AN ANALYSIS OF TECHNIQUES AND QUALITY ASSESSMENT FOR WEB PERFORMANCE OPTIMIZATION

Shailesh K S
Research Scholar, SOCIS, IGNOU, Maidan Garhi, Delhi - 110068
Shailesh.shivakumar@gmail.com

Dr. P. V. Suresh
Director, SOCIS, IGNOU, Maidan Garhi, Delhi - 110068
pvsuresh@ignou.ac.in

Abstract - Performance of web applications impact the user satisfaction, online revenue, search engine ranking, site traffic, SEO and other such key metrics. In this paper we have conducted the survey of state of the art tools, techniques, methodologies of various aspects of web performance optimization. We have elaborated key aspects of pro-active quality in web applications. We have elaborated on various techniques related to WPO such as asset optimization, monitoring and testing.

1. INTRODUCTION TO WEB PERFORMANCE OPTIMIZATION (WPO)

Web performance impacts the revenue [2] and ranking of the web pages [1], It also impacts the online revenue [56] [57]. Site performance also impacts the success rate [71].

WPO involves best practices and techniques to increase the speed of web pages [87]. WPO includes page components such as HTML content, presentation components, page elements, page assets and such. WPO provides techniques, best practices, thumb rules, methodologies for end-to-end web performance optimization.

1.1 Impact of WPO

WPO has impact in following aspects:

- Customer churn: Research indicates that customers would abandon the slower web pages [88] [91] [92]
- User impact: User experience is drastically impacted due to page performance
- Site Traffic: Site traffic is impacted if the page takes more than 3 seconds [89] and most users expect the page to load within 2 seconds [89].
- Revenue: For e-commerce sites, the shopping and check out performance is crucial for the conversion
- Multi-device optimization: An optimized web page also impacts the performance on various devices

1.2 Dimensions of WPO

The process of Web optimization can be analyzed from several dimensions:

- **Optimization of Request pipeline processing Systems:** In this category we will examine all the systems involved in the web request processing pipeline. This involves browser software, CDN, proxy server, network, load balancer, web server, application server, integration middleware, backend services, database server and such
- **Client-side and server side optimization:** Client-side performance optimization includes all optimizations performed on client-side presentation components such as HTML pages, images, assets and such. Server side optimization includes performance tuning of server –side components such as fine-tuning business components, setting optimal server configuration, right infrastructure sizing and such
- **Design time and run time optimization:** Design time optimization includes the static and offline performance optimizations activities such as performance code reviews, performance testing, and offline performance tuning and such. Run time and dynamic performance optimization activities include real-time performance monitoring and notification, run time performance optimization and such.
- **Web component optimization:** Another aspect of web optimization is to optimize each of the constituent’s web components such as HTML, images, JavaScript, CSS, Rich media files and such.
2. THE ROLE OF SECURITY IN WEB PERFORMANCE OPTIMIZATION

In this section we briefly discuss the impact of security on performance.

2.1 Security & WPO

Security is one of the key concerns for web applications. Security for web applications can be enforced at various levels. One of the most commonly used security constraint is to use secured socket layer (SSL) to ensure transport level security. SSL is a default choice for web pages hosting confidential information such as user credentials, user personal information and such. SSL would also impact the page performance [78] due to additional overhead. The most commonly used techniques for optimizing performance in such scenarios are as follows:

- Set proper expiration times for the objects so that browser can cache the objects appropriately [74]
- Use CDN which support SSL acceleration modules for forward caching

3. WEB PERFORMANCE DESIGN CHECKLIST

In this section, we will look at the design guidelines and best practices for achieving optimal web performance. The books [66] [67] provide excellent performance guidelines from web performance stand point.

Table 1: Categorized Performance Rules

<table>
<thead>
<tr>
<th>Category</th>
<th>Performance Rule</th>
<th>Impact on web performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Optimization</td>
<td>Reduce the number of HTTP Requests</td>
<td>Reduces the consumed bandwidth and data transferred</td>
</tr>
<tr>
<td></td>
<td>Merge the static assets such as JS and CSS files</td>
<td>Merging would reduce the number of HTTP requests and would improve the page response times by about 38% [66]</td>
</tr>
<tr>
<td></td>
<td>Remove all duplicate file includes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Remove all invalid URLs which result in HTTP 404</td>
<td>Avoids unnecessary and invalid HTTP requests</td>
</tr>
<tr>
<td></td>
<td>Load the JavaScripts asynchroneously</td>
<td>This would reduce the blocked loading of JS files</td>
</tr>
<tr>
<td></td>
<td>Minimize usage of iframes</td>
<td>iFrames block the loading of parent window till iframe source is loaded and hence affects load times</td>
</tr>
<tr>
<td></td>
<td>Minimize redirects</td>
<td>Minimize additional requests</td>
</tr>
<tr>
<td></td>
<td>Cache DNS records</td>
<td>We can reduce the DNS lookup time through DNS cache maintained at browser level</td>
</tr>
<tr>
<td></td>
<td>Remove any unused CSS, JS file includes</td>
<td>Minimizes HTTP requests</td>
</tr>
<tr>
<td>Web object size optimization</td>
<td>Minify JS and CSS files</td>
<td>Minification would reduce the size of JS and CSS file</td>
</tr>
<tr>
<td></td>
<td>Compress images</td>
<td>Compressed images would reduce overall page size</td>
</tr>
<tr>
<td></td>
<td>Leverage gzip for HTTP compression</td>
<td>Compression would reduce the overall page size by about 70% [66]</td>
</tr>
<tr>
<td></td>
<td>Remove white space in the HTML document</td>
<td>Removal of white space would reduce the overall page size</td>
</tr>
<tr>
<td>HTTP Header Optimization</td>
<td>Leverage cache headers for static assets (images, JS, CSS, JSON and other binary files) using Cache-Control header with max-age directive</td>
<td>Allows browsers to optimally cache the assets</td>
</tr>
<tr>
<td></td>
<td>Use expires header for the assets</td>
<td>Avoids additional resource request</td>
</tr>
<tr>
<td>Asset placement</td>
<td>Place CSS files at the top</td>
<td>CSS elements in the head tag</td>
</tr>
<tr>
<td></td>
<td>Place JS files at the bottom</td>
<td>JS files at the bottom would improve the perceived page load time. I would</td>
</tr>
<tr>
<td>Network optimization</td>
<td>Usage of CDN</td>
<td>CDN would optimize the resource request by serving the resource from nearest location to the requestor</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Usage of CDN</td>
<td>Allows browsers to download the content in parallel.</td>
</tr>
<tr>
<td>External Dependency optimization</td>
<td>Identify all external scripts and HTTP requests which impact the page performance and which block the page load and optimize them</td>
<td></td>
</tr>
</tbody>
</table>
| Web Application design optimization | • Perform regular and iterative performance testing to identify performance bottlenecks and fix them. Use automated and manual performance code reviews at regular intervals.  
• Use light-weight service based integration model and load the data asynchronously on-demand | Iterative performance testing uncovers performance bottleneck during early stages |
| Testing Type | Open Source Tool(s) |  |
| Web testing | • Selenium ([http://www.seleniumhq.org/](http://www.seleniumhq.org/))  
• Crawljax  
• Cucumber  
• HtmlUnit |  |
| Accessibility Testing | • W3c Accessibility validator ([http://www.w3.org/standards/webdesign/accessibility](http://www.w3.org/standards/webdesign/accessibility)) |  |
| HTML Compliance Testing | • W3c validator ([http://validator.w3.org/](http://validator.w3.org/)) |  |
| Security Testing | • OWASP.org web security testing tools |  |

3.1 Tools for web performance optimization

Table 3. below provides a list of tools that can be used for web performance optimization

<table>
<thead>
<tr>
<th>Testing Type</th>
<th>Open Source Tool(s)</th>
</tr>
</thead>
</table>
| Web testing  | • Selenium ([http://www.seleniumhq.org/](http://www.seleniumhq.org/))  
• Crawljax  
• Cucumber  
• HtmlUnit |  |
| Accessibility Testing | • W3c Accessibility validator ([http://www.w3.org/standards/webdesign/accessibility](http://www.w3.org/standards/webdesign/accessibility)) |  |
| HTML Compliance Testing | • W3c validator ([http://validator.w3.org/](http://validator.w3.org/)) |  |
| Security Testing | • OWASP.org web security testing tools |  |
### Web Performance Analysis
- Google PageSpeed (https://developers.google.com/speed/pagespeed/)
- Yahoo Yslow (http://developer.yahoo.com/yslow/)

### Performance Testing
- Jmeter (http://jmeter.apache.org/)
- LoadUI

### Static Code Analysis Tools
- Checkstyle (http://checkstyle.sourceforge.net/)
- PMD (http://pmd.sourceforge.net/)
- Findbugs (http://findbugs.sourceforge.net/)
- SonarSource (http://www.sonarsource.com/)

### Continuous Integration Tools
- Jenkins (http://jenkins-ci.org/)
- Hudson (http://hudson-ci.org/)

### Profiling Tool
- Jprofiler

### Service Testing
- SOAPUI (http://www.soapui.org/)

## 4. Asset Optimization

Static assets such as images, multi-media files and other binary files contribute to the web performance. On an average 70-80% of page contains non-HTML content such as images, JS, CSS [58] [66]. Hence optimizing the digital assets would be crucial for optimizing the performance of the overall web page. The books [66] [67] provides following techniques for optimizing the asset performance.

Page assets apart from the page HTML include the image graphics, media files (videos, flash), JavaScript files, Stylesheet files (CSS files) and JSON based data.

Let us look at optimizing each of these assets in table 4:

### Table 2: Asset Optimization Techniques

<table>
<thead>
<tr>
<th>Asset Category</th>
<th>Optimization Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Images</strong></td>
<td>• Use right format image for the appropriate channel.</td>
</tr>
<tr>
<td></td>
<td>• Use CSS sprites to minimize image load requests</td>
</tr>
<tr>
<td></td>
<td>• Compress images for optimal image sizing</td>
</tr>
<tr>
<td></td>
<td>• We can resize the image with minimal size in PNG format</td>
</tr>
<tr>
<td></td>
<td>• Use adaptive images to cater to multiple devices</td>
</tr>
<tr>
<td><strong>Resource requests</strong></td>
<td>Minimize resource requests by merging CSS/JS files to reduce multiple file requests</td>
</tr>
<tr>
<td><strong>JS/CSS Optimization</strong></td>
<td>All the JavaScripts and StyleSheets should be merged and minified. This would not only reduce the HTTP requests but also reduce the impact on overall page size. Also we can place the merged and minified file at the bottom of the page to optimize the perceived page load times</td>
</tr>
<tr>
<td><strong>Asynchronous asset loading</strong></td>
<td>Wherever possible, load the assets such as images asynchronously and only on demand. For instance we need not load the image that is outside of the current view until the user does a page scroll.</td>
</tr>
<tr>
<td><strong>Web-friendly data format</strong></td>
<td>Use optimal data container such as JSON instead of using XML. JSON is optimized for web and is the lightweight alternative to XML.</td>
</tr>
</tbody>
</table>
5. PRO-ACTIVE QUALITY IN WEB APPLICATIONS

In this section we will look at pro-active measures we can take for pro-active quality in web applications from performance stand point

5.1 Common Lead Indicators of a Future Quality Problems

Lead Indicators act as early indicators of potential quality issues which provides opportunity for project managers and project architects to pro-actively undertake course corrective actions. The key lead indicators are depicted in following diagram

5.2 Achieve Quality at Source

Given below are few methods to achieve the quality at the source.

Static Code Analyzers

- Automated IDE plugins which inspect the code for standards violation using customizable/configurable inspectors, duplicate code, critical issues such as memory/connection leaks
- Highly effective during unit testing stage
- Should be used as gating criteria prior for code check-in
- Examples include Checkstyle, PMD, Findbugs, SonarSource

Coding Checklists

- List of coding standards and best practices for programming languages
- Acts as an effective tool in self-review and peer-review
- Infosys standard checklists can be customized to suit project/client requirements

5.3 Continuous integration

Continuous Integration involves frequent and iterative process of building code, testing, integrating and deploying releases to catch issues early in the game. Various tools involved in this process are as follows:

1. **Build**
   1. Integrated code compilation – Maven
   2. Continuous build – Hudson, Jenkins
   3. Build reports - Hudson

2. **Code analysis**
   - Duplicate code analysis – PMD
   - Coding guidelines – Findbugs, CheckStyle, Sonar

3. **Validators**
   - Java code unit testing - Junit
   - Web Testing – Selenium
   - Defect report -- Bugzilla
   - Risk tracking
     - Business risk tracking and monitoring
     - Technical risk tracking and monitoring
5.4 Internal and external monitoring for better QoS
Various aspects of internal and external monitoring is depicted in following diagram:

6. PERFORMANCE FOCUSED DEVELOPMENT & TESTING
Performance focused development and testing involves the following:

- **Performance Testing**
  - Test performance of integrated application early in the game
  - Perform continuous performance testing
  - Open Source tools: Jmeter (for load testing), Load UI, SOAP UI (for web service testing)

- **Automated tools**
  - Tools at various phases of the project
  - Yahoo Yslow, Google PageSpeed for measuring page performance in development
  - Profiling tools such as Jprofiler during troubleshooting phase and load testing tools during performance testing

- **Monitoring**
  - Well-defined real-time application monitoring & notification process
  - Tools: Gomez and Web Analytics tools such as Omniture, Google Analytics
Finally we have elaborated various methods and processes involved in pro-active quality in web applications.

REFERENCES


