International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.803

Performance Analysis of Single Page Applications

Mounika Kothapalli

Software Engineer II at Microsoft
Email: *moni.kothapalli[at]gmail.com*

Abstract: ***Single Page Applications (SPAs) have been widely adopted as they offer seamless, responsive, and interactive user experience. This paper explores different performance optimization techniques along with details of advanced performance metrics that are needed for the evolving SPAs. It discusses the optimization strategies like code splitting, lazy loading, caching, and efficient state management along with server - side rendering. It also discusses real time examples of enterprise applications that benefitted from adopting performance first strategies. It concludes by stressing the importance of user - centric performance which strikes balance between performance optimization and application functionalities.***

Keywords: Single Page Applications (SPAs), performance analysis, performance optimization, performance metrics, case studies

1. Introduction

In Single Page Applications (SPAs) when the browser requests for client application, the web server responds with a single HTML page typically an index. html. The content updates dynamically as the user interacts with the application without the need to reload the entire page [1]. This allows a seamless and fluid user experience, similar to native applications which is achieved through leveraging JavaScript (JS) frameworks and libraries.

The concept emerged in the early 2000s, with the introduction of JS frameworks such as Backbone. js and AngularJS. As there is a demand for responsive, intuitive, and user - friendly web applications the SPAs became popular. Moreover, with the flexibility offered by other JS libraries such as React and Vue. js also helped to adopt these applications in real time.

User satisfaction and engagement is critical for any application to succeed. This emphasized the importance of performance in SPAs. In competitive market conditions where faster load times, smooth navigations are critical a superior user experience helps achieving the business objectives [2].

Research Background
This paper aims to provide a comprehensive analysis of SPAs performance in various aspects. The objectives are to:
- Explore the performance metrics which play key role.
- Look into different performance optimization techniques and best practices for SPAs.
- Evaluate performance monitoring and discuss available tools for analysis.
- Different case studies and real - world examples of SPA performance optimization.

With these objectives the paper tries to provide different insights and recommendations for developers and enterprises that plan to build and optimize SPAs.

2. Literature Review

The traditional web applications consist of multi - page model in which a new user interaction results in full page reload from the server. Hence, there is slowness in the response times and the experience is also not fluid. However, in SPAs each user action only dynamically loads specific content requested which results in faster load times and consist of different components as shown in Fig.1.

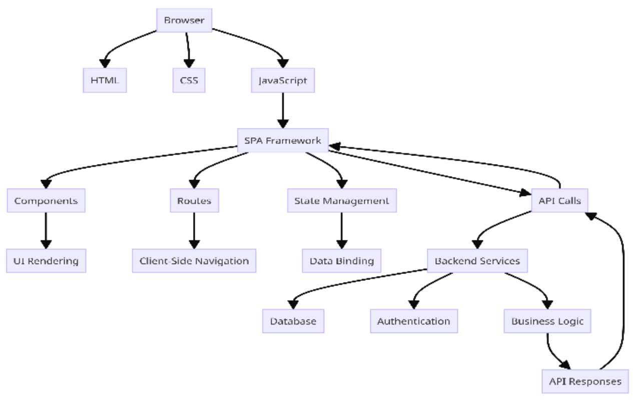


Figure 1: Architecture of a Single Page Application (SPA).

The key characteristics that keep SPAs apart are the more responsive and interactive user experience, as the application can update content without the need for entire page reload [3]. They also offer offline capabilities which allow users to continue interactive with the application even with slow or no internet connectivity. They rely heavily on JS frameworks (e. g., Angular, React and Vue) to render content on the client side which reduces the round trips to server. SPAs offer advantages like quick rendering of updates without the need for server round trips. This is similar to native mobile application experience.

However, there are some challenges in terms of initial load time which could be little slower compared to traditional web applications as the entire app and its dependencies needs to load [4]. The search engine optimization could be a challenge as SPAs rely on JavaScript to render content. The search engines may find it hard to index content that are dynamically loaded which impacts the search findings of the pages.

1) Performance Metrics for SPAs

Understanding the key metrics as shown in Fig.2 of performance would help in measuring and optimizing the Single Page Applications. These would give deep understanding of the application’s behavior and give insights into areas that need improvement.

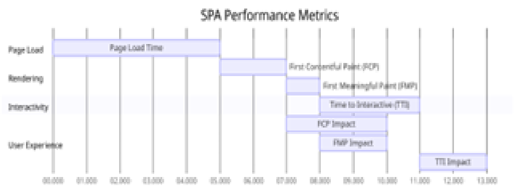


Figure 2: Performance metrics in a SPA

a) *Page load time*

Page load time gives the duration from user request to complete rendering of the page. It is critical as it decides user engagement with the application. This metric shows how quickly user can start interacting with the application [5].

b) *Time to Interactive (TTI)*

This measures how long it takes for the page to be fully interactive and be responsive to user input. In other words, the main thread is free of long tasks and the user can interact with the page without delays [6].

c) *First Contentful Paint (FCP)*

First Contentful Paint measures the time taken by first piece of the content to render from the DOM [6]. This helps in understanding how quickly users can see visual feedback from their actions on the application.

d) *First Meaningful Paint (FMP)*

This measures the duration for the primary content of the page to be visible and useful to the user. It also means the time taken for the most meaningful elements to be displayed.

e) *Frames Pers Second (FPS)*

This metric gives the smoothness and fluidity of a SPA’s animations and interactions. The higher the FPS the smoother the animations and a more responsive user interface. A consistent FPS indicates seamless and smooth user experience [5].

f) *Memory Usage*

This is another critical metric which helps identify how the memory resources are being used by JavaScript heavy applications. This also helps identify memory leaks, excessive garbage collection and other performance bottlenecks [6].

g) *Network Requests and Payload*

Network requests and payload size are important to consider when optimizing SPA performance. Reducing the number and size of network requests can significantly reduce the application’s load time and improve performance.

2) Performance optimization techniques for SPAs

There are various techniques as shown in Fig.3 to optimize the performance of Single Page Applications. These techniques improve the initial load time, responsiveness and efficiently utilize system resources.

a) *Code Splitting and Lazy Loading*

Code splitting involves dividing the entire code into smaller chunks that can be loaded on demand instead of all at once. And Lazy Loading further enhances this by delaying the loading of non - essential resources until they are requested for. These techniques primarily enhance the initial load time making the application responsive [7].

b) *Caching and Offline Support*

Caching allows storage of assets and data locally to lower the number of network requests. Service workers allow caching resources and handle offline requests. These speeds up the load times and offers offline functionality. With caching and fallback mechanisms SPAs improve reliability and performance [8].

Figure 3: Performance optimization techniques.

c) *State Management*

Libraries like Redux and MobX help manage the state of application which would allow predictable state and avoid event spaghetti [9]. This also allows reducing redundant re - renders and thereby improving performance and reducing memory overhead.

d) *Server - Side Rendering (SSR) and Pre - Rendering*

These techniques offer improved performance and SEO. SSR is a technique where rendering of the initial HTML is on the server and is sent to the client which reduces the first meaningful paint. Pre - rendering generates static HTML for specific routes at build time and improves the rendering time.

e) *Minimize DOM manipulations and Repaints*

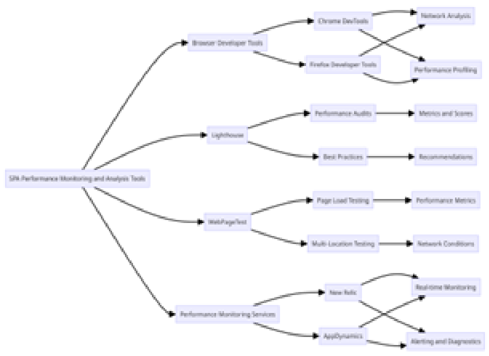
Excessive DOM manipulations lead to slow performance and increases rendering times. Using virtual DOM libraries like React and avoiding direct DOM changes would improve rendering speed [9].

f) *Virtualized. Lists and Infinite Scrolling*

For scenarios with large lists of data implementing virtualized lists and infinite scrolling greatly enhance performance. User can view only the content in the viewport so virtualized lists aim to render items that can be seen in the viewport which improves the rendering time. Infinite scrolling dynamically loads additional content as the user scrolls providing a seamless experience [10].

3) Performance monitoring and analysis tools

Different performance monitoring tools offer valuable insights into the application's behavior, performance bottlenecks and help make informed decisions to optimize them. Tools such as Chrome DevTools and Firefox Developer Tools are essential for monitoring and analyzing SPA performance [11]. They offer network monitoring, performance profiling and memory analysis. In addition, developers can measure resource load times, JavaScript execution times and DOM rendering.



Lighthouse is an open - source tool developed by Google that allows to audit the application performance. It also recommends steps to improve performance and offers metrics (FCP, TTI etc), accessibility, SEO and other best practices [12].

WebPageTest is another powerful tool for assessing the SPA performance. It lets the developers test their applications based off different locations in the world using different network conditions. It provides detailed performance metrics. Such as page load times, time to interactive. It also provides visual comparison tools and the ability to generate performance reports over time [13].

In production environment developers can leverage New Relic and AppDynamics to gain real - time insights into SPA performance. These are especially useful to do real user monitoring and infrastructure monitoring.

3. Case Studies and Real - World Examples

a) Real - world examples

Looking into different example apps which achieved significant performance improvement gives valuable insights for developers to adopt these techniques. First example is Twitter Lite which implemented code splitting, lazy loading and is a progressive web application (PWA) version of twitter. This reduced the initial page load time by 50% and an increase of 65% increase in pages per session [14].

Another example is Flipkart, which is India's largest e - commerce platform that adopted a performance focused approach in their SPA. It leveraged techniques like server - side rendering, code splitting and efficient state management with redux. Through these techniques it improves Time to interactive (TTI) by 30% and saw a 40% reduction in bounce rates [15].

Other examples include Gmail and Trello which implemented service workers, lazy loading to significantly improve user experience.

b) Lessons learned and best practices

Large enterprise applications such as Google, Facebook, Flipkart, and Airbnb have described the best practices they implemented for their SPAs. Lessons include the implementing efficient caching strategies, prioritizing user centric performance metrics, continuous monitoring, and iterative improvements. In addition, server - side rendering, state management, lazy loading and code splitting are vital.

4. Future Trends and Challenges

a) Emerging Technologies

Technologies are emerging at rapid pace and with the advancements in JavaScript frameworks it has significant impact on SPA performance. HTTP/3 has better performance over HTTP/2 in terms of latency and data transfer. The newer JS frameworks also enhancing performance making it easier for developers to focus simply on development.

b) Challenges in Performance Measurement

Measure SPA performance is a complex task as it holds lot of dynamic content. Traditional metrics alone are not enough instead advanced metrics like Time to interactive (TTI) and Total Blocking Time (TBT) need to be used. In addition, considering different devices and network speeds adds complexity. The developers need to use a combination of tools and techniques to identify and address the performance bottlenecks [16].

c) Balancing performance with user experience and functionality

The main challenge is offering high peformance without compromising the funcrtionality. Whiele performance is critical it should not be at the cost of user experience and functionality. Developers need to prioritize the user - centric performance improvements that enhance usability without compromising the features and capabilities of the application [17].

5. Conclusion

a) Recap of key discussions

This paper discussed the performance metrics and highlighted the importance of optimization in SPAs. Key findings such as code splitting, lazy loading, state management and caching are discussed. The usage of tools such as Chrome DevTools, lighthouse was also emphasized as critical for analysis and optimization.

Given the dynamic nature of web applications and the evolution of JavaScript frameworks there is a need for continuous improvements and iterative optimization through audits by leveraging the tools and technologies

b) Recommendations to developers

For developers starting off with SPAs it is recommended to adopt performance first mindset for the entire development lifecycle. Key recommendations include utilize techniques such as code splitting, lazy loading, and server - side rendering to enhance performance. Use tools like Chrome

DevTools, Lighthous and other performance monitoring services to optimize performance. Most importantly, prioritize user - centric performance optimization thereby not compromising the user experience. Finally staying updated with emerging technologies would offer significant benefits.

References

[1] M. S. Mikowski and J. C. Powell, Single page web applications: JavaScript end - to - end. Manning Publications, 2013.

[2] R. Kohavi and R. Longbotham, "Online controlled experiments and A/B testing, " Encyclopedia of Machine Learning and Data Mining, vol.7, no.8, pp.922 - 929, 2017.

[3] J. Gregory and R. Nayak, "Single Page Applications: The Future of Web Development, " in Proc. International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, pp.2137 - 2141, doi: 10.1109/ICACCI.2018.8554467.

[4] U. Wickramarachchi and A. Priyanga, "A comparative analysis of single page application frameworks, " in Proc. IEEE International Conference on Industrial and Information Systems (ICIIS), 2018, pp.437 - 442, doi: 10.1109/ICIINFS.2018.8721413.

[5] A. Arvind and P. S. Banu, "A Study on Single Page Application Frameworks, " in Proc.3rd International Conference on Computing Methodologies and Communication (ICCMC), 2019, pp.1116 - 1119, doi: 10.1109/ICCMC.2019.8819692.

[6] A. Ahmad, M. Azam, and M. Naeem, "Performance Evaluation of Progressive Web Apps and Single Page Applications, " in Proc.22nd International Multitopic Conference (INMIC), 2019, pp.1 - 6, doi: 10.1109/INMIC48123.2019.9022759.

[7] T. Chadha and R. Narang, "Performance Optimization of Single Page Application Using Progressive Web App and Webpack, " in Proc.4th International Conference on Internet of Things: Smart Innovation and Usages (IoT - SIU), 2019, pp.1 - 6, doi: 10.1109/IoT - SIU.2019.8777650.

[8] L. Zhu, H. Jiang, and Z. Zhu, "Research on the Application of Service Workers in Web Application, " in Proc. IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2019, pp.1451 - 1455, doi: 10.1109/IMCEC46724.2019.8983852.

[9] A. Fedosejev, React 16 Tooling. Packt Publishing, 2017.

[10] J. M. Chung, S. H. Kim, and J. H. Kim, "Performance optimization of single - page applications through the use of in - memory cache and client - side rendering, " in Proc. International Conference on Information and Communication Technology Convergence (ICTC), 2018, pp.371 - 373, doi: 10.1109/ICTC.2018.8539687.

[11] J. Dugan and M. Panar, "Mastering Browser Developer Tools, " in Proc. Graz University of Technology, 2017, pp.1 - 8. [Online]. Available: https: //www.researchgate. net/profile/Markus - Panar/publication/319101021_Mastering_Browser_Developer_Tools/links/59e5b1e9a6fdcc1b1d96e8c4/Mastering - Browser - Developer - Tools. pdf

[12] Lighthouse, Google Developers. Accessed: Jun.7, 2020. [Online]. Available: https: //developers. google. com/web/tools/lighthouse

[13] WebPageTest Documentation. Accessed: Jun.7, 2020. [Online]. Available: https: //docs. webpagetest. org/

[14] A. Osmani, "A Tinder Progressive Web App Performance Case Study, " Medium, Nov.28, 2017. [Online]. Available: https: //medium. com/[at]addyosmani/a - tinder - progressive - web - app - performance - case - study - 78919d98ece0

[15] A. Raghavan, "How We Built Flipkart Lite: A Progressive Web App, " Flipkart Engineering Blog, May 21, 2018. [Online]. Available: https: //tech. flipkart. com/how - we - built - flipkart - lite - a - progressive - web - app - 49170ca8e071

[16] S. Chaudhary, H. Somani, and S. Mukherjea, "Performance Testing of Single Page Applications: Challenges and Solutions, " in Proc. IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2019, pp.110 - 115, doi: 10.1109/ICSTW.2019.00037.

[17] J. Rzepka and K. Benedyczak, "Performance and User Experience Analysis of Single Page Application Frameworks, " in Proc. International Conference on Computer Networks, 2019, pp.178 - 191, doi: 10.1007/978 - 3 - 030 - 21952 - 9_14.