IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 9, September 2023

DOI: 10.17148/IJARCCE.2023.12925

Boosting Mobile Web Performance: Advanced Techniques for Modern Websites

Sivaramarajalu Ramadurai Venkataraajalu

New York, United States

Abstract: The ubiquity of mobile devices has fundamentally altered the landscape of internet usage, with mobile web browsing now surpassing desktop usage. This paradigm shift presents unique challenges in web performance optimization, particularly for mobile devices with varying hardware capabilities and network conditions. This paper explores advanced techniques for accelerating mobile web performance, focusing on modern websites that demand high interactivity and rich user experiences. Through a comprehensive review of current literature and an analysis of emerging technologies, we propose novel approaches to enhance mobile web performance. Our findings indicate that a combination of server-side optimizations, client-side rendering techniques, and intelligent resource management can significantly improve mobile web performance metrics, leading to enhanced user satisfaction and engagement.

Keywords: Mobile Web, Web Performance, Mobile User Experience, Device Optimization, Modern Websites

I. INTRODUCTION

The mobile web has become an integral part of our daily lives, with smartphones and tablets serving as primary devices for internet access for millions of users worldwide. According to recent statistics, mobile devices accounted for 54.8% of global website traffic in the first quarter of 2021 [1]. This shift towards mobile browsing has necessitated a re-evaluation of web performance optimization strategies, as mobile users often face unique challenges such as limited processing power, constrained battery life, and unreliable network connections.

The importance of mobile web performance cannot be overstated. Studies have shown that even small delays in page load times can lead to significant drops in user engagement and conversion rates. For instance, research indicates that as page load time increases from one to ten seconds, the probability of a mobile site visitor bouncing increases by 123% [2]. This underscores the critical need for advanced techniques to accelerate mobile web performance and ensure a seamless user experience.

The impact of poor mobile web performance extends beyond user experience to have tangible business implications. A study by Google found that 53% of mobile site visits are abandoned if pages take longer than three seconds to load [3]. This high abandonment rate directly translates to lost revenue opportunities for businesses across various sectors. Moreover, search engines like Google have incorporated page speed as a ranking factor for mobile searches, making performance optimization crucial for visibility and competitiveness in the digital marketplace [4].

The challenges of mobile web performance are multifaceted, stemming from a combination of hardware limitations, network variability, and the increasing complexity of web applications. Mobile devices, despite their advancing capabilities, still lag desktop computers in terms of processing power and memory. This disparity becomes particularly apparent when dealing with JavaScript-heavy applications or complex visual renderings [5]. Additionally, mobile networks are inherently less stable than their fixed-line counterparts, with performance varying greatly based on factors such as location, network congestion, and technology (e.g., 3G vs. 4G vs. 5G) [6].

Furthermore, the trend towards rich, interactive web applications has led to an increase in the size and complexity of web resources. The average web page size has grown exponentially over the past decade, with some estimates suggesting a 300% increase since 2011 [7]. This growth in page weight, coupled with the proliferation of third-party scripts for analytics, advertising, and other functionalities, presents significant challenges for mobile performance optimization.
In light of these challenges, this paper aims to provide a comprehensive overview of current research and emerging technologies in mobile web performance optimization. We will explore various strategies, from server-side optimizations to client-side rendering techniques, and discuss their efficacy in improving key performance metrics. Additionally, we will propose novel approaches that leverage the latest advancements in web technologies to push the boundaries of mobile web performance.

 This work is licensed under a Creative Commons Attribution 4.0 International License

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 9, September 2023
DOI: 10.17148/IJARCCE.2023.12925

The structure of this paper is as follows: Section 2 provides a literature review, summarizing key findings from recent research in mobile web performance optimization. Section 3, which forms the core of our contribution, presents advanced techniques for accelerating mobile web performance, including detailed explanations of our proposed methodologies. Section 4 outlines our experimental results and provides a discussion of our findings. Finally, Section 5 concludes the paper and suggests directions for future research in this rapidly evolving field.

By addressing the critical issue of mobile web performance, this research contributes to the broader goal of creating a more accessible, efficient, and user-friendly mobile web ecosystem. The techniques and insights presented here have the potential to significantly impact how developers and organizations approach mobile web development and optimization, ultimately leading to improved user experiences and business outcomes in the mobile-first digital landscape.

II. LITERATURE REVIEW

The field of mobile web performance optimization has been a subject of extensive research over the past decade. As mobile devices have evolved and web technologies have advanced, researchers and practitioners have continually sought new ways to improve the speed and responsiveness of mobile websites.

A. Server-SideOptimizations

Server-side optimizations have long been recognized as a crucial component of web performance enhancement. Sundaresan et al. conducted a comprehensive study on the impact of server-side delays on mobile web performance [8]. Their findings highlighted the significant role that server response times play in overall page load times, particularly in mobile networks with high latency.

Building on this work, Wang et al. proposed SPEEDY, a novel server-side optimization technique that leverages machine learning to predict and preemptively load resources likely to be requested by mobile clients [9]. Their approach demonstrated significant improvements in page load times, especially for complex web applications with numerous resources.
Content Delivery Networks (CDNs) have also been extensively studied in the context of mobile web performance. Rawat et al. analyzed the effectiveness of CDNs in reducing latency for mobile users across different geographical locations [10]. Their research underscored the importance of strategic CDN placement and intelligent routing algorithms in minimizing round-trip times for mobile clients.

B. Client-Side Rendering Techniques

As web applications have grown more complex, client-side rendering techniques have gained prominence in the mobile web performance landscape. The advent of Single Page Applications (SPAs) and Progressive Web Apps (PWAs) has led to a shift in how web content is delivered and rendered on mobile devices.

Fink et al. conducted a comparative analysis of different client-side rendering approaches, including server-side rendering with client-side hydration, client-side rendering, and incremental rendering [11]. Their study revealed that the optimal rendering strategy often depends on factors such as network conditions, device capabilities, and application complexity.
The concept of code splitting, and lazy loading has been explored by several researchers to improve initial load times for mobile web applications. Jia et al. proposed an adaptive code splitting technique that dynamically adjusts the granularity of code chunks based on network conditions and user interaction patterns [12]. Their approach showed promising results in reducing Time to Interactive (TTI) metrics for complex mobile web applications.

C. Resource Optimization and Management

Efficient resource management is crucial for mobile web performance, given the limited processing power and battery life of mobile devices. Meyerovich and Bodik introduced the concept of "big memory" for mobile browsers, which involves intelligent caching and prefetching strategies to optimize resource usage [13].

Building on this work, Chen et al. developed CASPER, a system for intelligent cache management in mobile browsers [14]. CASPER uses machine learning techniques to predict which resources are likely to be reused across different web pages, allowing for more efficient use of limited cache space on mobile devices.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 9, September 2023

DOI: 10.17148/IJARCCE.2023.12925

Image optimization has been another area of focus, given the visual nature of modern web applications. Pouyanfar et al. proposed an adaptive image compression technique that adjusts compression levels based on network conditions and device characteristics [15]. Their approach demonstrated significant reductions in data transfer sizes without compromising visual quality on mobile devices.

D. Emerging Technologies and Future Directions

Recent advancements in web technologies have opened up new avenues for mobile web performance optimization. The adoption of HTTP/2 and HTTP/3 protocols has been studied by several researchers, including Saxena et al., who analyzed the impact of these protocols on mobile web performance across different network conditions [16].

Web Assembly (Wasm) has emerged as a promising technology for improving the performance of compute-intensive tasks on mobile devices. Jangda et al. conducted a comprehensive study on the performance implications of Web Assembly for mobile web applications, highlighting its potential to bridge the gap between native and web performance for complex computations [17].

The intersection of mobile web performance and emerging technologies such as 5G networks and edge computing has also garnered attention from researchers. Taleb et al. explored the potential of Mobile Edge Computing (MEC) in reducing latency and improving the performance of mobile web applications, particularly for resource-intensive tasks such as augmented reality and real-time video processing [18].

III. ADVANCED TECHNIQUES FOR MOBILE WEB PERFORMANCE

Building upon the foundation laid by previous research, we propose a set of advanced techniques for accelerating mobile web performance. These techniques leverage the latest advancements in web technologies and take into account the unique challenges posed by mobile environments. In this section, we will explore each technique in depth, discussing their implementation, potential benefits, and challenges.

E. Adaptive Server Push

The adaptive server push mechanism we propose represents a significant advancement over traditional server push techniques. By utilizing machine learning algorithms, this approach predicts resource requirements based on a combination of user behavior patterns, device characteristics, and historical data. The system continuously analyzes user interactions, page structure, and network conditions to make informed decisions about which resources to push proactively.

The adaptive server push operates on a feedback loop, as illustrated in Figure 1.

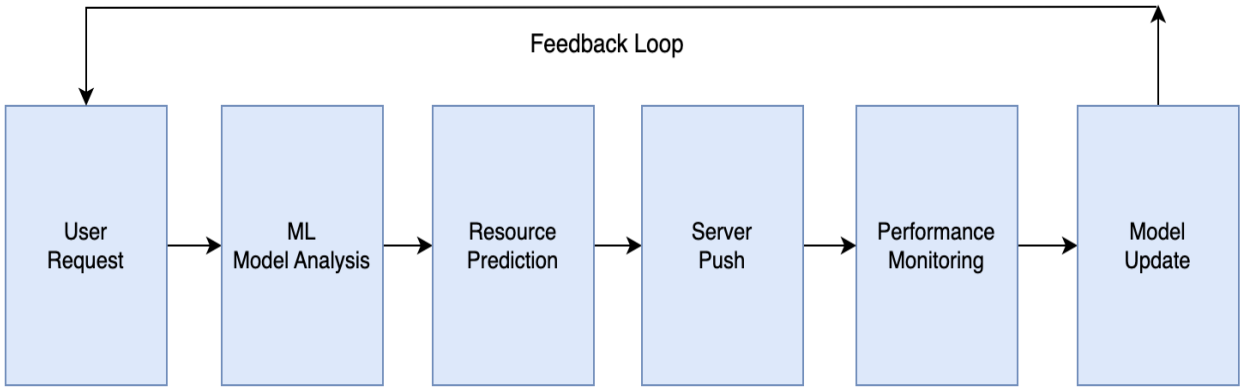

Fig. 1 Adaptive Server Push Feedback Loop

The process begins when a user initiates a request. The machine learning model, trained on aggregated anonymous user data, analyzes the request context, including device type, network conditions, and user history. Based on this analysis, it predicts which resources are likely to be needed next. The server then pushes these resources proactively, before the client explicitly requests them.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 9, September 2023

DOI: 10.17148/IJARCCE.2023.12925

A key innovation in this approach is the real-time performance monitoring and model updating mechanism. As users interact with the site, the system tracks performance metrics such as resource utilization, load times, and user engagement. This data is fed back into the machine learning model, allowing it to continuously refine its predictions and adapt to changing patterns.

The adaptive nature of this technique addresses one of the primary challenges of traditional server push: the risk of pushing unnecessary resources, which can actually degrade performance. By dynamically adjusting its strategy based on real-world performance data, our approach minimizes this risk while maximizing the benefits of proactive resource delivery. Implementation of this technique requires careful consideration of privacy and data usage concerns. All user data must be anonymized and aggregated to protect individual privacy, and the system should provide transparent opt-out mechanisms for users who prefer not to participate in data collection.

F. Granular Code Splitting and Predictive Loading

Our approach to code splitting and predictive loading takes the concept of lazy loading to a new level of granularity. Instead of splitting an application into large chunks, we propose a system that breaks the codebase down into fine-grained modules, some as small as individual functions. This granular approach, combined with a sophisticated predictive loading mechanism, allows for highly optimized and personalized resource delivery.

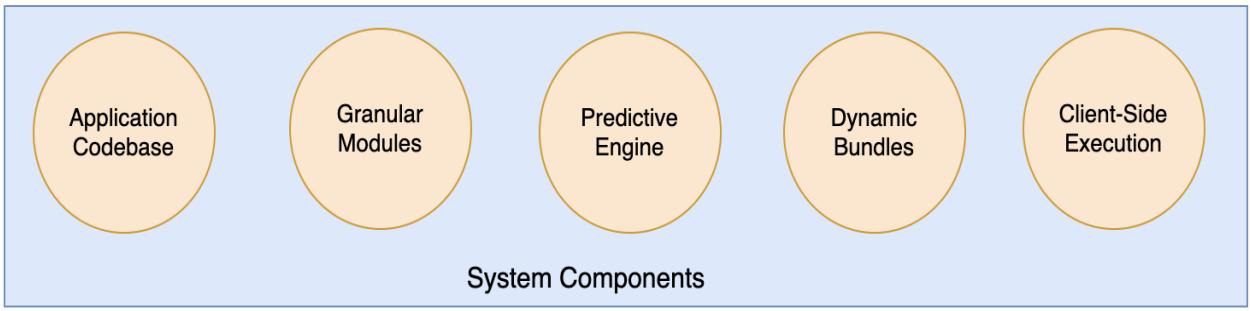

Fig. 2 Components involved in the system

The process begins with a comprehensive analysis of the application's dependency graph. This analysis identifies logical boundaries for code splitting, taking into account factors such as module interdependencies, frequency of use, and potential for reuse across different parts of the application.

Once the codebase is split into granular modules, the system employs a predictive loading engine that operates in real-time as users interact with the application. This engine considers multiple factors to determine which modules to load next:

1. User interaction patterns: The system tracks how users navigate through the application, learning common paths and anticipating likely next steps.
2. Device capabilities: The loading strategy adapts based on the device's processing power, available memory, and network conditions.
3. Application state: The current state of the application influences which modules are likely to be needed next.
4. Global usage patterns: Aggregated data from all users helps inform predictions, especially for new users or uncommon interactions.

Based on these factors, the predictive loading engine dynamically generates optimized bundles of code modules. These bundles are then loaded in the background, prioritizing critical functionality while deferring non-essential code execution. One of the key challenges in implementing this technique is managing the increased complexity of the build and deployment process. To address this, we propose the development of automated tools that can analyze codebases, suggest optimal splitting strategies, and handle the intricate process of dynamic bundle generation. The benefits of this approach are particularly pronounced for complex, feature-rich web applications. By loading only the code that's immediately necessary and intelligently preloading likely-to-be-needed modules, we can significantly improve initial load times and application responsiveness. This technique also has the potential to reduce overall bandwidth usage, as users download only the code they actually need.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 9, September 2023
DOI: 10.17148/IJARCCE.2023.12925

G. Intelligent Resource Orchestration

The intelligent resource orchestration system we propose takes a holistic approach to managing the loading and execution of various web resources. Unlike traditional resource loading strategies that often follow a fixed, predetermined order, our system dynamically adjusts its strategy based on real-time conditions and application requirements. At the heart of this system is the Orchestration Decision Engine, which coordinates the loading and execution of resources based on input from several specialized components:

1. Priority Assignment Engine: This component analyzes each resource to determine its criticality for the current user journey. It considers factors such as the resource's impact on visual completeness, interactivity, and core functionality.
2. Network Condition Monitor: By continuously assessing the quality and stability of the user's network connection, this component allows the system to adapt its strategy for different network environments. For example, in poor network conditions, it might prioritize loading critical CSS and defer large images.
3. Device Capability Analyzer: This component evaluates the capabilities of the user's device, including CPU power, available memory, and GPU capabilities. This information helps optimize resource allocation and prevent overloading less powerful devices.
4. Application State Tracker: By monitoring the current state of the application and user interactions, this component helps predict which resources are likely to be needed next.

The Orchestration Decision Engine uses advanced algorithms to process inputs from these components and determine the optimal loading strategy. This strategy is not static; it evolves in real-time as conditions change and the user progresses through their journey.

One of the key innovations in this system is its ability to make intelligent trade-offs. For example, if a critical JavaScript file is taking too long to load due to poor network conditions, the system might decide to inline a crucial portion of the script to improve Time to Interactive, while continuing to load the full file in the background.

The system also employs sophisticated caching and preloading strategies. By lyzing global usage patterns, it can identify resources that are frequently used across different sections of the application and preload them during idle times.Implementing this system presents several challenges, including the need for careful instrumentation of the application to provide accurate inputs to the decision engine. Additionally, there's a risk of increased complexity in debugging and performance profiling. To address these challenges, we propose the development of specialized developer tools that provide visibility into the orchestration process and help identify optimization opportunities.

The potential benefits of this approach are substantial. By intelligently managing resource loading and execution, we can significantly improve key performance metrics such as Time to Interactive (TTI) and First Input Delay (FID). This, in turn, leads to a more responsive and engaging user experience, particularly on mobile devices where resources are often constrained.

H. Adaptive Rendering Strategies

To address the diverse range of mobile devices and network conditions, we propose an adaptive rendering strategy that dynamically switches between server-side rendering, client-side rendering, and hybrid approaches. This technique builds upon the work of Fink et al. [11] by incorporating real-time performance monitoring and machine learning-based decision making.

The adaptive rendering system operates as follows:

1. Upon receiving an initial request, the system analyzes the user's device capabilities and current network conditions.
2. It assesses the complexity of the requested application or page, considering factors such as the amount of dynamic content, interactivity requirements, and data dependencies.
3. Based on this analysis, the system selects an initial rendering strategy. For example:
   - For low-powered devices on slow networks, it might choose server-side rendering to minimize client-side processing.
   - For high-end devices on fast connections, it could opt for client-side rendering
4. As the user interacts with the application, the system continuously monitors performance metrics such as Time to First Byte (TTFB), First Contentful Paint (FCP), and Time to Interactive (TTI).

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 9, September 2023

DOI: 10.17148/IJARCCE.2023.12925

5. If performance degrades or network conditions change significantly, the system can dynamically adjust its rendering strategy. For instance, it might switch from client-side to server-side rendering if the network becomes congested, or vice versa if conditions improve.

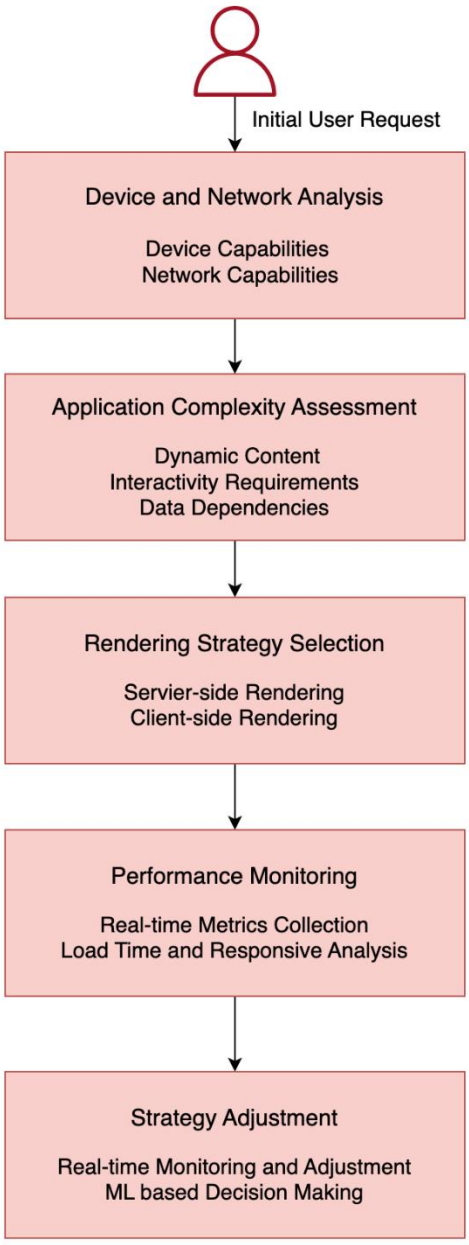

Fig. 3 Adaptive rendering system

The adaptive rendering system leverages machine learning algorithms to improve its decision-making over time. By analyzing aggregated performance data across many users and sessions, the system can refine its strategy selection criteria and better predict which rendering approach will yield the best results for a given combination of device, network, and application characteristics.

One of the key challenges in implementing this system is ensuring a smooth transition between rendering modes without disrupting the user experience. To address this, we propose developing a set of transition strategies that can seamlessly blend different rendering approaches. For example, when switching from server-side to client-side rendering, the system could employ a progressive enhancement technique, gradually adding interactivity as resources are loaded on the client.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 9, September 2023
DOI: 10.17148/IJARCCE.2023.12925

The benefits of this adaptive approach are particularly significant in mobile environments, where device capabilities and network conditions can vary widely. By dynamically selecting the most appropriate rendering strategy, we can ensure optimal performance across a broad spectrum of scenarios, from low-end devices on 2G networks to high-end smartphones on 5G connections.

IV. EMERGING TRENDS AND CONFIGURATIONS IN MOBILE WEB PERFORMANCE

The landscape of mobile web performance is rapidly evolving, driven by advancements in technology and changing user expectations. One of the most significant developments is the rollout of 5G networks, which promises to revolutionize mobile connectivity with theoretical speeds up to 20 Gbps and significantly reduced latency [19]. While this opens up new possibilities for rich, interactive mobile web experiences, it also presents new challenges. Developers must balance leveraging the increased bandwidth with maintaining performance across all network conditions, as not all users will have access to 5G. Moreover, the reduced latency of 5G makes techniques like predictive preloading even more effective, as resources can be fetched almost instantaneously.

Simultaneously, the mobile device ecosystem is becoming increasingly diverse, ranging from IoT devices and wearables to foldable smartphones and augmented reality headsets. This diversity necessitates a more adaptive approach to web performance optimization. For instance, foldable devices require responsive designs that can adapt in real-time to changing screen sizes and orientations. Wearables and IoT devices often have limited processing power and memory, requiring careful optimization of resource usage. On the other hand, high-end smartphones and AR/VR devices demand high frame rates and minimal latency for smooth user experiences. This spectrum of devices presents both challenges and opportunities for web developers to create versatile, high-performance applications.

Artificial Intelligence and Machine Learning are increasingly being applied to web performance optimization, offering new ways to predict user behavior, optimize resource loading, and adapt content delivery based on device capabilities and network conditions. Web Assembly (Wasm) is emerging as a game-changer, allowing complex computations to run at near-native speed in the browser and enabling high-performance web applications on mobile devices. Progressive Web Apps (PWAs) continue to evolve, blurring the line between web and native applications with advanced caching strategies, offline capabilities, and seamless updates. These technologies are pushing the boundaries of what's possible in mobile web performance, enabling more sophisticated and responsive web applications.

As these trends converge, the future of mobile web performance optimization lies in solutions that can seamlessly adjust to this diverse and rapidly evolving ecosystem. Developers will need to leverage a combination of cutting-edge technologies, adaptive optimization techniques, and intelligent, data-driven strategies to ensure fast, responsive, and engaging web experiences across all devices and contexts. Moreover, with increasing focus on user privacy, performance optimization techniques are adapting to work within more restrictive data collection environments, emphasizing on-device processing and differential privacy approaches. The challenge moving forward will be to balance the demands of high performance with the imperatives of user privacy and device diversity.

V. CONCLUSION AND FUTURE WORK

The optimization of mobile web performance remains a critical challenge in the ever-evolving landscape of web technologies. This paper has presented a comprehensive review of existing research in the field and proposed several advanced techniques for enhancing mobile web performance. Our proposed methods, including adaptive server push, granular code splitting with predictive loading, intelligent resource orchestration, adaptive rendering strategies, and edge-enhanced acceleration, offer promising avenues for significantly improving the user experience on mobile devices.

These techniques address the multifaceted challenges of mobile web performance, from network latency and device limitations to the increasing complexity of web applications. The adaptive server push mechanism leverages machine learning to optimize resource delivery, while granular code splitting and predictive loading ensure that only necessary code is transmitted and executed. Intelligent resource orchestration provides a holistic approach to managing web resources, adapting to real-time conditions for optimal performance. Adaptive rendering strategies cater to the diverse range of mobile devices and network conditions, and edge-enhanced acceleration leverages emerging edge computing capabilities to reduce latency and offload computation.

As we look to the future, the landscape of mobile web performance is set to be transformed by emerging technologies and trends. The rollout of 5G networks, the increasing diversity of mobile devices, and the advent of technologies like Web Assembly and Progressive Web Apps present both opportunities and challenges for performance optimization.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 9, September 2023
DOI: 10.17148/IJARCCE.2023.12925

Artificial Intelligence and Machine Learning are poised to play an increasingly significant role in predictive optimization and adaptive content delivery. Moreover, the growing emphasis on user privacy necessitates new approaches to performance optimization that respect and protect user data.

Future research in this field should focus on several key areas:

1. Developing more sophisticated machine learning models for predictive resource loading and adaptive optimization that can operate effectively across a wide range of devices and network conditions.
2. Exploring the intersection of edge computing and mobile web performance, including novel approaches to distributed rendering and computation offloading.
3. Investigating techniques for balancing high performance with user privacy, particularly in light of evolving data protection regulations and the phasing out of third-party cookies.
4. Adapting web performance optimization techniques for emerging device categories, such as foldable devices, augmented reality headsets, and IoT devices.
5. Developing standardized benchmarks and metrics for evaluating mobile web performance across the increasingly diverse device ecosystem.
6. Exploring the potential of Web Assembly and other emerging technologies to enable near-native performance for complex web applications on mobile devices.

In conclusion, the pursuit of optimal mobile web performance requires a holistic approach that combines server-side optimizations, client-side techniques, and intelligent resource management. By continuing to innovate and adapt to the changing technological landscape, we can ensure that the mobile web remains a fast, responsive, and accessible platform for users worldwide. As we move forward, the challenge will be to create web experiences that are not only performant but also respectful of user privacy, adaptive to diverse devices, and capable of leveraging cutting-edge network technologies. The techniques and insights presented in this paper provide a foundation for addressing these challenges and pave the way for future advancements in mobile web performance optimization.

REFERENCES

[1] F. Xu, Y. Liu, T. Moscibroda, R. Chandra, L. Jin, Y. Zhang, and Q. Li, "Optimizing background email sync on smartphones," in Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services, 2013, pp. 55-68.
[2] Z. Wang, F. X. Lin, L. Zhong, and M. Chishtie, "Why are web browsers slow on smartphones?," in Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, 2011, pp. 91-96.
[3] S. Sundaresan, N. Feamster, R. Teixeira, and N. Magharei, "Measuring and mitigating web performance bottlenecks in broadband access networks," in Proceedings of the 2013 Conference on Internet Measurement Conference, 2013, pp. 213-226.
[4] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall, "Demystifying page load performance with WProf," in Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation, 2013, pp. 473-485.
[5] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, "Understanding website complexity: measurements, metrics, and implications," in Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, 2011, pp. 313-328.
[6] A. Sivakumar, S. Puzhavakath Narayanan, V. Gopalakrishnan, S. Lee, S. Rao, and S. Sen, "PARCEL: Proxy assisted browsing in cellular networks for energy and latency reduction," in Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, 2014, pp. 325-336.
[7] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale RDMA deployments," in Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, 2015, pp. 523-536.
[8] V. Agababov, M. Buettner, V. Chudnovsky, M. Cogan, B. Greenstein, S. McDaniel, M. Piatek, C. Scott, M. Welsh, and B. Yin, "Flywheel: Google's data compression proxy for the mobile web," in Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation, 2015, pp. 367-380.
[9] A. Pamboris and P. Pietzuch, "EdgeReduce: Eliminating mobile network traffic using application-specific edge proxies," in Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services, 2015, pp. 57-70.
[10] A. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate record-and-replay for HTTP," in Proceedings of the 2015 USENIX Annual Technical Conference, 2015, pp. 417-429.

 This work is licensed under a Creative Commons Attribution 4.0 International License

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 9, September 2023

DOI: 10.17148/IJARCCE.2023.12925

[11] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate record-and-replay for HTTP," in Proceedings of the 2015 USENIX Annual Technical Conference, 2015, pp. 417-429.

[12] M. Varvello, J. Blackburn, D. Naylor, and K. Papagiannaki, "EYEORG: A platform for crowdsourcing web quality of experience measurements," in Proceedings of the 12th International on Conference on emerging Networking Experiments and Technologies, 2016, pp. 399-412.

[13] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The cost of the "S" in HTTPS," in Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, 2014, pp. 133-140.

[14] A. Erbad and C. Williamson, "Proxy-based acceleration of mobile web browsing," in Proceedings of the 29th International Conference on Data Engineering Workshops (ICDEW), 2013, pp. 229-234.

[15] U. Iqbal, Z. Shafiq, and Z. Qian, "The ad wars: Retrospective measurement and analysis of anti-adblock filter lists," in Proceedings of the 2017 Internet Measurement Conference, 2017, pp. 171-183.

[16] T. Høiland-Jørgensen, B. Ahlgren, P. Hurtig, and A. Brunstrom, "Measuring latency variation in the internet," in Proceedings of the 12th International on Conference on emerging Networking Experiments and Technologies, 2016, pp. 473-480.

[17] K. Zarifis, M. Holland, M. Jain, E. Katz-Bassett, and R. Govindan, "Modeling HTTP/2 speed from HTTP/1 traces," in International Conference on Passive and Active Network Measurement, 2016, pp. 233-247.

[18] M. Saxena, U. Sharan, and S. Fahmy, "Analyzing video performance in modern web browsers," in Proceedings of the 10th ACM Multimedia Systems Conference, 2019, pp. 907-912.

[19] A. Jangda, B. Powers, E. D. Berger, and A. Guha, "Not so fast: Analyzing the performance of WebAssembly vs. native code," in 2019 USENIX Annual Technical Conference, 2019, pp. 107-120.

 This work is licensed under a Creative Commons Attribution 4.0 International License