UNIVERSITY OF WATERLOO

Faculty of Engineering

Department of Electrical and Computer Engineering

Analysis of Various Web and Search Optimization Methods

QuinStreet, Inc.

Foster City, California

Prepared by

Michael A. Soares

ID [removed]

userid masoares

3A Computer Engineering

11 January 2011

[address removed]

11 January 2011

Dr. Manoj Sachdev, chair
Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario
N2L 3G1

Dear Sir:

This report, entitled "Analysis of Various Web and Search Optimization Methods", was prepared as my 3A Work Report for QuinStreet, Inc. This report is in fulfillment of the course WKRPT 400. The main purpose of this report is to quantitatively analyse the performance improvement of various web and search optimization methods including the optimization of PHP code, the use of file caching, and the use of content delivery networks on one of QuinStreet's web sites in terms of load time, items loaded, and bytes loaded and to offer several recommendations to QuinStreet which may help in improving the performance of some of its other web sites.

I was employed as a PHP Web Developer Intern working alongside several other developers as well as my team's PHP Tech Lead, Jaime Wheeler, who also oversaw my role. My primary responsibilities included fixing web site issues, doing custom PHP and JavaScript development for a variety of QuinStreet's new and existing web sites, as well as working with Mr. Wheeler to improve the performance of several web sites.

I would like to thank Mr. Jaime Wheeler for providing me with encouragement, valuable ideas and information over the work term, which helped me in finalizing this report. I hereby confirm that I have received no further help other than what is mentioned above in writing this report. I also confirm this report has not been previously submitted for academic credit at this or any other academic institution.

Sincerely,

Michael A. Soares
ID [removed]

Contributions

For the past four months, I was employed at QuinStreet, Inc. ("QS") in Foster City, California as a PHP Web Developer Intern. I was one of the many developers on the Financial Services team, which also consisted of a group of producers who managed the content and initiatives on all of QS's finance web sites. I worked and collaborated with several of the other developers and producers on a daily basis, as well as my supervisor, Mr. Jaime Wheeler, dealing mostly with custom PHP: Hypertext Processor ("PHP") and JavaScript development and web site performance improvements. Strategizing with my colleagues for Munchkin, a popular card game, was also a daily activity.

My colleagues and I were primarily responsible for fixing various issues with existing finance web sites and implementing new features and other initiatives on said web sites, whether said features or initiatives were for content or performance. Web sites and new web site features were usually developed using PHP (on the CodeIgniter Model-View-Controller framework), Cascading Style Sheets ("CSS"), and the jQuery JavaScript ("JS") library. All work was done collaboratively through the use of a file repository with source control. Code was thoroughly tested and then documented so that future developers modifying any existing code would not have a difficult time doing so.

While at QS, my primary responsibilities involved working with new and existing code, but more specifically encompassed:

- Developing a high quality life insurance calculator in a relatively short amount of time for use on one of QS's high profile web sites using PHP, CSS, and jQuery,
- Conducting A/B tests on a variety of QS's web sites to measure changes in performance and other analytics,
- Correcting web site bugs/issues submitted by producers, and
- Researching new ways of and implementing a variety of key initiatives for reducing the actual and perceived load times on some of QS's top performing web sites in order to improve user experience and search engine optimization ("SEO").

For the majority of the term and as I already mentioned, I was diligent in aiding my supervisor implement a variety of key initiatives for reducing the actual and perceived load time on some of QS's top performing web sites. The web sites these initiatives were implemented on were excellent candidates since some received upwards of tens of thousands of unique hits per day and a decrease in load time (or increase in performance) would most likely keep visitors on the web site for longer periods of time and most probably attract new visitors as well.

Due to the fact that the implementation of these speed initiatives required me to conduct a variety of different benchmarks, sometimes on the development versions of the web sites in addition to the production ones, the research, implementation phases, and analyses that were done in regards to these speed initiatives proved to be quite suitable for a work term report.

This is the main relationship between this report, the knowledge I gained, and the tasks I performed while working at QuinStreet. The data collected and the analyses performed in this work term report are beneficial to me in many ways, primarily because they have given me the opportunity to learn well beyond what I thought I would as PHP Web Developer Intern. It made me realize that there is more to development than just coding and that the load time of a web site can, over time, make a difference in how visited or unvisited a web site goes on the Internet. This project and this subsequent report have also provided me with the ability to benchmark web site performance and evaluate the resultant quantitative data.

In the broader scheme of things, my research on this report topic should prove to be beneficial for QuinStreet, not only on the Financial Services team, but other teams as well. Since QS has a rather large portfolio of web sites, it must make every effort possible to optimize its largest and fastest growing ones first in order to attract new visitors. In this report, I provide QS with several recommendations on the speed initiatives implemented and which ones reduced the load time of web sites the most (in terms of percentage).

Executive Summary

The main purpose and scope of this report is to qualitatively and quantitatively analyse the key initiatives and methods used to reduce the load time on one of QuinStreet's more popular web sites before and after the change. This report will suggest to QuinStreet which methods are the most optimal in terms of implementation and change, as well as which methods should be looked into more so as to optimize their sites more. I have identified several recommendations in this report that will optimize the performance of QuinStreet's web sites which will allow QuinStreet to most likely generate more revenue by retaining its web site visitors for a longer amount of time due to the reduced load time of its web pages.

The major points in this report are that caching dynamically generated files, using a content delivery network, removing function calling from loop conditions, and forcing string concatenation where possible all reduce the load time of web sites by a reasonable amount. The first section sets out the scope, purpose, and outline of the report. The second section describes the key initiatives and methods QuinStreet used to reduce the load time of one of it's more popular web sites. The third section describes the test tools used to measure the load time on the same web site before and after the implementation of each of the key initiatives already mentioned. The final sections provide conclusions and recommendations based on the analyses in the preceding sections.

The major conclusions of this report will confirm that using a content delivery network does not reduce the load time of a web site as much as caching dynamically generated files does. Enabling file caching for these types of files actually reduces the load time of web site by almost two times than any other method tested. In addition, it is not known what the actual effect will be in production of removing any function calls from within loop conditions or forcing string concatenation, despite the development and production environments being relatively similar to each other, configuration-wise.

Major recommendations in this report are also identified in that QuinStreet should enable the use of file caching across its entire portfolio of web sites. It should also use a content delivery network for its web sites with the ability to cache files to minimize the load and traffic on one server and distributing it across multiple ones, as well as to minimize load time. Lastly, QuinStreet should further investigate whether modifying iterative blocks of PHP code and using string concatenation as is mentioned in this report will have as optimal as an effect in the production environment.

Table of Contents

Contributions iii
Executive Summary v
List of Figures viii
List of Tables ix

1 Introduction 1

1.1 Web and Search Engine Optimization 1
1.2 Purpose 2
1.3 Scope 2
1.4 Outline 2

2 Performance Enhancements 4

2.1 Simple and Effective Speed Optimization Methods 4
2.1.1 Counting in `for`-Loop Conditions 4
2.1.2 Single Quotes vs. Double Quotes in String Concatenation 5
2.2 File Caching 6
2.3 Using a Content Delivery Network 6

3 Quantifying Performance Enhancements 8

3.1 Introduction 8
3.2 Measuring Performance Changes 8
3.3 Load Time Results 9
3.3.1 Using a `count` Variable for Inserting Content Dynamically 9
3.3.2 Forcing the Use of Single Quotes and String Concatenation 10
3.3.3 Enabling File Caching 10
3.3.4 Serving Content Through a Content Delivery Network 11
3.4 Optimal Performance Enhancers 11

4 Conclusions 13

5 Recommendations 14

Glossary 15
References 16
Appendix A – Using a `count` Variable for Inserting Content Dynamically – Test Results 17
Appendix B – Forcing the Use of Single Quotes Where Possible – Test Results 18
Appendix C – Enabling File Caching – Test Results 19
Appendix D – Using a Content Delivery Network – Test Results 21

List of Figures

Figure 1. A generic `for`-loop used to dynamically insert content on a webpage. 4

Figure 2. A generic `for`-loop used to dynamically insert content on a webpage with the applied `count` modification. 5

Figure 3. Examples of string concatenation and string parsing using single and double quotes, respectively. 5

Figure 4. Bar graph showing the load time differences between the 4 methods used. 12

List of Tables

Table 1. Full set of results (including averages) before and after the `count` change was made. 17

Table 2. Full set of results (including averages) before and after the string concatenation change was made. 18

Table 3. Full set of results before file caching was enabled. 19

Table 4. Full set of results after file caching was enabled. 20

Table 5. Full set of results before a content delivery network was used. 21

Table 6. Full set of results after a content delivery network was used. 22

1 Introduction

QuinStreet, Inc. ("QS") is a vertical marketing and online media company which targets such verticals as education, financial services, business-to-business, and travel [1]. QS delivers qualified clicks and inquiries to its clients at a reduced cost which enables its clients to increase their sales with greater scalability. In other words, QS matches qualified visitors to its web sites (i.e., customers) to their clients (e.g., insurance companies).

QS is currently looking into optimizing some of its top performing web sites by reducing the amount of time it takes to load individual web pages; this optimization process is included in the broader scheme of web and search engine optimization ("SEO"). As a result, this will potentially attract more visitors by increasing QS's web sites' search engine ranks and by increasing the average time visitors spend on its web sites [2]. Through various other and unrelated steps, this will help QS generate more revenue in the long run due to the way the company operates (explained above).

In this section, the purpose and scope of the report are both set out and essential background information is presented on the topic.

1.1 Web and Search Engine Optimization

In the context of this report, web and search engine optimization refer to the optimization of a whole web site or individual web pages by reducing the time it takes for content (i.e., images, text, external scripts, etc.) to fully load. However, in general, search engine optimization usually refers to the application of any method used that would cause the rank(s) of a whole web site or individual web pages (or relevance) to increase [3]. Apart from increasing a web site's performance, some other methods known for increasing a web site's rank include, but are not limited to, giving a web site better visibility by linking to it from several other relevant or popular web sites, using relevant keywords in

the file names for web pages or in the content on web sites, and preventing undesirable content from being indexed by a search engine [3].

1.2 Purpose

Because QS generates most of its revenue by selling qualified clicks to its clients, it must be able to collect as many of these as possible without losing them to competitors because of a small number of slow-loading web pages. Optimizing more of QS's web sites will potentially increase the number of visitors each web site gets. Thus, this report will explain several different methods of optimizing web pages, it will analyse the quantitative performance differences of each of the methods, and give some technical reasoning as to which performance-enhancing method will work the best (i.e., attract more visitors based on the reduction in load time).

1.3 Scope

This report will include both qualitative and quantitative analyses of the methods used to reduce the load time of some of QS's web sites and include technical reasoning as to which method will work the best in the long run.

1.4 Outline

The sections in this report identify and summarize the performance-enhancing methods QS's Financial Services team has used on one of its web sites chosen for preliminary web and search engine optimization. This report also provides qualitative and quantitative analyses of the performance-enhancing methods and provides a comparison of each one against every single one of the others. A glossary has also been included for easy reference of technical terms used in this report. Section 2 introduces the main SEO topic discussed in this report (i.e., speed) and the related performance-enhancing methods and outlines some of their key features. Section 3 describes the tools used to gather the resultant data. It also outlines and explains the outcomes of applying each of the

performance-enhancing methods to several of QS's similarly structured web sites and attempts to provide justifications for the results obtained. Finally, conclusions and recommendations are outlined at the end of the report.

2 Performance Enhancements

2.1 Simple and Effective Speed Optimization Methods

Some simple and effective speed optimization methods that were implemented as a part of QS's key speed initiatives included conducting several, but very informal code reviews of existing PHP code. Most of these code reviews looked at the way content was being inserted onto web pages (i.e., whether it was being inserted dynamically or statically). These optimization methods are presented in this section, separate from the others in 2.2 and 2.3, as they did not require changes that were overly complex.

2.1.1 Counting in `for`-Loop Conditions

In most of the cases that content was being inserted dynamically, the code that had been originally written required iterating through arrays containing strings of content in a way resembling that in Figure 1 (below).

```
<?php
	// ...
	for($i = 1; $i < count($content_array); ++$i) {
		// insert content
	}
	// ...
?>
```

Figure 1. A generic `for`-loop used to dynamically insert content on a webpage.

If content was being inserted dynamically using a `for`-loop similar to that in Figure 1 (above) using the `count` function in the condition, the value of `count($content_array)` would have to be calculated on each iteration of the loop even though the variable `$content_array` was not being changed. What this meant for the webpage this code was used on was that valuable processor time was being wasted on the server side in order to simply recalculate said value over and over again instead of simply storing it in a temporary variable, such as in Figure 2.

```
<?php
      // ...
      $content_count = count($content_array);
      for($i = 1; $i < $content_count; ++$i) {
            // insert content
      }
      // ...
?>
```

Figure 2. A generic `for`-loop used to dynamically insert content on a webpage with the applied `count` modification.

The results of moving the `count` function out of the `for`-loop's condition and into a separate variable like in Figure 2 are discussed in detail 3.3.1.

2.1.2 Single Quotes vs. Double Quotes in String Concatenation

One thing that some PHP developers do not often realize is the difference between using single quotes and double quotes when dealing with strings. Because PHP is a parsed language, that is, PHP code is read and executed without doing any pre-compilation of said code, optimizations on strings cannot be done at the time that code is executed. As a result, strings sometimes use more memory when combined using surrounding double quotes instead of single quotes and string concatenation [4]. Strings surrounded by double quotes have their variables parsed instead of being simply concatenated when using single quotes, demonstrated in the example in Figure 3 below.

```
<?php
      $var1 = 'string';
      $var2 = 'this is a '.$var.' with single quotes';
      $var3 = "this is a $var with double quotes";
?>
```

Figure 3. Examples of string concatenation and string parsing using single and double quotes, respectively.

The results of conducting a code review on one of QS's web sites and converting double-quoted strings to single-quoted strings like in Figure 3, where possible, is discussed in 3.

2.2 File Caching

In the context of web development, file caching refers to the pre-generated and subsequent serving of any non-media and web-related files (e.g., HTML, JavaScript, CSS, etc.), which may have otherwise been dynamically generated on each page load. Take a HTML webpage dynamically generated using PHP on each page load as an example. If that page were to make several calls to an external database and then have to do something with said data, it could take several seconds before the page could actually be generated and served up to the user accessing it. A caching tool used for file caching pre-generates these dynamic pages by means of a single page load or some other trigger, stores them temporarily until they expire, and then serves the same pre-generated file up to multiple users. This helps speed up page loads tremendously and reduces the load on a server and database by making fewer database calls and requiring dynamic pages to be generated fewer times than usual.

2.3 Using a Content Delivery Network

A content delivery network ("CDN") is a group or network of computers distributed throughout an area (e.g., various points in North America and the world) used to maximize the bandwidth used by clients accessing data on said computers. As opposed to accessing data on a centralized server, clients accessing a web site making use of a CDN will be able to retrieve the same data from a server that is located closer to their geographical location, reducing the time it takes for data to be transferred to their computer and eliminating the possible bottleneck created when using a single centralized server.

CDNs, such as those used QS, have the ability to cache files as well, as explained in 2.3. Thus, retrieving data from a nearby server that has already pre-generated a requested

webpage can be increasingly faster than by just serving files from a CDN with file caching disabled, as will be shown in 3.

3 Quantifying Performance Enhancements

3.1 Introduction

Before having measured the performance enhancements of the various methods presented in 2, it was initially believed that using a CDN with the ability to cache files would increase the performance (i.e., reduce the load time) on one of QS's web sites the most. This was based on the following facts:

1) The primary purpose of a CDN is to minimize the amount of traffic going to any one server by redirecting any one request to a server that is closest to the user making said request, and
2) Caching dynamic pages for long enough periods of time where data does not change will significantly reduce the master server's load.

3.2 Measuring Performance Changes

After implementing QuinStreet's key speed initiatives, the load times of several web pages were measured using two tools. For smaller changes that could be tested in a local development environment, a free tool called Hammerhead for the Mozilla Firefox browser was used to find the difference in load times with the browser cache disabled [5]. This tool would render each webpage locally within the browser over any set of iterations. This only gave developers a rough idea of how the changes affected performance and whether the changes should be put into production or not.

The second tool used was a web-based tool called Keynote [6]. Since this tool is run remotely, it gives developers a broader idea of how their changes affect performance. Keynote simply runs a load test against any web site from multiple locations in the world that one can choose and sends back the results from its servers. Keynote was usually run after larger performance-affecting changes were made in the production environment (i.e., 20% change or more using Hammerhead in the development environment).

3.3 Load Time Results

As presented in 2, 4 different performance-enhancing methods were implemented on one of QS's content-heavy web sites. The load time changes for each method, whether it was implemented solely in the development environment or in the live/production environment, are presented here. Those that were implemented solely in the development environment were tested using the Hammerhead tool presented in 3.2, whereas those in the live/production environment were tested using Keynote.

3.3.1 Using a `count` Variable for Inserting Content Dynamically

After moving the `count` function out of all `for`-loop conditions (where found) and into separate variables as discussed in 2.1.1 on one of QS's web sites, the load time of a content-heavy page in the development environment was reduced from an average of 4.29 seconds to an average of 3.58 seconds, or by 18.04%, after using Hammerhead to reload the page with an empty cache 20 times (see Appendix A for the full set of results). Thus, a change as small and as simple as this can make a reasonable difference on content-heavy web pages.

The results of this change do not come as a surprise. Suppose a page, similar to the ones on QS's web site, being requested had m number of `for`-loops in which content was being dynamically inserted and each of those loops iterated approximately n number of times for each piece of content being inserted with `count($content_array)` being called in each of their conditions, where $m \ll n$. In such a case, and similar to the pages that were actually tested, the runtime for each page generation would be $O(mn)$ in big-O notation, or simply $O(n)$ since $m \ll n$. After implementing the change presented here, the runtime was reduced to $O(m) = O(1)$ since part of the condition (i.e., `count($content_array)`) was moved outside of the loops and replaced with a single variable, calculated once per loop instead of n times.

3.3.2 Forcing the Use of Single Quotes and String Concatenation

Tracking down the use of double quotes where they were being unnecessarily used was not very difficult to accomplish. A simple search and replace of double-quoted strings and quick analysis of the code was all that was needed to implement this change. Using string concatenation with single quotes as opposed to string parsing in strings with variables surrounded by double quotes reduced the load time of a content-heavy webpage in a development environment from an average of 3.58 seconds to an average of 3.09 seconds, or by 14.69%, after using Hammerhead to reload the page with an empty cache 20 times (see Appendix B for the full set of results).

The technical explanation for this reduced load time can be found in the opcodes generated by PHP when going through each of the strings. The resultant opcodes generated by PHP when variables are inserted into strings surrounded by double quotes is much longer than that of single-quoted strings being concatenated with other variables and thus causes the processor to interpret more instructions than should actually be necessary [7].

3.3.3 Enabling File Caching

Prior to enabling file caching for some JS and CSS files on one of QS's web sites, the same JS and CSS files were being minified (i.e., whitespace and comments were removed) and concatenated together so as to reduce the number of requests a browser would have to make and to reduce the number of bytes being downloaded. This process took quite upwards of several seconds, thus file caching had to be turned on to reduce that number. After doing so in the production environment, the web site performed extremely well. The average load time on the site had been reduced by an average of 31.8% (measured using Keynote) where the site was accessed from different points across the United States, with the highest reduction being the load time from an area in San Francisco, California at 50.9% (see Appendix C for the full set of results).

Like the method presented in 3.3.1, this also did not come as much of a surprise. The minification and concatenation process, as previously explained, is what took the most time to do. Performing this once, caching the minified and concatenated files, and then serving those files when requested definitely improved performance by quite a bit. Had files already been static before file caching was turned on, this method would have had little to no effect.

3.3.4 Serving Content Through a Content Delivery Network

Serving content through a CDN must be done with extreme caution. In the case of the QS web site at hand, the content on it was updated at most twice per day. Since content was only being dynamically inserted and not dynamically generated throughout the day, it was assumed that no major problems would be encountered. By making use of a CDN with the ability to do site-wide file caching (i.e., including HTML files as well as CSS and JS ones), the site load time was reduced by an average of 18.64% (measured using Keynote), with the highest reduction being the load time from an area in Dallas, Texas at 28.41% (see Appendix D for the full set of results).

3.4 Optimal Performance Enhancers

It was expected that the reduction in load time by the CDN would be the most out of all of the methods implemented and presented in 2. Obviously, this was not the case; file caching of the JS and CSS files outperformed the usage of the CDN by almost twice as much. However, using the CDN was quite successful in further reducing the load time of the web site by quite a bit. Overall, each performance method did a relatively good job at reducing the load time on the web site. 3 out of the 4 methods presented reduced the load time by more or less the same amount (percentage-wise) despite the small differences between the development and production environments, as can be seen in Figure 4 on the following page.

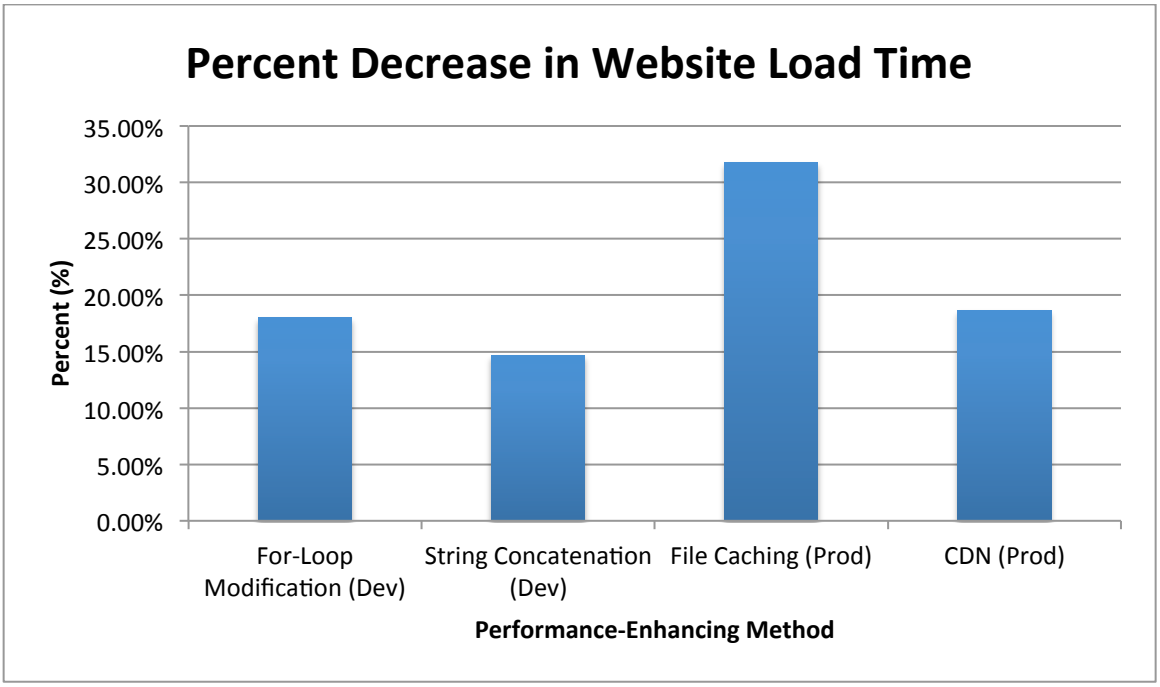

Figure 4. Bar graph showing the load time differences between the 4 methods used.

Thus, as can be seen in Figure 4 above, the best method to use when wanting to minimize the amount of time necessary to load a webpage is file caching when dealing with dynamically generated page as was the case here, followed by using a CDN. The 2 other methods can be implemented at the discretion of the developer, although they do offer quite an enhancement to an already slow web site.

4 Conclusions

From the analysis in the report body, it is concluded that the implementation of a file cache on any web site or webpage where files are being dynamically generated or modified unnecessarily by PHP code will result in reduced load time for said web site or webpage. Doing so will also reduce the server load, including the processor time and memory used. Enabling a file cache where only static files are served up to a user will have little to no effect.

Apart from file caching, using a content delivery network to serve files will also deliver a performance boost, though not as much as that when enabling file caching.

Lastly, while modifying `for`-loops to exclude a function from being called from within a condition as well as using string concatenation and single quotes will help boost performance in a development environment, it is not known what the actual effect will be in production, despite the development and production environments being relatively similar to each other, configuration-wise.

5 Recommendations

Based on the analysis and conclusions put forth in this report, it is recommended that QuinStreet implement the following recommendations across their entire portfolio of web sites:

1) Enable the use of file caching across all web sites, keeping in mind that this will only affect dynamically changing files the most.
2) Use a content delivery network with the ability to cache files to minimize the load and traffic on one server and distributing it across multiple ones, as well as to minimize load time.
3) Further investigate whether modifying iterative blocks of PHP code (i.e., `for`-loops) and using string concatenation as mentioned in this report will have as optimal as an effect in the production environment as was found and analysed in the development environment.

By implementing one or more of the above recommendations, these performance-enhancing methods are sure to help QuinStreet receive more visitors to their web sites and, as a result, allow them to generate more revenue than they are currently generating. QuinStreet will be able to benefit from optimized web sites the most by caching their web sites' files and using CDNs where possible.

Glossary

Cache – A cache transparently stores data so that any subsequent requests for said data will result in it being served faster.

CDN – An acronym for "Content Delivery Network" – A content delivery network is a group or network of computers distributed throughout an area (e.g., various points in North America and the world) used to maximize the bandwidth used by clients accessing data on said computers.

CSS – An acronym for "Cascading Style Sheets" – CSS is a style sheet language that is used to format and present a document written in a markup language such as HTML.

HTML – An acronym for "HyperText Markup Language" – HTML is a markup language predominantly used for web pages.

JS – An acronym for "JavaScript" **–** In the context of this report, JavaScript refers to the client-side code that is run by a JavaScript-capable browser after a web page has fully loaded.

Minification – Minification is the process of removing all unnecessary source code (i.e., white space, comments, etc.) without changing the functionality of the surrounding code.

Opcodes – Short for "Operation Codes" – Opcodes are the machine language instructions that specify what operations the machine at hand must perform.

PHP – An acronym for "PHP: Hypertext Preprocessor" – A hypertext preprocessor for interpreting and rendering coded web applications.

SEO – An acronym for "Search Engine Optimization" – The process of improving the ranking or visibility of a web site in search engines' results by means of reducing load time, writing articles with unique content, including keywords in on a web page, etc.

Server Load – For single-processor systems, the server load can be thought of as a percentage of system utilization over a period of time. For a system with multiple processors or processor cores, one must divide the number by the number of total processor cores, and it too represents system utilization over a period of time.

References

[1] QuinStreet, Inc., "What We Do | QuinStreet," 2009. [Online]. Available: http://www.quinstreet.com/what_we_do/. [Accessed: Jan. 5, 2011].

[2] A. Singhal, "Using site speed in web search ranking," 2010. [Online]. Available: http://googlewebmastercentral.blogspot.com/2010/04/using-site-speed-in-web-search-ranking.html. [Accessed: Jan. 6, 2011].

[3] S. Nelson and J. Simek. "Optimizing Your Web Site: The ABC's of SEO". *Law Practice Management,* 34, pp. 18-21, April/May 2008.

[4] The PHP Group, "PHP: Strings - Manual," 2011. [Online]. Available: http://ca.php.net/types.string/. [Accessed: Jan. 7, 2011].

[5] S. Souders, "Hammerhead," 2008. [Online]. Available: http://stevesouders.com/hammerhead/. [Accessed: Jan. 7, 2011].

[6] Keynote Systems, "Web Load Testing Products," 2011. [Online]. Available: http://www.keynote.com/products/web_load_testing/. [Accessed: Jan. 7, 2011].

[7] Zvonko, "PHP Myth Busters: Using single quotes on string is faster then double quotes," 2010. [Online]. Available: http://www.codeforest.net/php-myth-busters-using-single-quotes-on-string-is-faster-then-double-quotes/. [Accessed: Jan. 8, 2011].

Appendix A – Using a `count` Variable for Inserting Content Dynamically – Test Results

The full set of test results, including the averages, for the load times collected after 20 iterations before and after the `count` change was made is included in Table 1 below.

Table 1. Full set of results (including averages) before and after the `count` change was made.

Iteration #	Load Time Before Change (s)	Load Time After Change (s)
1	5.23	3.34
2	3.34	3.33
3	4.52	4.07
4	4.35	3.32
5	4.22	3.87
6	3.89	3.54
7	4.38	3.96
8	3.45	5.23
9	4.23	2.87
10	4.34	4.23
11	4.46	2.34
12	5.07	3.52
13	4.32	3.35
14	4.87	3.43
15	4.21	4.33
16	4.25	3.38
17	4.19	2.45
18	3.87	3.23
19	4.2	3.2
20	4.39	4.58
Average	4.29	3.58

The iteration numbers in Table 1 above do not actually directly correlate with the results from before and after the change. They are included solely to differentiate between the results. A full analysis of the results above can be found in 3.3.1.

Appendix B – Forcing the Use of Single Quotes Where Possible – Test Results

The full set of test results, including the averages, for the load times collected after 20 iterations before and after the string concatenation change was made is included in Table 2 below.

Table 2. Full set of results (including averages) before and after the string concatenation change was made.

Iteration #	Load Time Before Change (s)	Load Time After Change (s)
1	3.76	2.89
2	3.2	3.01
3	4.03	3.73
4	3.21	3.23
5	3.35	3.12
6	3.54	2.86
7	3.53	3.1
8	2.89	4.01
9	2.87	2.76
10	4.23	3.97
11	5.23	2.34
12	3.39	2.55
13	3.35	3.21
14	3.15	3.25
15	4.23	3.54
16	3.52	2.96
17	3.43	2.33
18	3.05	3.32
19	4.55	2.65
20	3.11	3.01
Average	3.58	3.09

The iteration numbers in Table 2 above do not actually directly correlate with the results from before and after the change. They are included solely to differentiate between the results. A full analysis of the results above can be found in 3.3.2.

Appendix C – Enabling File Caching – Test Results

The full set of results for the load times, bytes downloaded, and DNS request time using Keynote before file caching was enabled is included in Table 3 below.

Table 3. Full set of results before file caching was enabled.

Locations:	Seattle Qwest	San Francisco AT&T	New York Cogent	Los Angeles Cogent	Dallas SBC	Chicago AT&T	Atlanta Sprint
Component	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)
Before (2010-09-28 11:45 AM PST)							
DNS Lookup	0.097	0.161	0.265	0.175	0.325	0.224	0.277
Initial Connection	0.024	0.014	0.217	0.098	0.049	0.059	0.086
SSL	0	0	0	0	0	0	0
Redirection	0	0	0	0	0	0	0
Request Time	0	0	0	0	0	0	0
First Byte Download	0.036	0.093	0.217	0.215	0.06	0.074	0.111
Base Page Download	0.279	0.208	1.114	0.318	0.447	0.25	0.234
Client Time	0	0	0	0	0	0	0
Content Download	3.275	3.875	17.075	11.827	5.338	5.728	7.361
Total Time	3.711	4.351	18.888	12.633	6.219	6.335	8.069
Avg. Bytes Downloaded	414150	414312	414149	414302	414296	414005	414294

A full analysis of the results above can be found in 3.3.3.

The full set of results for the load times, bytes downloaded, and DNS request time using Keynote after file caching was enabled is included in Table 4 below.

Table 4. Full set of results after file caching was enabled.

Locations:	Seattle Qwest	San Francisco AT&T	New York Cogent	Los Angeles Cogent	Dallas SBC	Chicago AT&T	Atlanta Sprint
Component	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)
After (2010-09-28 12:00 PM PST)							
DNS Lookup	0.098	0.055	0.219	0.07	0.288	0.18	0.365
Initial Connection	0.023	0.015	0.094	0.129	0.05	0.06	0.085
SSL	0	0	0	0	0	0	0
Redirection	0	0	0	0	0	0	0
Request Time	0	0	0	0	0	0	0
First Byte Download	0.038	0.13	0.212	0.23	0.165	0.074	0.347
Base Page Download	0.169	0.141	0.378	0.343	0.233	0.26	0.283
Client Time	0	0	0	0	0	0	0
Content Download	2.415	1.794	9.219	6.829	3.589	4.153	7.135
Total Time	2.743	2.135	10.122	7.601	4.325	4.727	8.215
Avg. Bytes Downloaded	404111	404174	404201	404187	403913	403877	404241

A full analysis of the results above can be found in 3.3.3.

Appendix D – Using a Content Delivery Network – Test Results

The full set of results for the load times, bytes downloaded, and DNS request time using Keynote after file caching was enabled is included in Table 5 below.

Table 5. Full set of results before a content delivery network was used.

Locations:	Seattle Qwest	San Francisco AT&T	New York Cogent	Los Angeles Cogent	Dallas SBC	Chicago AT&T	Atlanta Sprint
Component	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)
Before (2010-10-20 2:00 PM PST)							
DNS Lookup	0.003	0.012	0.012	0.02	0.018	0.017	0.018
Initial Connection	0.024	0.013	0.084	0.019	0.05	0.061	0.085
SSL	0	0	0	0	0	0	0
Redirection	0	0	0	0	0	0	0
Request Time	0	0	0	0	0.001	0	0.001
First Byte Download	0.049	0.035	0.106	0.029	0.061	0.085	0.099
Base Page Download	0.146	0.237	0.235	0.204	0.394	0.229	0.285
Client Time	0	0	0	0	0	0	0
Content Download	6.301	2.117	5.51	1.82	7.233	3.989	5.635
Total Time	6.523	2.414	5.947	2.092	7.757	4.381	6.123
Avg. Bytes Downloaded	381213	381088	380960	381207	380855	380987	380939

A full analysis of the results above can be found in 3.3.4.

The full set of results for the load times, bytes downloaded, and DNS request time using Keynote after file caching was enabled is included in Table 6 below.

Table 6. Full set of results after a content delivery network was used.

Locations:	Seattle Qwest	San Francisco AT&T	New York Cogent	Los Angeles Cogent	Dallas SBC	Chicago AT&T	Atlanta Sprint
Component	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)
After (2010-11-29 3:00 PM PST)							
DNS Lookup	0.192	0.051	0.205	0.071	0.195	0.202	0.216
Initial Connection	0.002	0.004	0.001	0.001	0.003	0.003	0.002
SSL	0	0	0	0	0	0	0
Redirection	0	0	0	0	0	0	0
Request Time	0	0	0.001	0.001	0	0	0
First Byte Download	0.432	0.529	0.555	0.429	0.356	0.486	0.515
Base Page Download	0.023	0.015	0.091	0.019	0.048	0.006	0.086
Client Time	0	0	0	0	0	0	0
Content Download	1.254	1.117	1.565	1.268	1.534	1.316	1.482
Total Time	1.903	1.716	2.418	1.789	2.136	2.013	2.301
Avg. Bytes Downloaded	325983	325927	325858	325803	325756	326036	325921

A full analysis of the results above can be found in 3.3.4.