STRATEGY FOR BUILDING HIGH PERFORMANCE WEB

Shailesh K S*
Research Scholar,SOCIS , IGNOU, Maidan Garhi, Delhi - 110068
shaileshkumar79@yahoo.com

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

Abstract

A responsive web page plays critical role in the overall success of the online channels. It directly impacts user experience and also influences the search engine rankings. Most of the web systems use personalized web to provide relevant and contextual information to web users.

Personalized web provide content, data and functionality based on various personalization parameters such as user interests, user preferences, user profile attributes, location, device and such. Personalized web is a key business enabler to drive the user traffic and keep the web users engaged by providing useful information. One of the main side effects of personalized web is on the performance. Traditional performance optimization techniques cannot be scaled and reused for personalized web due to information variation and security/privacy concerns. In addition to relevant content, web users would also expect good performance in personalization scenarios. Web architecture needs to design for optimal performance in personalization scenarios for long-term success of web systems.

In this paper we have tried to address this crucial issue by discussing various aspects of web optimization.

***Keywords*:** Caching strategy; Performance Engineering; Performance monitoring & maintenance; Web Performance Optimization; Web performance tools

1. Introduction

As web is becoming a greater influence on the community, there are various initiatives to enhance the web experience. Web pages are now made highly responsive, interactive and personalized and it is available on all mobile devices.

With increase in complexity, the page is stuffed with heavier images, has multiple scripts and style sheets that impact the page performance. This also has an impact on the mobile version of the pages.

Addressing the performance issues is a complex issue as it involves multiple layers and systems. We need to look at all the components involved in the delivery pipeline of the web page.

1.1. *Brief introduction about public web*

As the paper majorly discusses the performance improvements for the web, let us look at the prominent features of personalized web. In order to serve the most relevant content and drive enriching and engaging experiences, web systems are designed to “personalize” the web experience. Personalization is an umbrella term that would involves multiple flavors such as location-based personalization, role based personalization, context-based personalization, device-based personalization and such. All personalization techniques use relevant factors (such as location, user role, access device, user transaction history, user navigation history, user content rating and such) to serve the most relevant content in most suitable form for the web user. Here are some of the personalization scenarios in web:

- Users would see different web content based on their user roles or based on their profile attributes
- Content within a page section varies based on explicitly specified user preferences
- Content would be rendered in user’s language
- Product recommendations would vary based on user’s purchase history or user’s ratings
- Page layout would vary based on user’s access device
- Search results are based on user’s location

Though majority of personalization techniques need user login, it is possible to personalize based on other anonymous factors such as demographics, location, access device etc. that do not require user login. Personalization is an inevitable part in modern web systems. Most of modern web architectures adopt personalization to some extent.

In this paper we mainly discuss performance optimization techniques for optimizing content in personalized web scenario. As such other personalization scenarios such as page layout optimization, automatic content localization is not part of this paper.

1.2. *Performance Challenges in personalized web*

We are exploring personalized web from the performance prism. Traditional caching and content pre-fetching techniques fall short of effectively addressing performance challenges of personalized web. Basic reason for this is due to the fact that web-caching techniques treat all content uniformly and assumes that all users would use it. In personalized web scenario the content would vary based on various criteria. Hence it would not be possible to use the cached content across all user sessions.

Second problem related to performance web is about caching personalized content. It would simply be not possible to cache the content used by all users in cache; this model would quickly become unmanageable due to memory restrictions and cache size limitations. On similar lines, user-based content pre-fetching would not be scalable with increase in users.

In summary the variation in content types (web content, images and other personalized content), variation in user types and potential increase in number of users would pose challenges in adopting pre-fetching and caching techniques used for public scenarios.

In this paper we discuss a novel personalized performance optimization system consisting of various algorithms to address this problem.

1.3. *Paper organization*

In this paper we discuss related work in section 2. We then look at main performance optimization techniques and best practices for public web scenarios in section 3. Conclusion and results are summarized in section 5. Scope for future work is discussed in section 6.

2. Related Work

WPO concept was pioneered by Steve Souder's in 2004. There is good literature [5] [6] [10], which describe generic best practices, and rules that need to be followed to speed up the public web pages. There are also online resources [7] [8] [9] which provide thumb rules for optimizing web. The books and the online resources provide rich information about various optimization techniques that can be carried out for page components.

As far pre-fetching techniques is concerned, there are various techniques discussed in the literature. File-based pre-fetching algorithms use references [14], correlation between file accesses for automatic pre-fetching [15]. For pre-fetching in web scenarios, paper [12] discussed predictive pre-fetching for reducing the latency, access history and web log mining is discussed in paper [16] and [18]; paper [19] uses fetch probability using the objects lifetime and popularity. Weighted graph [13] and partial match prediction [20] and semantic techniques [21] are other pre-fetching techniques used.

3. Performance optimization for the web

In this section we briefly discuss the performance optimization techniques for public web scenarios. This lays the groundwork for understanding differences between public web and personalized web. The concepts discussed in this section could also be used for public pages in personalized web systems.

3.1. *Brief concepts and impact of WPO*

WPO involves best practices and techniques to increase the speed of web pages [1]. In the process we also need to look at other involved components as well.

WPO has impact in following aspects:

- Customer churn: Research indicates that customers would abandon the slower web pages [2]
- 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 [3] and most users expect the page to load within 2 seconds [3].
- 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

When we analyzed the main factors which added to the bottleneck of the public web performance, we found that the bulk of the page load time was spent in loading the page assets such as graphics, JavaScripts and CSS files.

Other research on WPO [5] [6] is also consistent with this observation. So when we adopt an 80-20 rule, the immediate focus should be to optimize these assets

3.2. *Common Performance pit falls for personalized web*

In this section let us look at the common anti-patterns that could lead to the performance issues

- Home page, gateway pages and landing pages are made very heavy with lot of images and scripts
- Using uncompressed images and scripts on the pages
- Not doing a performance testing for all real world scenarios.
- Using too many UI components and un-tested 3rd party widgets
- Not doing a mobile testing for all pages
- Not setting up a monitoring framework
- Not following performance design best practices
- Not adopting a sound multi-layer caching strategy

3.3. *Page Design Optimization for the personalized web*

While designing the pages, we need to prioritize the pages based on their business importance and their criticality to the end user. Normally the home pages and landing pages are important from end user standpoint and shopping/checkout pages are important from business standpoint.
We also should look at the process that is important for business and end user. For instance users needs an optimal user registration process and business needs an optimized shopping experience. The process spans multiple pages and page components.
Once we get the list of all pages and processes we should adopt the following design-time best practices:

- Keep the key pages simple in design. This involves using only necessary UI components.
- Minimize the number of HTTP calls for the page. HTTP calls are required to load the resources and hence reducing the resources would automatically reduce the HTTP requests.
- Explore means to load the page content asynchronously. We can leverage AJAX requests to load the page sections which provide non-blocking page loads
- Ensure that there are no white spaces in the page to reduce the page size
- Avoid using inline images, JavaScripts and style sheets. Externalize all the JS, CSS and images
- Use responsive web design (RWD) technique to cater to multiple devices and form factors. RWD consists of fluid grids, media queries that can auto-adjust based on the target device specifications.
- Ensure the page data is loaded only on demand and in lazy mode. For instance the list data or results data can be shown in paginated view and can be loaded only on user navigation.
- Maintain a performance checklist applicable for the technology platform and use it while developing the page.
- Keep the assets in optimized format. As this topic needs elaboration we have another section dedicated for this.
- Ensure that the page does not has any broken URLs
- Ensure that page request does not redirect which results in additional HTTP request
- Ensure that all duplicate links, scripts, images, content are eliminated on the page. This would reduce unnecessary page requests.
- Perform iterative and phase-wise performance code review and leverage static code analyzers to pro-actively identify any performance issues.
- Test the pages for all loads. We will discuss this in coming sections

- Test the pages for all languages and geographies by simulating the requests. This is elaborated in coming section.
- Test the page for all supported browsers and mobile devices.

3.4. *Asset Optimization for the personalized web*

As we have seen in previous sections, asset forms the bulk of page load time and page size. Hence let us look at various aspects of optimizing the assets.
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:

- Image optimization: All images should be compressed and stored in optimal format. The most optimal image format is PNG format. So instead of using a very high-resolution image, we can resize the image with minimal size in PNG format. Additionally we can also use adaptive images to cater to multiple devices. Another popular technique is to use CSS sprites to combine multiple images into a single image to reduce the image requests. We can also use image maps for optimizing images.
- Script optimization: All the JavaScripts can 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
- Stylesheet optimization: We can merge and minify the CSS files and place it at the top of the page.
- Asynchronous loading: 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.
- Use optimal data container such as JSON instead of using XML. JSON is optimized for web and is the lightweight alternative to XML.

3.5. *Content optimization for the personalized web*

A web page consists of multiple sections. Let us look at ways to optimize the content.

- While designing the content strategy think of content in chunks instead of a monolithic content. Modular content chunk will make the content reusable and enhance the caching optimization
- Use adaptive technique techniques (such as progressive enhancement/degradation) while creating the content. This will automatically make the content optimal for all devices
- Cache the content at chunk level. Fine tune the caching period based on the content update frequency.

3.6. *Content Delivery network (CDN) and web Server Optimization for the personalized web*

CDN offers multi-geo nodes, which can optimally serve the page assets. We can use the CDN network to forward cache the images, static pages, videos, JS, CSS, JSON and such binary content. This can be used to achieve optimal performance across various geos.
We also need to adopt a suitable cache invalidation mechanism to ensure that CDN cache is cleared once the asset republished.

At the web server layer, we can configure few parameters to improve the page performance:

- Cache headers can be set for binary/static assets based on update frequency. “Expires” and “Cache-Control” headers can be leveraged for this.
- Cache expiry can set for the static assets based on mime types
- We can configure performance accelerators such as mod_pagespeed for optimal performance

3.7. *Caching*

Caching is a pre-dominant factor that influences the performance. We need to adopt the caching at all layers: browser cache layer, web server cache layer, application server cache layer, services cache layer and database cache layer. The cache TTL (Time to live) should be configured to ensure the optimal cache freshness. Given below are the caching candidates at various layers:

- Browser layer: We can cache the static assets, scripts, images and JSON data
- CDN Layer: We can cache static assets and static pages
- Web server layer: We can cache the images, content fragments and scripts
- Application server layer: We can create custom caching framework to cache the frequently used look up values, query results, search results, service responses and such
- Services layer: We can cache the results of frequently invoked services
- Database layer: We can cache the results of frequently invoked queries

4. Conclusion

In this research paper, we have identified various performance techniques for page design, asset optimization, content optimization and CDN along with caching. The combination of these techniques would provide an average 30% improvement in the overall page performance.

5. Future scope

Personalized cache could be improved to scale to higher user load and content load with efficient cache invalidation algorithms. Currently with more than 500MB of content id cache, existing cached entries would be replaced based on LRU scheme. High content scenarios would contain large content ids that require more efficient cache management.
Second area of enhancement would be in the client-side caching. The techniques and algorithms discussed in this paper mainly cache on server side. In order to further reduce latency we should forward cache relevant data by leveraging browser cache, proxy cache and CDN networks.

References

[1] Web performance optimization: http://en.wikipedia.org/wiki/Web_performance_optimization
[2] For Impatient Web Users, an Eye Blink Is Just Too Long to Wait: http://www.nytimes.com/2012/03/01/technology/impatient-web-users-flee-slow-loading-sites.html?_r=2
[3] Akamai Report: http://www.akamai.com/html/about/press/releases/2009/press_091409.html
[4] Speed Is A Killer – Why Decreasing Page Load Time Can Drastically Increase Conversions: http://blog.kissmetrics.com/speed-is-a-killer/.
[5] S. Souders – Even Faster Web Sites: Performance Best Practices for Web Developers; O'Reilly Media, 2009
[6] S. Souders – High Performance Web Sites: Essential Knowledge for Front-End Engineers; O'Reilly Media, 2007
[7] Best Practices for Speeding Up Your Web Site: http://developer.yahoo.com/performance/rules.html
[8] Web Performance Best Practices: http://code.google.com/speed/page-speed/docs/rules_intro.html
[9] WPO – Web Performance Optimization: http://www.stevesouders.com/blog/2010/05/07/wpo-web-performance-optimization/
[10] S. Stefanov– Web Performance Daybook; O'Reilly Media, 2012
[11] James Grioen and Randy Appleton. Reducing File System Latency using a Predictive Approach", Proceedings of the 1994 Summer USENIX Technical Conference, Cambridge MA, June, 1994.
[12] Using Predictive Prefetching to Improve World Wide Web Latency: http://ccr.sigcomm.org/archive/1996/jul96/ccr-9607-mogul-padmanabhan.pdf
[13] Gu, Peng; Wang, Jun; Zhu, Yifeng; Jiang, Hong; and Shang, Pengju, "A Novel Weighted-Graph-Based Grouping Algorithm for Metadata Prefetching" (2010). CSE Journal Articles. Paper 44.
[14] D. Kotz and C.S. Ellis, "Practical Prefetching Techniques for Multiprocessor File Systems," J. Distributed and Parallel Databases vol. 1, no. 1, pp. 33-51, Jan. 1993
[15] H. Lei and D. Duchamp, "An Analytical Approach to File Prefetching," Proc. USENIX Ann. Technical Conf., Jan. 1997
[16] Lee, H., An, B., & Kim, E. (n.d.). Adaptive Prefetching Scheme Using Web Log Mining in Cluster-Based Web Systems. 2009 IEEE International Conference on Web Services.
[17] Dahlan, A., & Nishimura, T. (n.d.). Implementation of asynchronous predictive fetch to improve the performance of Ajax-enabled web applications. Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services - IiWAS '08.
[18] Yang, Q., Zhang, H., & Li, T. (n.d.). Mining web logs for prediction models in WWW caching and prefetching. Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '01.
[19] Venkataramani, A., Yalagandula, P., Kokku, R., Sharif, S., & Dahlin, M. (n.d.). The potential costs and benefits of long-term prefetching for content distribution. Computer Communications, 367-375.
[20] Bouras, C., Konidaris, A., & Kostoulas, D. (n.d.). Predictive Prefetching on the Web and Its Potential Impact in the Wide Area. World Wide Web, 143-179.
[21] Xu, C., & Ibrahim, T. (n.d.). Towards semantics-based prefetching to reduce Web access latency. 2003 Symposium on Applications and the Internet, 2003. Proceedings.