Front-End Performance Testing and Optimization

Abstract

Today, web user turnaround starts from more than 3 seconds of response time. This demands performance optimization on all application levels. Client side resources contributes on response time more than back-end. Optimizing the front-end performance from a single user point of view is a good practice before testing an application with high user loads. In this paper we will discuss the importance of web front-end resources optimization, resources to test the front-end performance, techniques to optimize the front-end performance and how these activities are complementary to load testing.

Introduction

The arrival of Web 2.0 has put lots of emphasis on the look and feel of web applications which puts more and more complexity on the front-end. Today, bad user experiences not only occur due to application, database, servers and infrastructures tuning but also due to the time it takes to load the web page and displaying its contents on end-user screen.

Therefore, identifying and resolving all client-side web application's performance issues without losing the look and feel of the web application are of utmost importance for good user experience. Another factor which makes the front-end performance more important is now Google rank websites in search results based on their web page speed.

What is a website Front-end?

Front-end or client-side is user interface or that particular part of an application (website or software) that user views on his/her screen. This interface helps user to interact directly with the application by entering desired/required commands and to access other application areas as well.

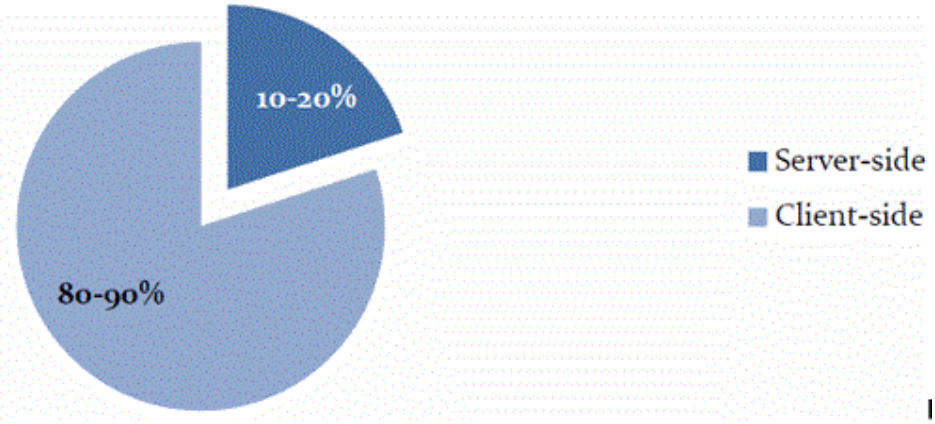
Importance of Front-end Performance Optimization

Few years back, when we talked about website performance optimization we meant optimizing its server-side only since web sites were mostly static and almost all the processing was done on server side. But the advent of Web 2.0 technologies, now web applications are dynamic. Client side should be given due importance as well besides server side processing. Web applications architecture has forced the performance engineers to rethink about the performance testing optimization strategies.

A web application performance can be improved at two levels.

- Back-end/Server side
- Front-end/Client side

Typically, application stakeholders (especially developers) strongly believe that back-end optimization is most important from performance perspectives. Server side bottlenecks are highly important because it can make the web application useless. But it's not the end of world in web application's performance optimization. Client side performance issues are even more critical from performance perspectives because they have more impact on user experience. Improving the back-end performance by 50% improves the overall application performance up to 10% only but application performance can be improved by 40% or more by only reducing the front-end time to half. Moreover, front-end performance optimization is quite simple and cost effective as compared to back-end performance optimization where redesigning application architecture and code, code profiling, adding or modifying hardware, distributing database etc. is required. A study at Yahoo found that on average only 10-20% of total page loading time is spent on the back-end and other 80-90% time is spent on the front-end.



Difference between Front-end performance testing and load testing ?

Front end performance testing is about "How fast does this page load ?" from **a single user point of view**. We also call this activity "Web performance optimization"

Load testing us is about "How fast does this page load when 1000 users are working on the application?" that is from **a multiple users point of view** where resources are used concurrently.

To know how much users your application can handle, to validate your business requirements or to test the overall performance of your application on a specific worst case scenario, you need to load test your application using a load testing tool like AgileLoad.

To optimize the rendering speed on the front-end for a single user, AgileLoad can easily be used.

Other specialized tools exist also on the market.

Front-end Performance Testing Tools

Few years ago, it wasn't an easy task for a web developer to figure out what was actually happening after user submits the request on a browser. But these days there are various tools available online which can help in identifying all the activities as the user hits the enter button in address bar. These tools,

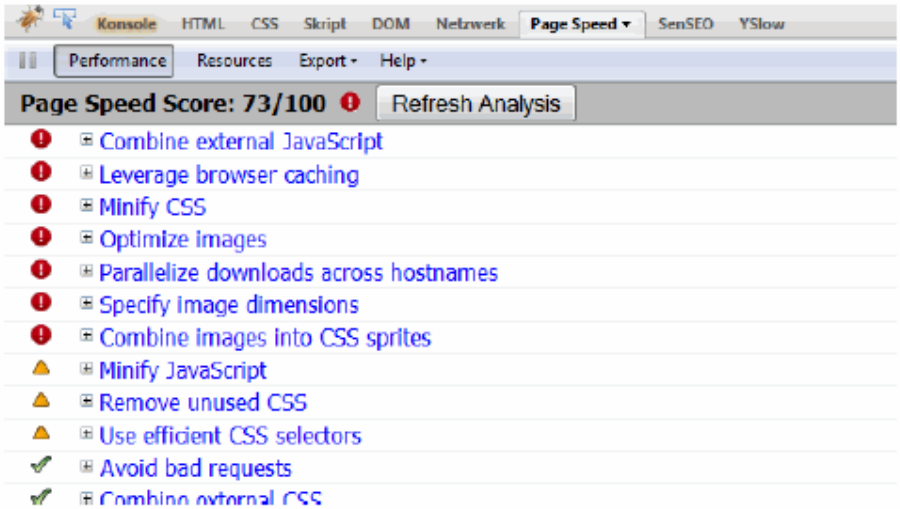
- Grades web page based on one of predefined rule set or a user-defined rule set
- They offer suggestions for improving the web page's performance
- Summarizes the web page's components
- Displays statistics about the web page

Some of these famous Front-end performance testing sources are,

- Page Speed
- Y-Slow
- Firebug
- Web page test
- Yottaa.com

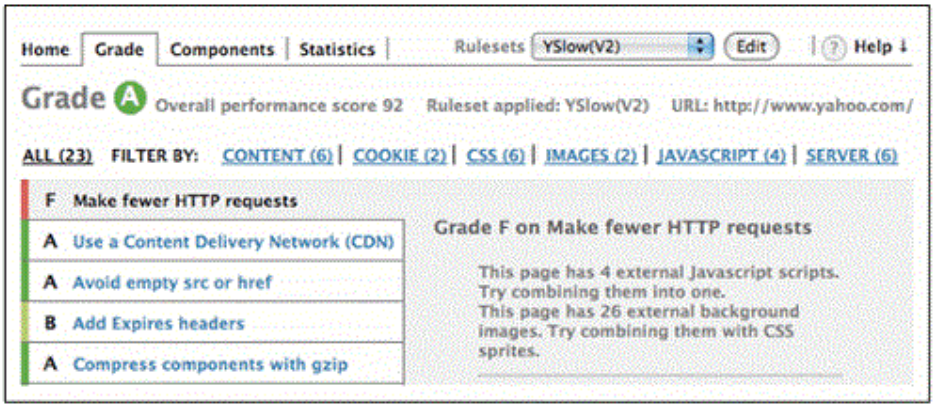
Page Speed

Page speed is an open source Firefox/Firebug ad-on launched by Google that evaluates the web page and provides suggestions to minimize web page loading time. Through this service web pages are fed through a Google server and various algorithms are applied to make them more efficient and fast. It makes web page retrieval faster when users access those pages through Google search engine.



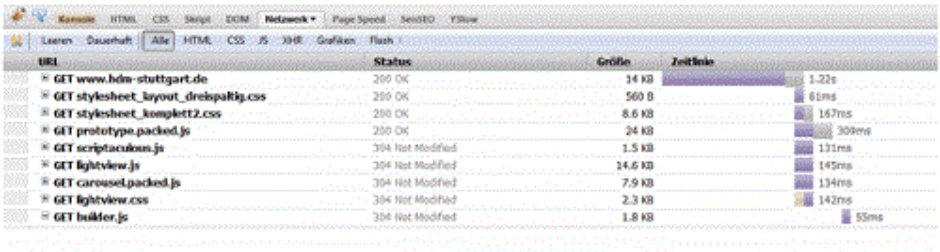
Y-Slow

Yahoo Y-Slow is a browser plug-in which tests the web page against various optimization rules defined by Yahoo performance team and recommend suggestions to optimize the web page.



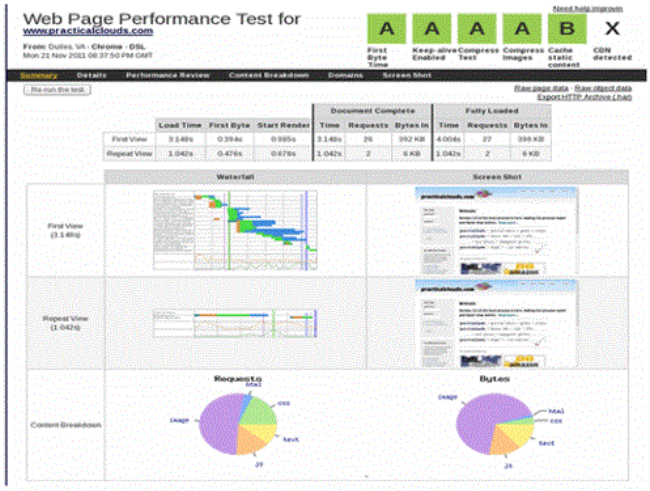
Firebug

Firebug is another browser plug-in which provides various services including debugging of front end development, tracking of all the network requests and profiling JavaScript function calls. Firebug is a favorite tool for most of the developers for client side performance evaluation and profiling.



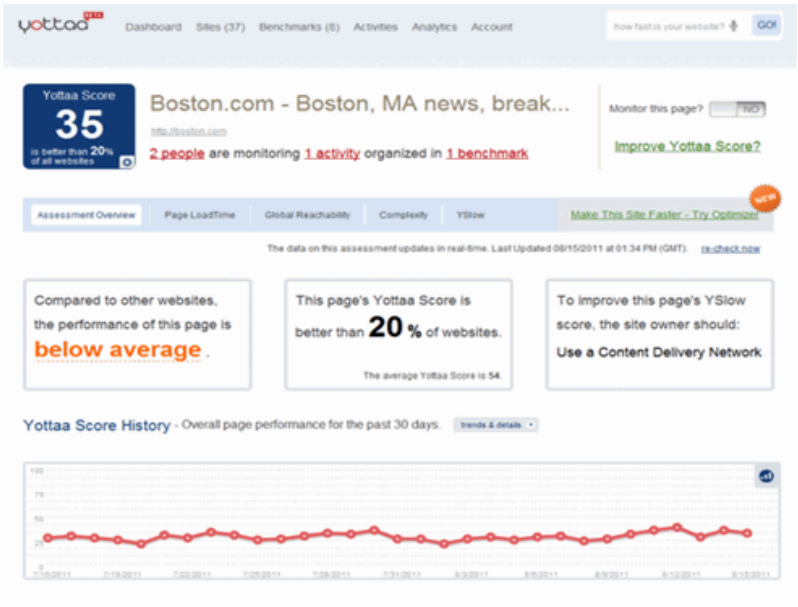
Web Page Test

Web page test is a free online service which provides the website front-end speed test facility. Website speed can be tested on all the famous web and mobile browsers from different geographical locations. It provides detailed information on all the application components which can be really helpful in application optimization.



Yottaa

Yottaa is web optimization solution that provides the web application Yottaa performance score and identifies areas which can contribute most to the application performance.

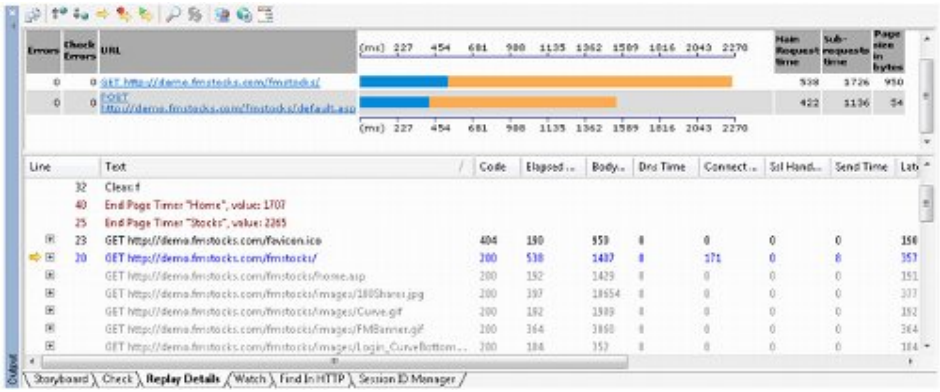


Front end Performance testing with AgileLoad

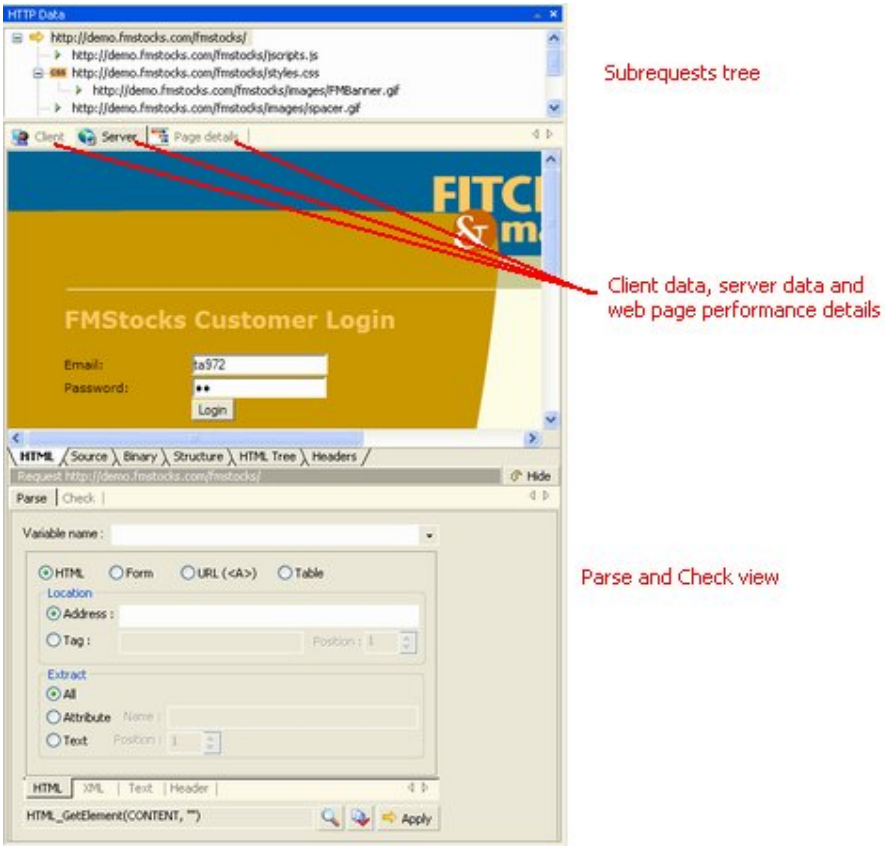
AgileLoad Script Editor captures and analyzes all the requests made between the user and the application to build a test scenario.

The **Replay** function validates the script generated by replaying and comparing each request with the initial scenario.

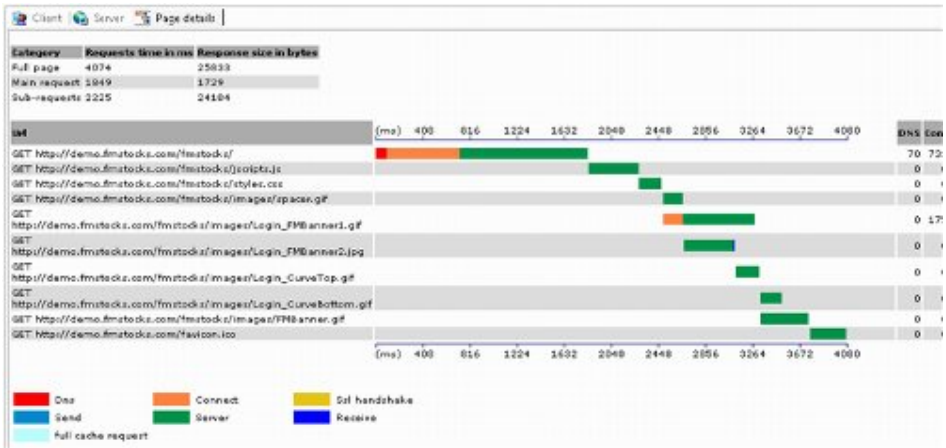
The Replay tab contains for each page of your test scenario a graphical bar chart which shows the time spent for the primary request (in blue) and the overall response time (in orange). It also gives you details of all the resources loaded, the time spent for each resources, the detailed HTTP response (Body, Client HTTP Header, Server HTTP Header) associated with each HTTP request.



For each page, you also retrieve, the HTML view, the source view, structure view, HTML tree view, HTML server headers view



A web page performance summary gives you details on DNS time, tcp connection time, SSL handshake time, the send time, server time, receive time, response http status, response size in bytes ecc..



This page speed waterfall highlights problematic resources to be optimized for each pages of your test scenario.

Front-end Implementation

The front-end of a web application is generally based on thin client architecture. Thin client doesn't process any data; it only presents the user interface (UI) to interact with the application. Similarly in typical web applications client-side contains web browser only. Internet Explorer, Google Chrome, Mozilla Firefox, Apple Safari and Opera are most common web browsers used these days. The browser responsibility is to communicate with the web server over the HTTP protocol, rendering the UI of the web application and allowing user inputs.

In web applications the user interface is generally rendered as an HTML document. This HTML document contains text, input fields, link to other resources, embedded objects and reference to other images, scripts, and style sheets. Web browser retrieves the root HTML document; parse its text and resolves referenced resources.

Afterward scripts are executed, style sheets are processed and other contents are rendered to the user. The execution of these events depends upon the selected browser. All these events can be executed concurrently or step by step by the selected browser.

User input is delivered through HTML elements (which are called forms). Forms are used to collect user inputs through input elements like text fields and check boxes etc. Form data is sent to server on successful form submission by HTTP request and corresponding response is rendered to the user.

Front-end Performance Optimization Techniques

In recent times lots of work has done on client side optimization. Yahoo and Google are pioneer in client side optimization. They not only providing services for front end performance measurements but also define set of rules for optimizing the application client side performance. These rules are now being followed all around to make applications faster. There is a big list of these rules and it will not be easy to cover all of them here, we will try to discuss few most important rules which can be more helpful in optimizing the client side performance.

1. Minimize HTTP Requests

An HTTP request is used to fetch root HTML document that may refer to other page resources like images, scripts and style sheets. Each of these resources must be fetched with every HTTP request. Every HTTP request adds performance overhead as it created network traffic between the client and server. Reducing the number of resources will decrease the HTTP requests required to render the web page and will improve the performance.

One approach to reduce the web page components is to simplify its design but it can affect its look and feel as well. So the best approach is to use the optimal resources but combine them to limit the user response time. Combining all the scripts and style sheets into a single script and style sheet respectively is a challenging task but it will greatly help in achieving the desired goal on performance optimization.

Similarly web page images can also be combined into by using techniques like CSS Sprites, Image maps and Inline Images.

2. Use a Content Delivery Network

This is the era of technology and web applications are being used all around the world. If the application is deployed on a single place, it can greatly affect users accessing the application from longer distance due to network delays. Applications can be deployed over different geographical locations to facilitate the users all over the globe. User response time can be greatly improved by just distributing static web contents on various locations instead of starting from the difficult task of redesigning the application to distribute the dynamic contents.

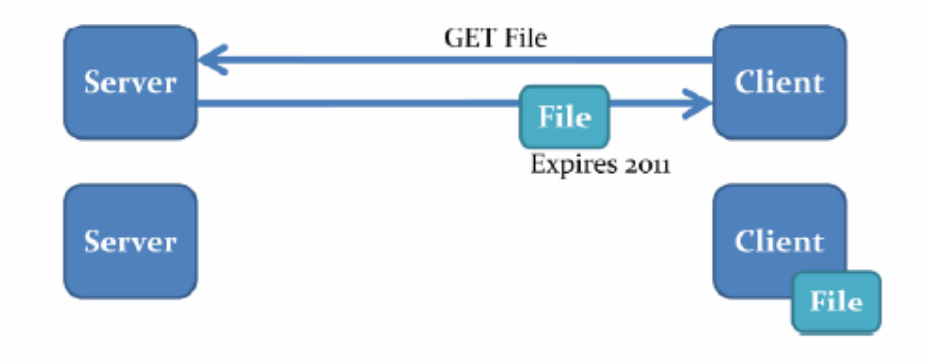
A content delivery network (CDN) is a collection of web servers distributed across various locations to provide web contents in an efficient manner. Based on less number of network hope counts, user request should be entertained from the closest web server.

Some large internet companies have developed their own CDN but it may not be cost effective decision for smaller companies and there are various CDN service providers in market whose services can be used to optimize the end user response time.

3. Add an Expire or Cache Control Header

Browser cache is another source of client side performance optimization. In these days applications are getting richer and richer as they are using various page components like images, style sheets, scripts etc. When first time user visits a web page, it makes lots of HTTP requests to download all the page resources. But he/she doesn't need to download all the resources on visiting the same page again. Today browsers have great ability to cache web page components to reuse them on visiting the same page again instead of requesting to web server for

same resources every time. This feature reduces the web page downloaded resources and HTTP requests as well. It's advised not to use any expiry date for a static component while a web server should uses an expiry header to tell the client how long a dynamic component can be cached.



4. Minify JavaScript and CSS

Removing the unnecessary characters for the code is called minification. Page load time can be optimized by removing all the additional sources like comments, new line commands, meta data, white spaces, new line commands etc. By removing the additional sources web page size is reduced and its download time as well. Obfuscation is another optimization technique applied on source code which even produced better results as compared to minification.

5. GZip Components

Today all modern browsers support compressed components. All the plain text documents like HTML, JS, CSS, and XML etc. can be compressed on server side before transferring to the web browser which will decompress these documents before displaying them to end user. An important point to be noted here is that binary files like images, PDF and SWF should not be compressed again because they are already compressed.

Compressing the already compressed elements will waste CPU utilization and can also increase the file size as well. You don't need to do anything with the code to compress the web page components; compression can be easily enabled on most of the web server through some basic configurations. Following table will show the impact of minification and compression on web pages size.

Source	Original Size	Minified Size	Compressed Size	Minified + Compressed size
HTML	101 KB	97 KB	17 KB	16 KB
JS	243 KB	195 KB	73 KB	63 KB
CSS	90 KB	68 KB	19 KB	14 KB
Total	434 KB	360 KB	109 KB	93 KB

6. Put Style Sheets at the Top

Research at Yahoo discovered putting the style sheets to the document HEAD allows the browser to render progressively and it makes the page loading faster. This method even more useful if the page size is larger. Instead of making the user to wait for rendering of all the page elements and getting bored on white screen, it makes good user experience to display page to see the page components gradually instead of waiting and then viewing all the components suddenly.

Few modern browsers including IE doesn't perform the progressive rendering on web page components on putting the style sheets at the bottom and frustrate the user with blank page.

7. Put Scripts at the Bottom

Downloading multiple sources concurrently is called parallel downloading. Parallel downloading improves the user experience by fetching all the required resources in less time. According to HTTP specification, browser doesn't download more than two components in parallel for a hostname. You can download more than two components concurrently by serving your resources from multiple hostnames. But scripts don't allow parallel downloading. The best solution to this problem is to put the scripts at the bottom. One can also use the DEFFER attribute to allow the browser to do the parallel downloading. But unfortunately Firefox doesn't support the DEFFER attribute and although IE support the DEFFER attribute but even then all the desired results may not be possible. So the best policy is to put the scripts at the bottom to allow all the browsers to perform parallel downloading.

8. Avoid CSS Expression

CSS expressions are used to set the Style sheet property dynamically. An example of the CSS expression can be to set the background color alternate after every 30 minutes. CSS expressions are evaluated very frequently and they are evaluated whenever any user action is performed. They are evaluated when the page is rendered and resized, page scroll down and even on a mouse hover. They are so frequently evaluated even on moving a mouse around the page can generate more than 10,000 evaluations.

One way to handle this situation is just use the one time CSS expression, when the first time expression is evaluated set those values as explicit style sheet values. If there isn't any choice other than using the dynamic style sheet values then use event handlers instead of using the CSS expressions.

9. Make JavaScript and CSS External

As the JavaScript and CSS files are cached by the browser, using them as external files can make the page response time faster. Both of these are in lined in HTML document and downloaded every time HTML document is downloaded which increase the size of the HTML document. Better approach is to make both JavaScript and CSS external files which are cached by the browser. In this way size of the HTML document is reduced but number of HTTP requests remains the same.

10. Image Optimization

These days web pages are made very attractive consists of lots of images. Web page load time can be greatly improved by optimizing these images only. Choosing the appropriate file format will greatly help in this cause. Normally JPG image format is used with high number of colors. PNG is best for rendered text and for images with alpha transparency.

11. Reduce Domain Name System (DNS) Lookups

The DNS maps domain names to IP addresses. DNS normally takes 20-120 milliseconds for DNS to lookup for DNS to lookup the IP address for given domains and browser can't download anything from this hostname unless DNS lookup is completed.

Although DNS lookups are cached for better performance and this DNS information is placed on the operating system's DNS cache. But most browsers have their own cache as well. IE cache the DNS lookup for 30 minutes by default and operating system cache has no use when DNS record exists in browser cache.

Number of DNS lookups will be equal to the unique hosts in the web page in case of empty browser cache. So reducing the number of unique host names will reduce the page load time.

12. Avoid Redirects

Redirects are accomplished by using HTTP status codes of 3xx especially the 301 (Moved permanently) and 302 (Found) status codes. Redirect is an indication that user needs to take some additional actions to complete the request. Main problem with redirects is they slow down the page load time. These redirects took place on various stages like when back slash (/) is not inserted at the end of the URL. Another redirect example is when an old website is connected to new one. Back slash redirects can be fixed in Apache by using Alias.

13. Remove Duplicate Scripts

Usually multiple developers work on a web application development and there is a chance of scripts duplication on web pages. Duplicate scripts really effect the web page performance. Additional execution, resources and HTTP requests will be required for those duplicate scripts which have no role at the end. Duplicate scripts may not have great effect when application is being accessed in Firefox but it really affect IE. Duplicate scripts insertion can be avoided by using the script management module.

14. Turnoff Entity Tags

Entity tags (ETags) are used to validate the browser cache data is updated. ETags compare the browser cached copy with the one on server cache to make sure browser has update data. The ETags has a limitation it only compares the browser cache to a unique server. This technique works well when there is only one hosting server. ETag will not work in a situation where application is hosted on multiple servers and browser gets the components from one server and validate it on another server. Especially ETags generated by IIS and Apache for the same component won't match from one server to another and user receives 200 response code instead of small, fast 304 response of ETag. So the best approach is to turn off the ETags when your application is hosted on multiple servers.

15. Make Ajax Cacheable and Small

One of the cited benefits of Ajax is that it provides instantaneous feedback to the user because it requests information asynchronously from the backend web server. However, using Ajax is no guarantee that the user won't be toying his thumbs waiting for those asynchronous JavaScript and XML responses to return. To improve performance, it's important to optimize these Ajax responses. The most important way to improve the performance of Ajax is to make the responses cacheable.

Conclusion

Web applications are becoming richer and richer in design and content and at the same time good user experience has become the most desirable attribute. There is misconception that application desired response time can be achieved by optimizing server side only. Research has showed that 80-90% of page load time is spend on client side and 40-50% page load time can be optimized by just focusing on front-end of the application as compared to 20% of server side optimization.

Also front-end performance optimization is not the same than back-end optimization. One is about improving the performance from a single user point of view; the other is focused on improving the performance from a multiple user point of view when resources are used concurrently.

Both tasks are complementary and can be tested with Agileload.