Global Journal of Enterprise Information System

DOI: 10.18311/gjeis/2017/15870

Performance Driven Development Framework for Web Applications

K. S. Shailesh* and P. V. Suresh

SOCIS, IGNOU, Maidan Garhi, New Delhi – 110068, Delhi, India; Shaileshkumar79@yahoo.com, pvsuresh@ignou.ac.in

Abstract

The performance of web applications is of paramount importance as it can impact end-user experience and the business revenue. Web Performance Optimization (WPO) deals with front-end performance engineering. Web performance would impact customer loyalty, SEO, web search ranking, SEO, site traffic, repeat visitors and overall online revenue. 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 identified key web performance patterns and proposed novel web performance driven development framework. We have elaborated on various techniques related to different phases of web performance driven development framework.

Keywords: Application, Framework, Images, Traffic, Video, Web

Manuscript Accepted: 13-Feb-2017; **Originality Check:** 24-Feb-2017; **Peer Reviewers Comment:** 08-Mar-2017; **Double Blind Reviewers Comment:** 13-Mar-2017; **Author Revert:** 21-Mar-2017; **Camera-Ready-Copy:** 28-Mar-2017)

1. Introduction to Web Performance Optimization (WPO)

Pages with good performance increase revenue[2,3] and improve the search engine ranking[1]. Page performance also has positive impact on user traffic[56,57] and the download rate impacts the perceived success by end users[71].

Web performance optimization (WPO) involves all methods to improve the performance of web page[87]. The key components used in WPO are web content, images, videos, CSS/JS files, XML/JSON files and such presentation components. WPO also involves various performance rules, techniques and processes to improve end to end performance optimization of the web page.

1.1 Organization of the Paper

The paper is organized as follows. We will start with introduction concepts of WPO and we will then examine various aspects of WPO such as impact and dimensions of WPO in the introduction section. In the next section we elaborate the performance driven development framework. We explain each step in the performance driven development framework including performance based design, performance based development, performance based testing and performance monitoring. In each of these phases we discuss relevant performance rules, design principles and performance best practices. Wherever applicable, we will provide the Java-based web application examples

1.2 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. The performance of landing/gateway pages and key processes is directly co-related to overall user experience

Figure 1. Impact of WPO.

- 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].
  - Conversion rate increases 74% when page loads within 2 seconds[96]

*Author for correspondence

  - Page abandonment rate increases to 40% if page takes > 3 seconds[97]
  - Nearly half the population expects the page to load within 2 seconds[97]
- Revenue: Online revenue is directly correlated to the performance of key pages and transactions for ecommerce sites
- Multi-device optimization: An optimized web page also impacts the performance on various devices.
- Search engine ranking: Google includes site speed in search ranking algorithm
- Omni-Channel advantage: A good performing page can also be easily access by mobile devices.

The impact of WPO on page performance is depicted in the Figure 1. The high level impact on three categories is depicted in Figure 2

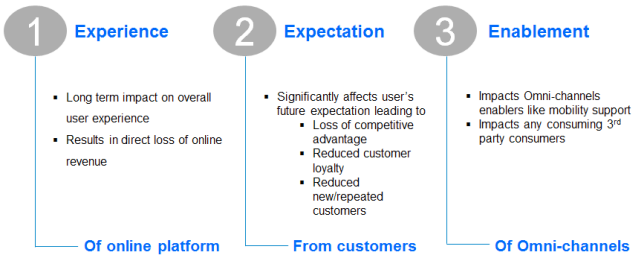

Figure 2. Impact categories of WPO.

1.3 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

Table 2. Categorized Performance Rules

Category	Performance Rule	Impact on web performance
Request Optimization	Reduce the number of HTTP Requests	Reduces the consumed bandwidth and data transferred
	Merge the static assets such as JS and CSS files	Merging would reduce the number of HTTP requests and would improve the page response times by about 38%[66]
	Remove all duplicate file includes	
	Remove all invalid URLs which result in HTTP 404	Avoids unnecessary and invalid HTTP requests
	Load the JavaScript sascynhronously	This would reduce the blocked loading of JS files
	Minimize usage of iframes	iFrames block the loading of parent window till iframe source is loaded and hence affects load times
	Minimize redirects	Minimize additional requests
	Cache DNS records	We can reduce the DNS lookup time through DNS cache maintained at browser level
	Remove any unused CSS, JS file includes	Minimizes HTTP requests
Web object size optimization	Minify JS and CSS files	Minification would reduce the size of JS and CSS file
	Compress images	Compresed images would reduce overall page size
	Leverage gzip for HTTP compression	Compression would reduce the overall page size by about 70%[66]
	Remove white space in the HTML document	Removal of white space would reduce the overall page size

HTTP Header Optimization	Leverage cache headers for static assets (images, JS, CSS, JSON and other binary files) using Cache-Control header with max-age directive	Allows browsers to optimally cache the assets
	Use expires header for the assets	Avoids additional resource request
Asset placement	Place CSS files at the top	CSS elements in the head tag
	Place JS files at the bottom	JS files at the bottom would improve the perceived page load time. I would avoid the blocked loading of other assets
	Externalize inline JS or CSS	Enables browser caching and parallel downloads
Image optimization	Asynchronous image load	Load the images on-demand and in asynchronous when they are visible in the user's view port.
	Optimize image size	Use the right size image based on the requesting device
	Convert JPEG image formats to progressive format	This would reduce the overall image size
	Optimize image dimensions	Specify the exact width and height for all images
	Use image maps	Reduces multiple image requests
	Use CSS sprites	All images are combined into a single one and the required image is displayed using style rules. This reduces number of image requests
	Other techniques: inline images	
Network optimization	Usage of CDN	CDN would optimize the resource request by serving the resource from nearest location to the requestor
	Use multiple asset hosting servers (for hosting images, videos and other multi-media content)	Allows browsers to download the content in parallel.
External Dependency optimization	Identify all external scripts and HTTP requests which impact the page performance and which block the page load and optimize them	
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

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. Performance Driven Development Framework

Performance driven development approach can be adapted in following phases for ground-up development project:

- Defining sound performance based design guidelines
- Implementing the design guidelines during development
- Thorough testing strategy to cover all performance scenarios
- Continuous real-time monitoring

The various phases are depicted in figure 3.

2.1 Web performance Design

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 perfor-

mance stand point. Web developers and architects can use this as reference while developing web applications. Some of these optimizations are also available as filters for Apache's mode_pagespeed module[52].

Figure 3. Performance Driven Development framework.

As 80%66 of load time is spent in making HTTP requests for non-HTML content, we could look at ways to optimize these web components in table 2.

The key page performance design principles are listed below:

- Light weight design
  - Include only core functionality on landing/gateway JSPs
  - Highly optimized/compressed marquee image and other media
  - SLA-based 3rd party integrations on frequently used JSPs
- Search centered experience
  - Position highly optimized search as key tool for information discovery
  - Provide intuitive information architecture
- Think Asynchronous alternatives
  - Use AJAX tags in page to optimize perceived page load time
  - Lazy loading data model for page components
- Omni-channels optimized
  - Page components for mobile devices
- Layered architecture
  - Separation of concerns

Given below are performance anti patterns in a typical Java web application:

- JSP Page size contributors
  - Media (Marquee image, flash, video)
  - JavaScript files
  - CSS
  - Uncompressed/un-optimized images
- JSP Includes/calls
  - Numerous JS/CSS includes
  - Duplicate calls
  - Broken links
  - Unnecessary calls
- Other common causes
  - Placement of JS/CSS calls
  - Bloated size of JSP
  - Frequent resource requests with huge payload
  - Inline styles and JS logic

Table 1. Performance Bottleneck and anti-patterns

Bottleneck Area	Performance anti-patterns
Web page Design	• Heavy landing/gateway pages • UI design with many components and functionality • Pages designed with huge images/flash files
Third-party components	Third-party Scripts and widgets would block page load and impact overall page performance.
Network bandwidth	Usage of sub-optimal network bandwidth across internal systems
Server configuration	Not adopting optimal server settings for parameters such as heap memory, thread pool size, connection pool size etc.
Infrastructure capacity	• Usage of sub-optimal memory, CPU, disk capacity for servers • Not conducting load testing, stress testing, endurance testing and related performance tests
Performance testing	• Not conducting all necessary performance tests (such as load test, stress test, endurance test) for web application • Conducting performance testing at the end of the application development • Not conducting Omni-channel testing to test performance on mobile platforms.
Page code	• Not conducting performance code review • Not performing iterative performance testing
Service calls	• Non validated frequent service calls • Heavy usage of synchronous service calls
Integration design	• 3rd Party component integration without proper SLA framework
Process validation	• Lack of performance testing of overall steps and/or for process/transaction
Omni channel strategy	• Absence of mobility enabled sites or lack of multi-device testing.

2.2 Performance based Development

Performance based development is the key step in the performance driven development framework. Firstly, in this phase we will identify the performance bottlenecks and performance anti patterns and then we will apply all the performance optimization rules and best practices. We will also look at content optimization and the impact of security on WPO. In this section we will also look at optimizing performance for existing web applications.

2.2.1 Web Performance bottlenecks and Web Performance anti-patterns

Let us look at common performance bottlenecks and anti-patterns which impact the web performance. Table 1 provides a list of commonly occurring performance bottleneck

2.2.2 Performance Optimization Practices

Given below are the performance optimization best practices and thumb rules which can be used during development stages:

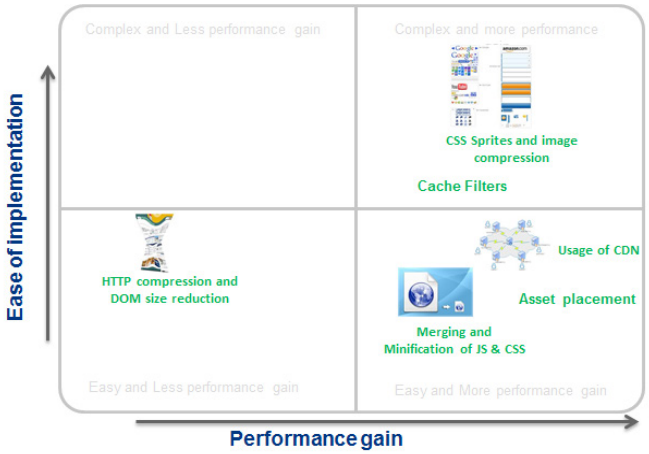

Figure 4. Performance Effort matrix.

Given below are the performance optimization rules for a Java-based web application and these rules can be used for performance code review and as a performance checklist:

- Maximum usage of AJAX
- Cache list items in web page
- Properly scoped managed beans for JSF
- Precompiled JSP
- Optimize intervals for checking JSP and servlet modification
- Usage of JSP cache
- Tune session timeouts (optimal 30 mins) and call session. invalidate() for logout
- Disable JSP/Servlet auto-reload
- Heavy objects in HttpSession needs to be avoided
- Usage of gzip compression wherever supported
- Include the static JSP fragments like header, footer using include directive instead of include action
- Wherever required pre-load servlet and cache the application data using "load-on-startup" feature
- Perform client side validation to avoid un-necessary server round trip
- One time creation of cached data in init() method
- Disable auto-reload feature unless required.

Figure 4 provides the performance effort matrix which provides the effort needed for implementing each of the performance rules and its impact on the overall page performance.

2.2.3 Content Optimization

A web page consists of multiple content sections. The content would come from HTML content fragment or from web content stored in CMS. Let us look at ways to optimize the web content retrieved from CMS.

- 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

Asset Category	Optimization Rules	Impact on page performance
Static Assets (JS, CSS, JSON, XML)	Merge into minimal sets Minify the merged files CSS at Top and JS files at bottom	Merging reduces number of HTTP requests Minification reduces the overall page size speeding the page load Appropriate positioning improves perceived page load On an average 30% page size reduction through merging and minification.
Binary Assets (Image, media, Flash)	Use compressed format and CSS Sprites Use CDN for edge side caching	Optimizes the overall page size Improves performance on mobile devices CDN would provide optimized performance for multi-geographies. On an average 25% page size reduction through usage of CSS sprites On an average 10% improvement in page load time through CDN caching.
Web page related	Remove any duplicate/un-necessary calls Reduce the white spaces Compress/g-zip HTML content	Reduces number of HTTP requests Optimized DOM size for the end web page On an average 20% page size reduction through HTML compression

- 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. Perform on-demand chunk cache invalidation when new content is published.
- Adopt user-friendly information architecture for easier and faster discovery of relevant content
- Tag the content chunks with relevant metadata and tags that helps in accurate information discovery.

2.2.4 Security and 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

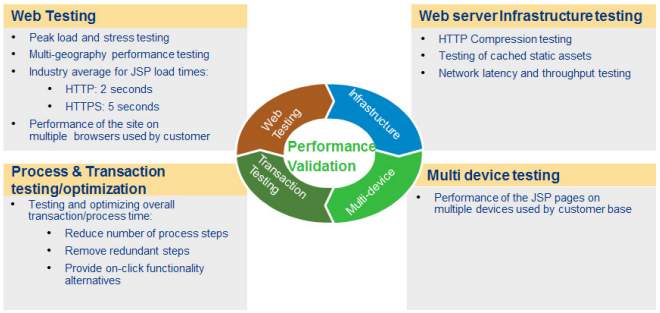

Figure 5. Dimensions of performance testing.

2.2.5 Performance Optimization of Existing Web Applications

In order to optimize the performance of the existing web application we can follow these three-step process:

- Apply 80-20 rule: Identify 20% of pages/processes which is most frequently used
- Root cause analysis: Leverage tools to identify the component-wise and asset-wise size and load times
- Iterate & Monitor: Iteratively cover remaining pages and transactions

2.3 Web Performance Testing

Figure 5 provides the four dimensions of a performance testing.

2.3.1 Web Testing

In this category the web application is subject to peak load and stress load testing. The testing would be conducted for application distributed across various geographies and on all supported browsers and user devices. The industry standard for page load time is 2 second for HTTP and 5 seconds for HTTPS pages. The testing would be conducted to validate this performance SLA at various loads.

2.3.2 Infrastructure Testing

Various infrastructure components such as web server, database server, application server would be tested and monitored at various loads. Server resources such as CPU, memory would be monitored along with network latency and throughput during the testing process.

2.3.3 Process and Transaction Testing

In this phase the key business transactions and processes are tested end to end. Processes like product checkout process, user registration process and such crucial business activities are ideal candidates for testing. We would explore optimization alternatives such as reduction in process steps, process automation, removal of redundant process steps, providing one-click alternatives (such as one-click checkout) in this phase.

2.3.4 Multi Device Testing

In this phase the functionality will be tested for performance and user experience on various devices and browsers.

2.4 Web Performance Monitoring

The last step in performance based design is the continuous performance monitoring. Various steps and activities of the performance monitoring process is depicted in figure 6.

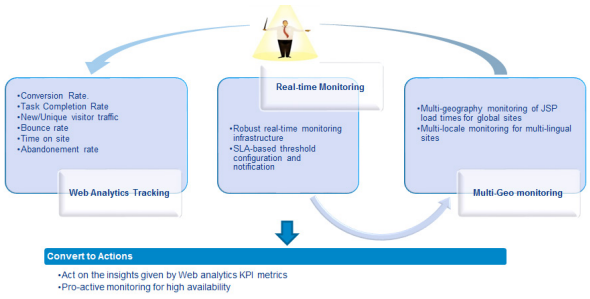

Figure 6. Performance monitoring process.

2.4.1 Performance Tracking

In this phase, we would use web analytics tools to track and monitor page performance metrics such as visitor traffic, page load time (PLT), conversion rate, time on site, bounce rate, abandonment rate and such.

2.4.2 Real Time Monitoring Infrastructure

A monitoring infrastructure would including real time monitoring software and configuration setup to check for performance SLAs on continuous basis.

The performance metrics and SLAs will be tracked in real time and reports/notifications would be sent to site administrators in case of SLA violation

2.4.3 Multi Geo Monitoring

For a globally distributed web site a real-time monitoring would be done from various goes to test the performance for various languages and geo-specific sites.

All the insights gathered through monitoring would be converted into actions and the pro-active monitoring would be an essential component for high availability.

2.4.4 Web Performance Governance

Governance is defining "What" will be governed by "Who" and "How". Figure 7 provides various aspects of performance governance. Performance governance will consists of various governing bodies from Business, IT and Enterprise Architecture, Security and Infrastructure Groups

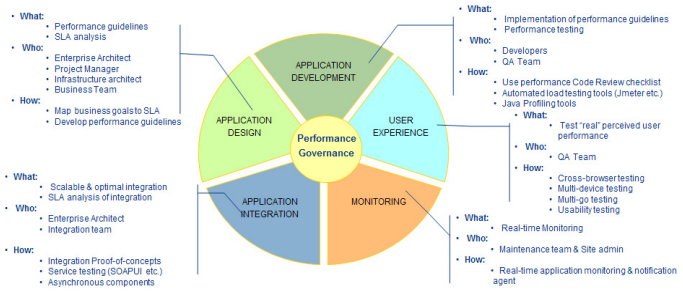

Figure 7. Performance governance.

3. Conclusion

In this paper we examined various aspects of web performance optimization. We proposed a novel performance driven development framework and detailed various aspects of the framework. In each of the phases, we elaborated the web performance optimization techniques such as asset optimization, performance design checklist, performance thumb rules, and performance best practices needed in those phases. Finally we elaborated on performance governance and its various elements.

4. References

1. Google. Using site speed in web search ranking. Available from: http://googlewebmastercentral.blogspot.com/2010/04/using-site-speed-in-web-search-ranking.html. 2010 Apr 9.
2. Galletta DF, Henry R, McCoy S, Polak P. Web Site Delays: How Tolerant are Users? Journal of the Association for Information Systems. 2004; 5(1).
3. Shopzilla: faster page load time = 12% revenue increase. Available from: http://www.strangeloopnetworks.com/ resources/ infographics/web-performanceand-ecommerce/shopzilla-faster-pages-12-revenue-increase/
4. Wang J. A survey of web caching schemes for the Internet. SIGCOMM Comput Commun Rev. 1999 Oct; 29(5):36–46.
5. Chankhunthod A, Danzig PB, Neerdaels C, Schwartz MF, Worrel KJ. A hierarchical Internet object cache, Usenix'96. 1996 Jan.
6. Wang Z. Cachemesh: a distributed cache system for World WideWeb, Web Cache Workshop. 1997.
7. Fan L, Cao P, Almeida J, Broder AZ. Summary cache: a scalable wide-area Web cache sharing protocol, Proceedings of Sigcomm'98.
8. Michel S, Nguyen K, Rosenstein A, Zhang L, Floyd S, Jacobson V. Adaptive Web caching: towards a new caching architecture, Computer Network and ISDN Systems. 1998 Nov.
9. Yang J, Wang W, Muntz R, Wang J. Access driven Web caching, UCLA Technical Report #990007
10. Cunha CR, Jaccoud CFB. Determining WWW user's next access and its application to pre-fetching. Proceedings of ISCC'97: The second IEEE Symposium on Computers and Communications. 1997 Jul.
11. Cohen E, Krishnamurthy B, Rexford J. Improving end-to-end performance of the Web using server volumes and proxy _fillters. Proceedings of Sigcomm'98
12. Markatos EP, Chronaki CE. A TOP-10 approach to prefetching on Web. Proceedings of INET'98.
13. Palpanas T, Mendelzon A. Web prefetching using partial match prediction. Proceedings of WCW'99.
14. Cohen E, Krishnamurthy B, Rexford J. Efficient algorithms for predicting requests to Webservers. Proceedings of Infocom'99
15. Levy-Abegnoli E, Iyengar A, Song J, Dias D. Design and performance of Web server accelerator. Proceedings of Infocom'99.
16. Challenger J, Iyengar A, Witting K, Ferstat C, Reed P. A Publishing System for Efficiently Creating Dynamic Web Content. Proceedings of IEEE INFOCOM 2000. 2000 Mar.
17. Verma DC. Content Distribution Networks: An Engineering Approach. John Wiley & Sons. 2002.
18. Iyengar A, Nahum EM, Shaikh A, Tewari R. Enhancing Web Performance. In Proceedings of the IFIP 17th World Computer Congress - TC6 Stream on Communication Systems: The State of the Art. In: Lyman C, Kluwer BV Editors. Deventer, The Netherlands, The Netherlands. 2002; 95–126.
19. Killelea P. Web Performance Tuning: speeding up the web. O'Reilly Media, Inc. 2002.

20. Available from: http://httpd.apache.org/docs/current/misc/perf-tuning.html
21. Cardellini V, Colajanni M, Yu PS. Dynamic load balancing on scalable Web-server systems. Yorktown Heights, NY: IBM T.J. Watson Research Center. 1998.
22. Acharjee U. Personalized and Artificial Intelligence Web Caching and Prefetching. Master thesis, University of Ottawa,Canada. 2006.
23. Huang YF, Hsu JM. Mining web logs to improve hit ratios of prefetching and caching. Knowledge-Based Systems. 2008; 21(1):62–9.
24. Pallis G, Vakali A, Pokorny J. A clustering-based prefetching scheme on a Web cache environment. Computers and Electrical Engineering. 2008; 34(4):309–23.
25. Wong AKY. Web Cache Replacement Policies: A Pragmatic Approach. IEEE Network Magazine. 2006; 20(1):28–34.
26. Chen HT. Pre-fetching and Re-fetching in Web caching systems: Algorithms and Simulation, Master Thesis, Trent University, Peterborough, Ontario, Canada. 2008.
27. Chen T. Obtaining the optimal cache document replacement policy for the caching system of an EC Website. European Journal of Operational Research. Amsterdam. 2007; 181(2):828.
28. Kumar C, Norris JB. A new approach for a proxy-level Web caching mechanism. Decision Support Systems, Elsevier. 2008; 46(1):52–60.
29. Kumar C. Performance evaluation for implementations of a network of proxy caches. Decision Support Systems. 2009; 46(2):492–500.
30. Domenech J, Gil JA, Sahuquillo J, Pont A. Using current web page structure to improve prefetching performance. Computer Network Journal. 2010; 54(9):1404–17.
31. Padmanabhan VN, Mogul JC. Using predictive prefetching to improve World Wide Web latency. Computer Communication Review. 1996; 26(3):22–36.
32. Cobb J, ElAarag H. Web proxy cache replacement scheme based on back-propagation neural network. Journal of System and Software. 2008; 81(9):1539–58.
33. Ali W, Shamsuddin SM. Intelligent Client-side Web Caching Scheme Based on Least recently Used Algorithm and Neuro-Fuzzy System. The sixth International Symposium on Neural Networks(ISNN 2009), Lecture Notes in Computer Science (LNCS), Springer-Verlag Berlin Heidelberg. 2009; 5552:70–9.
34. ElAarag H, Romano S. Improvement of the neural network proxy cache replacement strategy. Proceedings of the 2009 Spring Simulation Multi Conference, (SSM'09), San Diego, California. 2009. p. 90.
35. Sulaiman S, Shamsuddin SM, Forkan F, Abraham A. Intelligent web caching using neuro computing and particle swarm optimization algorithm. Proceedings of the 2008 Second Asia International Conference on Modelling and Simulation (AMS 08), IEEE Computer Society. 2008. p. 642–7.
36. Tian W, Choi B, Phoha VV. An Adaptive Web Cache Access Predictor Using Neural Network. Proceedings of the 15th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Developments in Applied Artificial Intelligence, Lecture Notes In Computer Science(LNCS), Springer-Verlag London, UK. 2002; 2358:450–9.
37. Markatos EP, Chronaki CE. A Top-10 approach to prefetching on the Web. Proceedings of INET'98 Geneva, Switzerland. 1998; 276–90.
38. Jiang Y, Wu MY, Shu W. Web prefetching: Costs, benefits and performance. Proceedings of the 11th International World Wide Web Conference, New York, ACM. 2002.
39. Tang N, Vemuri R. An artificial immune system approach to document clustering. Proceedings of the Twentieth ACM Symposium on Applied Computing. SantaFe, New Mexico, USA. 2005; 918–22.
40. Ali W, Shamsuddin S, Ismail A. A survey of web caching and prefetching. Int J Adv Soft Comput Appl. 2011; 3(1):18–44.
41. Liu M, Wang F, Zeng D, Yang L. An Overview of world wide Web Caching. International Conference on Systems Man and Cybernetics, IEEE. 2001. p. 3045–50.
42. Barish G, Obraczka K. World Wide Web Caching: Trends and Techniques. 2000.
43. Pons Alexander P. Improving the performance of client web object retrieval. Journal of Systems and Software. 2005; 74(3).
44. Shi L, Han Y-J, Ding XG, Wei L. An SPN based Integrated Model for Web Prefetching and Caching. Springer Journal of Computer Science and Technology. 2006; 21(4).
45. Hussai S, McLeod RD. Intelligent Prefetching at a Proxy Server. Proceedings IEEE Conference on Electrical and Computer Engineering, Cananda.
46. Hung Y, Chen ALP. Prediction of Web Page Accesses by Proxy Server Log. ACM Journal of World Wide Web. 2002; 5(1).
47. Xu C-Z, Ibrahim TI. Towards Semantics-Based Prefetching to Reduce Web Access Latency. Proceedings IEEE Computer Society Symposium, SAINT'03, U.S.A.
48. Xu C-Z, Ibrahim TI. Semantics-Based Personalized Prefetching to Improve Web performance. Proceedings IEEE Conference on Distributed Computing System, U.S.A.
49. Kirchner H, Krummenacher R, Edwards D, Rissel T. A Location-aware Prefetching Mechanism. Project work at Distributed Information Systems Laboratory LSIR. 2004.
50. Agababov V, Buettner M, Chudnovsky V, Cogan M, Greenstein B, McDaniel S, Piatek M, Scott C, Welsh M, Yin B. Flywheel: Google's Data Compression Proxy for the Mobile Web. In Proceedings of NSDI. 2015.
51. The Chromium Projects. SPDY. Available from: https://www.chromium.org/spdy, 2015.
52. mod pagespeed. Available from: http://www.modpagespeed.com/
53. Zhou W, Li Q, Caesar M, Godfrey B. ASAP: A Low Latency Transport Layer. In Proc of the International Conference on Emerging Networking Experiments and Technologies (CoNEXT). 2011.
54. TCP pre-connect. Available from: http://www.igvita.com/2012/06/04/chrome-networking-dns-prefetch-and-tcppre-connect/
55. Radhakrishnan S, Cheng Y, Chu J, Jain A, Raghavan B. TCP Fast Open. In Proc of the International Conference on Emerging Networking EXperiments and Technologies (CoNEXT). 2011.
56. Lohr S. For Impatient Web Users, an Eye Blink Is Just Too Long to Wait. Available from: http://www.nytimes.com/2012/03/01/technology/ impatient-web-users-flee-slow-loading-sites.html, 2012 Mar.

57. Souders S. Velocity and the bottom line. Available from: http://radar.oreilly.com/2009/07/velocity-making-your-site-fast.html, 2009 Jul.
58. Netravali R, Mickens J, Balakrishnan H. Polaris: faster page loads using fine-grained dependency tracking. In 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16). 2016.
59. Datta A, Dutta K, Fishman D, Ramamritham K, Thomas H, VanderMeer D. A Comparative Study of Alternative Middle Tier Caching Solutions to Support Dynamic Web Content Acceleration. In Proceedings of 27th International Conference on Very Large Data Bases (VLDB). 2001 Sep.
60. Datta A, Dutta K, Ramamritham K, Thomas HM, Vander Meer DE. Dynamic Content Acceleration: A Caching Solution to Enable Scalable Dynamic Web Page Generation. In Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD). 2001 May.
61. Iyengar A, Challenger J. Improving Web Server Performance by Caching Dynamic Data. In Proceedings of Usenix Symposium on Internet Technologies and Systems. 1997 Dec.
62. Zukerman I, Albercht D, Nicholson A. Predicting Users' Requests on WWW. In Proceedings of 7th International Conference on User Modeling. 1999 Jun.
63. Zhu H, Yang T. Cachuma: Class-based Cache Management for Dynamic Web Content. Technical Report TRCS00-13, Dept of Computer Science, The University of California at Santa Barbara. 2000 Jun.
64. Su Z, Yang Q, Lu Y, Zhang H. What Next: A Prediction System for Web Requests using N-gram Sequence Models. In Proceedings of 1st International Conference on Web Information System and Engineering. 2000 Jun.
65. Wang Z, Crowcroft J. Prefetching in World Wide Web. In Proceedings of IEEE Global Telecommunications Internet Mini-Conference. 1996 Nov.
66. Souders S. High performance web sites: Essential knowledge for frontend engineers. Farnham: O'Reilly. 2007.
67. Souders S. Even faster web sites. Sebastopol: O'Reilly. 2009.
68. Sundaresan S, Magharei N, Feamster N, Teixeira R. Characterizing and Mitigating Web Performance Bottlenecks in Broadband Access Networks. 2013.
69. Cohen E, Kaplan H. Proactive caching of DNS records: Addressing a performance bottleneck. In Symposium on Applications and the Internet (SAINT). 2001; 85–94.
70. Feldmann A, Caceres R, Douglis F, Glass G, Rabinovich M. Performance of web proxy caching in heterogeneous bandwidth environments. In Proc IEEE INFOCOM, New York, NY. 1999 Mar.
71. Palmer JW. Web site usability, design, and performance metrics. Information Systems Research. 2002; 13(2):151–67.
72. Yagoub K, Florescu D, Issarny V, Valduriez P. Caching strategies for data intensive Web sites. Proceedings of the 26th International Conference on Very Large Data Bases. 2000 May.
73. Podlipnig S, Boszormenyi L. A survey of web cache replacement strategies. ACM Computing Surveys (CSUR). 2003; 35(4):374–98.
74. Iyengar A, Rosu D. Architecting Web sites for high performance. Sci Program. 2002 Jan; 10(1):75–89.
75. Adali S, Candan KS, Papakonstantinou Y, Subrahmanian VS. Query caching and optimization in distributed mediator systems. In ACM SIGMOD Record. ACM. Chicago. 1996 Jun; 25(2):137–46.
76. Challenger J, Iyengar A, Witting K, Ferstat C, Reed P. A Publishing System for Efficiently Creating Dynamic web Content Proceedings of IEEE INFOCOM. 2000.
77. Challenger J, Iyengar A, Dantzig P. A Scalable System for Consistently Caching Dynamic Web Data. Proceedings of IEEE INFOCOM'99
78. Apostolopoulos G, Peris V, Saha D. Transport Layer Security: How much does it really cost? Proceedings of IEEE INFOCOM. 1999.
79. Vakali A, Pallis G. Content delivery networks: Status and trends. Internet Computing, IEEE. 2003; 7(6):68–74.
80. Ravi J, Yu Z, Shi W. A survey on dynamic Web content generation and delivery techniques. Journal of Network and Computer Applications. 2009; 32(5):943–60.
81. Sivasubramanian S, Pierre G, van Steen M, Alonso G. Analysis of caching and replication strategies for Web applications. IEEE Internet Comput. 2007; 11(1):60–6.
82. Yagoub K, Florescu D, Issarny V, Valduriez P. Caching strategies for data intensive Web sites. Proceedings of the 26th International Conference on Very Large Data Bases. 2000 May.
83. Fortino G, Mastroianni C. Special section: enhancing content networks with P2P, grid and agent technologies. Future Gener Comp Syst. 2008; 24(3):177–9
84. Candan KS, Li WS, Luo Q, Hsiung WP, Agrawal D. Enabling dynamic content caching for database-driven web sites. In ACM SIGMOD Record, ACM. 2001 May; 30(2):532–43.
85. Best Practices for Speeding up Your Web Site: Available from: http://developer.yahoo.com/performance/rules.html
86. Ravi J, Yu Z, Shi W. A survey on dynamic Web content generation and delivery techniques. J Netw Comput Appl. 3 2009 Sep; 2(5):943–60.
87. Web performance optimization: Available from: http://en.wikipedia.org/wiki/Web_performance_optimization
88. For Impatient Web Users, an Eye Blink Is Just Too Long to Wait: Available from: http://www.nytimes.com/2012/03/01/technology/impatient-web-users-flee-slow-loading-sites.html?_r=2
89. Akamai Report: Available from: http://www.akamai.com/html/about/press/releases/2009/press_091409.html
90. Rempel G. Defining Standards for Web Page Performance in Business Applications. Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering - ICPE '15. 2015.
91. Galletta DF, Henry R, McCoy S, Polak P. Web Site Delays: How Tolerant are Users? Journal of the Association for Information Systems. 2004; 5(1):1.
92. Hoxmeier JA, DiCesare C. System response time and user satisfaction: an experimental study of browser based applications. Proceedings of the Americas Conference on Information Systems, Association for Information Systems, Long Beach, CA, USA. 2000. p. 140–5.
93. Google Page Speed Insights and Rules: Available from: https://developers.google.com/speed/docs/insights/rules

94. Kambhampaty S, Modali VS, Bertoli M, Casale G, Serazzi G. Performance Modeling for Web based J2EE and .NET Applications. In Proc of world Academy of Science, Engineering and Technology. 2005 Oct; 8.
95. Barker T. High performance responsive design: Building faster sites across devices (1st ed). O'Reilly Media. 2014.
96. Available from: http://www.compuware.com/application-performance-management/performance-index-faq.html
97. Available from: http://www.akamai.com/html/about/press/releases/2009/press_091409.html

Citation:
K. S. Shailesh and P. V. Suresh
"Performance Driven Development Framework for Web Applications",
Global Journal of Enterprise Information System. Volume-9, Issue-1, January-March, 2017. (http://informaticsjournals.com/index.php/gjeis)
Conflict of Interest:
Author of a Paper had no conflict neither financially nor academically.