is already viewing the page.

14. Preload Page Resources in the Browser

Auto-preloading is a powerful performance technique in which all user paths through a website are observed and recorded. Based on this massive amount of aggregated data, the auto-preloading engine can predict where a user is likely to go based on the page they are currently on and the previous pages in their path. The engine loads the resources for those "next" pages in the user's browser cache, enabling the page to render up to 70% faster.

Note that this is a data-intensive, highly dynamic technique that can only be performed by an automated solution.

15. Implement an Automated Web Performance Optimization Solution

While many of the performance techniques outlined above can be performed manually by developers, hand-coding pages for performance is specialized, time-consuming work. It is also a never-ending task, particularly on highly dynamic sites that contain hundreds of objects per page, as both browser requirements and page requirements continue to develop. Automated front-end performance optimization solutions, such as Radware FastView, apply a range of performance techniques that deliver faster pages consistently and reliably across the entire site.

Takeaways

1. As retail web pages grow larger and more complex, performance degradation continues to escalate.
Today, a typical page takes 6 seconds or longer to render primary content – more than twice as slow as the ideal user experience. This represents a 27% slowdown in just one year. Ultimately, this decline in user experience hurts both shoppers and site owners.

2. Images are one of the leading causes of "page bloat".
Images comprise almost half of the average top 100 page's total weight, making them simultaneously one of the single greatest performance drains while also presenting one of the single greatest performance opportunities. Most site owners are not fully taking advantage of image optimization techniques – such as image compression and progressive JPEGs – which can dramatically improve both real and perceived load times.

3. Modern web pages are more complex than ever, and this complexity comes with a performance price tag.
The median web page contains 100 resources, with some pages easily containing hundreds of resources. With this many resources, some kind of performance penalty is inevitable. Poorly placed style sheets, badly executed JavaScript, and third-party scripts (such as tracking beacons and page analytics) that block the rest of the page from rendering – these are all examples of things that can go wrong in today's complex, dynamic web pages.

4. Many sites commit the same usability mistakes.
All of the problems outlined above lead to the same recurring problems at the user experience end of the equation. These problems include rendering the most critical page content last and serving popups that block primary content before it renders. If you focus your performance measurement efforts on metrics such as start render and load time, you may not identify these problems. To better understand how visitors see a site – and to identify usability issues that might otherwise be missed – it is crucial to scrutinize how pages perform frame by frame, and over a real-world connection.

5. These problems are all fixable.
Despite the limitations imposed by challenges like high-resolution hero images and unpredictable third-party scripts, it is possible to render primary page content in three seconds or less. There are dozens of web performance techniques – from consolidating page resources to deferring non-essential scripts – that empower site owners to deliver the best possible experience to their customers.

Methodology

The tests in this study were conducted using an online tool called WebPagetest – an open-source project primarily developed and supported by Google – which simulates page load times from a real user's perspective using real browsers.

Radware tested the home page of every site in the Alexa Retail 500 nine consecutive times. The system automatically clears the cache between tests. The median test result for each home page was recorded and used in our calculations.

The tests were conducted on June 11, 2014, via the WebPagetest.org server in Dulles, VA, using Chrome 35 on a DSL connection.

In very few cases, WebPage test rendered a blank page or an error in which none of the page rendered. These instances were represented as null in the test appendix.

Also, in very few cases, WebPagetest.org rendered a page in more than 60 seconds (the default timeout for webpagetest.org). In these cases, 60 seconds was used for the result instead of null.

To identify the Time to Interact (TTI) for each page, we generated a timed filmstrip view of the median page load for each site in the Alexa Retail 100. Time to Interact is defined as the moment that the featured page content and primary call-to-action button or menu is rendered in the frame.

About Radware

Radware (NASDAQ: RDWR), is a global leader of application delivery and application security solutions for virtual and cloud data centers. Its award-winning solutions portfolio delivers full resilience for business-critical applications, maximum IT efficiency, and complete business agility. Radware's solutions empower more than 10,000 enterprise and carrier customers worldwide to adapt to market challenges quickly, maintain business continuity, and achieve maximum productivity while keeping costs down. For more information, please visit www.radware.com.

Sources

1 The Growing Need for Speed: How Site Performance Increasingly Influences Search Rankings. Retail Touch Points, May 2011
2 Real User Monitoring at Walmart.com. February 2012
3 AutoAnything Cuts Page Load Time in Half and Revs Up Sales by 13%
4 Top 500 Shopping Sites, Alexa.com
5 StatCounter Global Browser Stats. June 2014
6 PhoCusWright, Consumer Response to Travel Site Performance. June 2010
7 Website Response Times. June 2010
8 HTTP Archive. June 15, 2014
9 HTTP Archive. June 15, 2014
10 Patrick Meenan. Progressive JPEGs FTW! June 2013
11 Website Response Times
12 Progressive JPEGs FTW!